Proceedings of the
ACM SIGKDD 2013 Full-day Workshop on

Interactive Data Exploration and Analytics

IDEA

2013

Chicago, Illinois, USA
Aug 11, 2013

poloclub.gatech.edu/idea2013

Editors & Organizers: Polo Chau, Jilles Vreeken, Matthijs van Leeuwen, Christos Faloutsos

Proceedings of the
ACM SIGKDD Workshop on
Interactive Data Exploration and Analytics

These proceedings are also included in the ACM Digital Library.

IDEA'13, August 11, 2013, Chicago, IL, USA

ACM SIGKDD Workshop on Interactive Data Exploration and Analytics

General Chairs

Duen Horng (Polo) Chau (Georgia Tech)
Jilles Vreeken (University of Antwerp)
Matthijs van Leeuwen (KU Leuven)
Christos Faloutsos (Carnegie Mellon University)

Program Committee

Adam Perer (IBM, USA)
Albert Bifet (Yahoo! Labs, Barcelona, Spain)
Aris Gionis (Aalto University, Finland)
Arno Knobbe (Leiden University, the Netherlands)
Chris Johnson (University of Utah, USA)
Cody Dunne (UMD, USA)
David Gotz (IBM, USA)
Geoff Webb (Monash University, Australia)
George Forman (HP Labs)
Hanghang Tong (City University of New York)
Jaakko Hollmen (Aalto University, Finland)
Jacob Eisenstein (Georgia Tech)
Jaegul Choo (Georgia Tech)
Jiawei Han (University of Illinois at Urbana-Champaign)
Jimeng Sun (IBM, USA)
John Stasko (Georgia Tech)
Kai Puolamäki (Finnish Institute of Occupational Health, Finland)
Katharina Morik (TU Dortmund, Germany)
Kayur Patel (Google)
Leman Akoglu (Stony Brook University)
Mario Boley (Fraunhofer IAIS, University of Bonn)
Marti Hearst (UC Berkeley, USA)
Martin Theobald (University of Antwerp, Belgium)
Nan Cao (IBM, USA)
Naren Ramakrishnan (Virginia Tech, USA)
Nikolaj Tatti (Aalto University, Finland)
Parikshit Ram (Georgia Tech, USA)
Pauli Mietinnen (Max Planck Institute for Informatics, Germany)
Saleema Amershi (Microsoft Research)
Tijl De Bie (University of Bristol, UK)
Tim (Jia-Yu) Pan (Google)
Tina Eliassi-Rad (Rutgers)
Tino Weinkauf (Max Planck Institute for Informatics, Germany)
Toon Calders (Université Libre de Bruxelles, Belgium)
Zhicheng 'Leo' Liu (Stanford)

External Reviewer

Umashanthi Pavalanathan (Georgia Tech)

Preface

We have entered the era of big data. Massive datasets, surpassing terabytes and petabytes in size are now commonplace. They arise in numerous settings in science, government, and enterprises, and technology exists by which we can collect and store such massive amounts of information. Yet, making sense of these data remains a fundamental challenge. We lack the means to exploratively analyze databases of this scale. Currently, few technologies allow us to freely "wander" around the data, and make discoveries by following our intuition, or serendipity. While standard data mining aims at finding highly interesting results, it is typically computationally demanding and time consuming, thus may not be well-suited for interactive exploration of large datasets.

Interactive data mining techniques that aptly integrate human intuition, by means of visualization and intuitive **human-computer interaction** techniques, and **machine computation** support have been shown to help people gain significant insights into a wide range of problems. However, as datasets are being generated in larger volumes, higher velocity, and greater variety, creating effective interactive data mining techniques becomes a much harder task.

It is exactly this research, experiences and practices that we want to discuss at IDEA, the workshop on Interactive Data Exploration and Analytic. In a nutshell, IDEA addresses the development of data mining techniques that allow users to interactively explore their data. We focus and emphasize on **interactivity** and effective **integration** of techniques from **data mining**, **visualization** and **human-computer interaction**. In other words, we explore how the best of these different but related domains can be combined such that the *sum is greater than the parts*.

The main program of IDEA'13 consists of eleven papers covering various aspects of interactive data exploration and analytics. These papers were selected from a total of 25 submissions after a thorough reviewing process. We sincerely thank the authors of the submissions and the attendees of the workshop. We wish to thank the members of our program committee for their help in selecting a set of high-quality papers. Furthermore, we are very grateful to Marti Hearst and Haesun Park for keynote presentations about their recent work on interactive data exploration and visualization.

Polo Chau & Jilles Vreeken & Matthijs van Leeuwen & Christos Faloutsos

Antwerp, July 2013

Table of Contents

Invited Talk

Interactive Visual Analytics for High Dimensional Data

Haesun Park
School of Computational Science and Engineering
Georgia Institute of Tech, Atlanta
hpark@cc.gatech.edu

Abstract

Many modern data sets can be represented in high dimensional vector spaces and have benefited from computational methods that utilize advanced techniques from numerical linear algebra and optimization. Visual analytics approaches have contributed greatly to data understanding and analysis due to utilization of both automated algorithms and human's quick visual perception and interaction. However, visual analytics targeting high dimensional large-scale data has been challenging due to low dimensional screen space with limited pixels to represent data. Among various computational techniques supporting visual analytics, dimension reduction and clustering have played essential roles by reducing the dimension and volume to visually manageable scales.

In this talk, we present some of the key foundational methods for supervised dimension reduction such as linear discriminant analysis (LDA), dimension reduction and clustering/topic discovery by nonnegative matrix factorization (NMF), and visual spatial alignment for effective fusion and comparisons by Orthogonal Procrustes. We demonstrate how these methods can effectively support interactive visual analytic tasks that involve large-scale document and image data sets.

Bio

Prof. Haesun Park received her B.S. degree in Mathematics from Seoul National University, Seoul Korea, in 1981 with summa cum laude and the University President's Medal for the top graduate, and her M.S. and Ph.D. degrees in Computer Science from Cornell University, Ithaca, NY, in 1985 and 1987, respectively. She has been a professor in the School of Computational Science and Engineering at the Georgia Institute of Technology, Atlanta, Georgia since 2005. Before joining Georgia Tech, she was on faculty at University of Minnesota, Twin Cities, and program director at the National Science Foundation, Arlington, VA. She has published extensively in the areas including numerical algorithms, data analysis, visual analytics, text mining, and parallel computing. She has been the director of the NSF/DHS FODAVA-Lead (Foundations of Data and Visual Analytics) center and executive director of Center for Data Analytics at Georgia Tech. She has served on numerous editorial boards including IEEE Transactions on Pattern Analysis and Machine Intelligence, SIAM Journal on Matrix Analysis and Applications, SIAM Journal on Scientific Computing, and has served as a conference co-chair for SIAM International Conference on Data Mining in 2008 and 2009. In 2013, she was elected as a SIAM Fellow.

IDEA'13, August 11, 2013, Chicago, IL, USA.
ACM 978-1-4503-2329-1.

Building Blocks for Exploratory Data Analysis Tools

Sara Alspaugh
UC Berkeley
alspaugh@eecs.berkeley.edu

Archana Ganapathi
Splunk, Inc.
aganapathi@splunk.com

Marti A. Hearst
UC Berkeley
hearst@berkeley.edu

Randy Katz
UC Berkeley
randy@eecs.berkeley.edu

Abstract

Data exploration is largely manual and labor intensive. Although there are various tools and statistical techniques that can be applied to data sets, there is little help to identify what questions to ask of a data set, let alone what domain knowledge is useful in answering the questions. In this paper, we study user queries against production data sets in Splunk. Specifically, we characterize the interplay between data sets and the operations used to analyze them using latent semantic analysis, and discuss how this characterization serves as a building block for a data analysis recommendation system. This is a work-in-progress paper.

1. INTRODUCTION

Visual data exploration is a key part of data analysis, but it remains ad hoc, requiring intensive manual intervention to identify useful information. Although many tools exist for cleaning and visualizing data, the intermediate step of determining which questions to ask and how to visualize their results requires a combination of deep domain knowledge and a willingness to try many approaches. For users without domain knowledge who are tasked with gleaning insights from a mass of data, the first step to understanding which aspects of the data are important, interesting, or unusual often requires involves iteratively applying standard techniques and interpreting results. Users with domain knowledge or very specific questions in mind already know which techniques yield important insights and based on the semantics of their problem and their data. As a result, domain knowledge lets users consider fewer possibilities and quickly progress to deeper questions.

Leveraging the facts that:

(1) data exploration often involves applying common techniques, therefore different data exploration scenarios typically have overlapping analysis steps, and
(2) domain experts are likely to have a better idea a priori what is important to investigate, limiting the amount of work they have to do to reach later stages of analysis,

our goal is to build a tool that could make intelligent recommendations to users as to which data exploration actions to take. The anticipated result is that users with time constraints or without domain knowledge can make quick and efficient work of data exploration and be well on their way to formulating hypotheses, assessing assumptions, and planning their modeling approach.

This tool would make data exploration more efficient by drawing user attention to more fruitful paths of inquiry and providing users with intuitive control over exploration operations. Such a tool should draw on a combination of classic domain-agnostic exploratory data analysis techniques and more domain-specific techniques learned by observing what other users with similar data sets found useful. It should use these along with modern methods for determining when the information resulting from a technique was interesting or unexpected [16]. It should also provide intuitive mechanisms for interacting with the output of such techniques for hands-on exploration. Making high-quality suggestions regarding which actions to take and reacting intuitively to user input requires a thorough qualitative and quantitative understanding of the data analysis process. Such suggestions are prerequesites to applying recommendation, searching, and ranking techniques to the problem of efficient exploratory data analysis.

As a first step toward creating such a tool, we study records of data analysis activity from Splunk users. (Splunk is a commercial data analysis product.) The logs contain queries from thousands of users, written in the Splunk query language. These logs provide information only about what a certain set of users, primarily IT professionals, do with certain types of data, primarily IT system data. Moreover, they represent only a portion of the actions the users perform with one particular tool, and as such, only tell us about a particular portion of the data analysis process. Nonetheless, these queries provide a valuable starting point to quantitatively characterize the data exploration process. This characterization is a precursor to acquiring the intuition and data needed to build an intelligent data exploration recommendation tool.

In this paper, we focus on the interplay between certain data set features and the analysis operations applied to those data sets, and discuss how this information could be used by a recommendation tool. In particular, we adopt a technique from natural language processing and use it to measure similarity between data set column names and the ways in which they are used. We then discuss how such a technique could be extended for use by intuitive, intelligent data exploration

IDEA'13, August 11th, 2013, Chicago, IL, USA.

tools. We begin, in Section 2, by providing context for the vision we outlined above and relate this idea to existing tools. We describe Splunk and the data set we explore in Section 3. In Section 4, we elaborate on the methodology we use analyzing the ways different types of data is used and present our main result. We discuss future directions in Section 5. Lastly, in Section 6, we outline the goals for building a recommendation system for exploratory data analysis.

2. RELATED WORK

The field of information visualization and data exploration is very large; this section focuses on related work on tools for automated suggestions for portions of the exploratory data analysis process. Many intuitive user interface features that would be ideal to have for an exploratory data analysis tool are available in Tableau [1] which is descended from earlier research in exploratory data analysis such as Polaris [14] and Mackinlay's earlier work [9]. Tableau includes a "Show Me" feature, which is meant to reduce the burden on users by automatically creating a good visualization of a view of their data. The user selects a tuple of columns, and Tableau returns a list of possible visualizations templates that could be used with these columns. It does this by automatically detecting whether columns are numerical or categorical, and suggesting all visualizations which are compatible with the selected column types. However, despite the ease with which users can use Tableau to create attractive visualizations, even with the "Show Me" feature, they are still left with the task of manually indicating which columns of the data to visualize and which of several visualization templates to use to visualize them, and this is done one tuple at a time. For large data sets which have many aspects to explore, this overhead could be significant.

The vision for VizDeck is very similar to that which we have described here in that it aims to automatically suggests visualizations for exploratory data analysis [8]. VizDeck supports seven visualization types, such as histograms and scatter plots, and a few transformations, such as filter. It generates all type-compatible visualizations for a given data set, then ranks them. VizDeck ranks visualizations by recording a few statistical measures for each visualization when a user indicates that a certain visualization is interesting, for example, the entropy of the columns involved. It then attempts to predict unseen scores from these statistical measures based on their similarity to other highly ranked visualizations. This differs from our strategy, which, as described in Section 5, is to predict what transformations or visualizations are desirable for a data set based on its similarity to other data sets for which transformations and visualizations have been recorded. Moreover, while generating all type-compatible visualizations works for small data sets, this approach may not scale to larger data sets and larger sets of visualizations. (See Section 6 for a discussion of this.) However, it is valuable to compare to and build from the techniques used in this and the systems described above.

SAGE is a collection of tools for constructing graphical designs by selecting graphical elements, mapping them to data, combining them, and for finding and customizing prior designs. [11]. Published work on SAGE touches upon a number of ideas related to those we present here. For example, users of SAGE can partially sketch out designs in a number of ways and SAGE attempts to intelligently infer remaining properties, like the mapping of data to visual elements. In addition, SAGE provides users the ability to browse and search related designs based on criteria such as the graphical elements used and the data types of the data mapped to these graphical elements.

DataWrangler is a tool for facilitating data cleaning, in particular, data reformatting [6]. Although this use case is slightly different and possibly subsumed by exploratory data analysis, DataWrangler is relevant in that it uses statistics to disambiguate user actions and rank the most likely desired operation as suggestions to the user. It defines a set of data cleaning transformations and upon user input, provides some guesses as to which of these transformations the user might want to take to re-format the data.

In a similar vein, Profiler is a tool for data quality assessment, for instance, finding missing values, outliers, and misspellings [7]. Profiler is related because data quality assessment is again a subset of exploratory data analysis, and because it provides automatic visualization suggestions for anomaly assessment.

Earlier work took more of an artificial intelligence approach, rather than a data-centric approach. Schiff developed a method to automatically create cognitively motivated visualizations from first principles using very small data sets [12]. St. Amant and Cohen developed a knowledge-based planning system called AIDE to help users strike a balance between conventional statistical packages and automated systems which do not ask the user for any kind of input [13]. Casner automatically designed graphic presentations based on an analysis of a logical description of a task the user was attempting to undertake [4]. More recently, other authors have used an ontology to enumerate valid data mining processes [2].

A common feature of these systems is that they attempt to provide intelligent suggestions or recommendations as part of a broader purpose. As the literature on techniques recommendation systems is very large, the techniques used to date by exploratory data analysis systems represent only a small fraction of proposed approaches [5]. However, they do represent some of the first few attempts to apply recommendation in the domain of visualization.

3. CASE STUDY: SPLUNK QUERIES

To begin to quantitatively describe our point in the space of data analysis pipelines, we collected queries from users of Splunk[1]. Splunk is a platform for indexing and analyzing large quantities of data from heterogeneous data sources, especially machine-generated logs. Customers use Splunk for a variety of data analysis needs, including root cause failure detection, web analytics, A/B testing and product usage statistics. Consequently, the types of data sets indexed in Splunk also span a wide range, such as system event logs, web access logs, customer records, call detail records, and product usage logs.

3.1 Definitions and Overview

Table 1 lists the terminology and definitions introduced in this section. To use Splunk, the user indicates where the data they want Splunk to collect and index is located. For example, data might be in a log directory on a file system or collected from a remote server via a certain port. Upon collection, Splunk organizes this data temporally into

[1] `www.splunk.com`

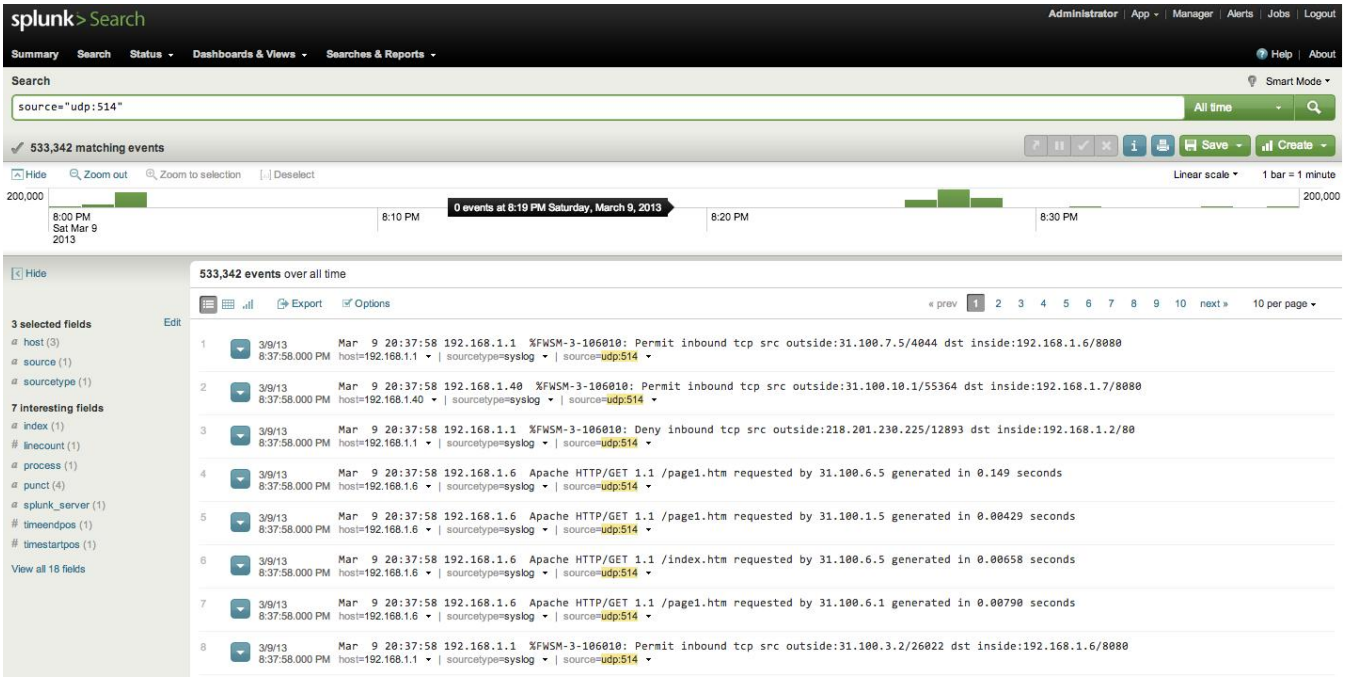


Figure 1: The default GUI view displays the first several events indexed, with extracted fields highlighted on the side, and a histogram of the number of events over time displayed along the top. The user types their query into the search bar at the top of this view.

events by delineating events based on their timestamp, and processes these events using a MapReduce-like architecture, details of which can be found in [3]. Splunk does not require that the user specify a schema for this data, as much log data is semi-structured or unstructured, and there is often no notion of a schema that can be imposed on the data a priori. Rather, *fields* and *values* are extracted from events at run time based on the *source type*. Specifically, when a user defines a new source type, Splunk guides the user in constructing regular expressions to extract fields and values from each incoming raw event. Splunk includes a query language for searching and manipulating data and a graphical user interface (GUI) with tools for visualizing query results. The queries we collected were written in this language.

Users almost always compose such queries in the GUI. The default GUI view displays the first several events indexed, with extracted fields highlighted on the left hand side, and a histogram of the number of events over time displayed along the top. A screen shot of this default view is shown in Figure 1. The user types their query into the search bar at the top of this view. The query consists of a set of *stages* separated by the pipe character, and each stage in turn consists of a *command* and *arguments*. We refer to a query with arguments removed as a query *template*. Splunk passes events through each stage of a query. Each stage filters, transforms or enriches data it receives from the previous stage, and pipes it to the subsequent stage in the query, updating the results that are displayed to the user as they are processed. A simple example of a query is a plain text search for specific strings or matching field-value pairs. A more complex example can perform more advanced operations, such as clustering the data using k-means.

We manually grouped all of the operations users perform in Splunk into a set of *use cases*. By use case, we mean an abstract way in which an argument can be used. For instance, an argument can be used as a filter, aggregated, sorted on, or grouped by, and each of these are use cases. Table 2 lists all use cases we encountered, along with examples and the shorthand labels we use to refer to each. Note that these use cases do not represent all possible use cases in Splunk, but rather are just those we encountered in our examination of a subset of arguments used in the queries.

When the user enters a query that performs a filter, the GUI updates to display events which pass through the filter. When the user uses a query to add or transform a field, the GUI displays events in updated form. Most queries result in visualizations such as tables, time series, and histograms, that appear in the GUI when the query is executed. Users can specify these visualizations by typing queries in the default view, as described above, and can also create "apps," which are custom views that display the results of pre-specified queries, possibly in real time, which is useful for things like monitoring and reporting. Although the visualizations users can create in Splunk do not represent the full breadth of all possible visualizations, they still capture a useful set of standard and commonly used ones.

3.2 Splunk Query Details

The Splunk query language is modeled after the Unix `grep` command and pipe operator. Below is an example query that provides a count of errors by detailed status code:

`search` error `|` `stats count by` status `|` `lookup` statuscodes status `OUTPUT` statusdesc

In this example, there are three stages; `search`, `stats`, and

Term	*Definition*
event	a raw, timestamped item of data indexed by Splunk, similar to a tuple or row in databases
field	a key corresponding to a value in an event, similar to the concept of a column name
value	part of an event corresponding to a certain field, similar to a particular column entry in a particular row
query	a small program written in the Splunk query language, consisting of pipelined stages
stage	a portion of a query syntactically between pipes and conceptually a single action or operation
command	the part of a stage that indicates what action or operation to perform on the data
operation	an abstract category made of similar actions or commands, for example, filter or aggregate are operations
argument	the parts of a stage that indicate what fields, values, or option values to use with a command
use case	an extension of the concept operation to account for how an argument is used in an operation
template	a query string with arguments removed, like a skeleton or scaffolding

Table 1: Terminology describing Splunk data.

Use case	*Example*	*Label*
filter events containing *value*	`search` *value*	FILTER VALUE
filter events on values of *field*	`search` *field* `=200`	FILTER ON FIELD
queries sorted by *field*	`sort` *field*	SORT BY
field projected	`table` *field*	PROJECT
field renamed as *field*	`rename foo as` *field*	RENAME
field passed to `top`	`top` *field*	TOP
field aggregated	`stats count` *field*	AGGREGATE
grouped by *field*	`stats count foo by` *field*	GROUP BY
field used as function domain	`eval` *field*`=(foo+bar)/1024`	DOMAIN
arg used in arithmetic transformation	`eval foo=`*arg*`*100`	ARITHMETIC
arg used in conditional	`eval foo=case(bar>bat, bar,`*arg*`)`	CONDITIONAL
field used in other transformation	`eval foo=tonumber(`*field*`)`	FIELD IN TRANSFORMATION
value used in other transformation	`eval foo=replace(bar,`*value*`)`	VALUE IN TRANSFORMATION
value passed to option	`head limit=`*value*	OPTION

Table 2: Use cases, examples, and the labels for each in Figure 4. For each argument, the frequency with which it appeared in each use case was tallied up and applied in the LSA calculation described.

`lookup` are the commands in each stage, `count by` and `OUTPUT` are functions and option flags passed to these commands, and "error", "status", "statuscodes", and "statusdesc" are arguments. In particular, "status" and "statusdesc" are fields.

To see how this query operates, consider the following toy data set:

0.0	-	**error**	404
0.5	-	OK	200
0.7	-	**error**	500
1.5	-	OK	200

The first stage filters out all events not containing the word "error". After this stage, the data looks like:

0.0	-	**error**	404
0.7	-	**error**	500

The second stage aggregates events by applying the `count` function over events grouped according to the "status" field, to produce the number of events in each "status" group.

count	status
1	404
1	500

The final stage performs a join on the "status" field between the data and an outside table that contains descriptions of each of the codes in the "status" field, and puts the corresponding descriptions into the "statusdesc" field.

count	status	statusdesc
1	404	Not Found
1	500	Internal Server Error

We collected 50,000 such queries from users of a cloud installation of Splunk. The data set consists of a list of query strings along with an anonymized unique identifier for the issuing user and the time of issuance. Table 3 summarizes some basic information about this query set. To

Total users	602
Total queries	50,000
Unique queries	7,012
Parseable unique queries	5,813

Table 3: Characteristics of the set of queries analyzed. These metrics give some measures of the size of the data set. Note that the approximately 17% of the data set that is not parseable are queries that contain infrequently used commands or queries that are malformed. We plan to extend the parser to support these infrequently used commands in the near future.

parse these queries, we custom-wrote an open source parser for this query language so that it could be separately available from the closed-source Splunk system. This parser is capable of parsing 63 most common commands of 128 seen in our dataset, thus parsing a large proportion of gathered queries.

It is important to note that we do not have access to any information about the data sets over which the queries were issued because these data sets are proprietary and thus unavailable. Having access only to query logs is a common occurrence for data analysis, and algorithms that can work under these circumstances are therefore important to develop. Further, by manually inspecting the queries and using them to partially reconstruct some data sets using the fields and values mentioned in the queries, we are fairly certain that these queries were issued over multiple different (albeit similar) sources of data, thus suggesting the results presented here will generalize across different datasets.

4. CHARACTERIZING DATA ANALYSES

At a high level, the recommendation problem is: given a data set to explore, what visualizations and other such information should the tool suggest to the user? This problem can be viewed abstractly as one of finding the right mapping from points in the space of all possible data sets to points in the space of all possible exploratory data analysis visualizations; however, this formulation is a difficult multi-class multi-label classification problem. Moreover, we currently lack the data to solve this problem as stated above since we don't have access to full user data sets, nor the "correct" labels, which are in this case, complete and user-approved exploratory data analysis visualizations. However, the user query data from Splunk still allows us to explore one key idea regarding the solution of this problem, which is that users are likely to explore and analyze semantically similar data sets in similar ways. This suggests that a reasonable strategy for making exploratory data analysis visualization recommendations for a given data set is to recommend the same things that users found useful for other, semantically similar data sets. We discuss one way of measuring semantic similarity between data sets and the operations applied to them.

4.1 Latent Semantic Analysis

We borrow a technique from natural language processing called latent semantic analysis (LSA) [10]. To use LSA, we start with a matrix of frequencies called the term-document

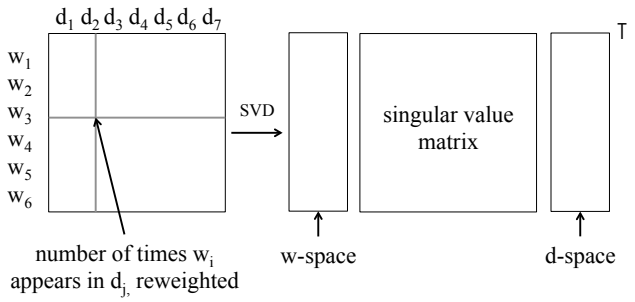


Figure 2: LSA starts with a frequency matrix for which each entry M_{ij} represents a count of the number of times word i appears in document j. After some optional re-weighting, SVD is applied to this matrix to obtain an approximation of the original frequency matrix that expresses relationships of similarity between the words and documents. This similarity between elements is measured by the distance between their corresponding points in the lower-dimensional space.

matrix M. In this matrix, M_{ij} is the number of times term i appeared in document j. It is often useful to re-weight the terms of this matrix using factors like tf-idf [10]. We then find low-rank approximation to this matrix using singular value decomposition (SVD). This yields three matrices $U\Sigma V^T$, which can be used to compute points corresponding to the terms and documents projected onto a lower-dimensional space. A schematic of this process is shown in Figure 2. The reason this is useful is that whereas the original matrix lists only the number of times each term actually appeared in each document, LSA provides access to all terms that are relevant to each document by capturing how similar terms and documents are to one another.

We apply LSA by replacing the term-document matrix with an argument-use case matrix, so that M_{ij} now contains the number of times argument i was used in use case j, where argument and use case are used in the sense defined in Section 3. We consider only the arguments that are used by more than three users and in more than three different query templates. We primarily consider only arguments that are fields (i.e., that represent column names), for reasons that will be clear later. To generate the frequency of arguments in use cases, we first parse each query and extract out all arguments. Then, guided by a classification of Splunk commands into more generic operation categories (e.g., the `search`, `regex`, and `where` commands are classified as filters) we manually inspect each query and develop a set of rules for deciding which use case a given argument occurrence falls into. We then encoded these rules to process the queries and count the the frequency of arguments in use cases. These steps, summarized in Figure 3, yield the desired frequency matrix. The idea is that applying LSA to these frequencies will capture how similar arguments and use cases are to one another.

4.2 Results

The results of applying LSA to our problem after re-expression, as described above, are depicted visually in Figure 4. There are a number of interesting observations. Some

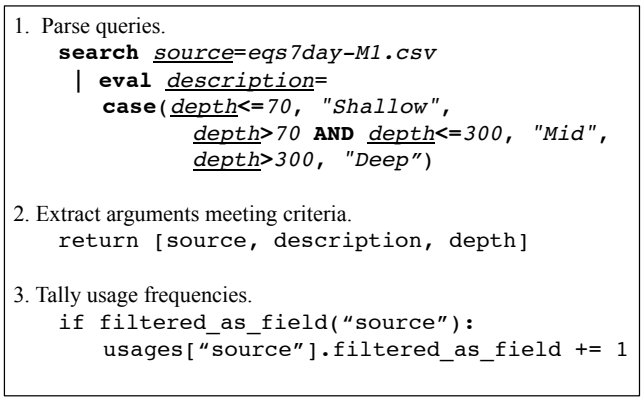


Figure 3: Analyzing queries is challenging because they are essentially small programs. This figure shows the steps we used to compute frequencies from the query data for use with LSA. We first parsed the query to extract arguments. In the figure, arguments are italicized. Those that are fields are also underlined. The rest of the query that is part of the command is in bold. We select arguments that are used by at least three users and in at least three query templates. We then tally the number of times each argument appears in each of the use cases listed in Table 2.

apparently semantically similar arguments are placed close to one another; for example, `uid`, `user_id`, and `user` get placed close together. Some arguments that are used in similar ways are placed close to one another; for example, `60` and `1024`, which are both used to convert values, are close together. Use cases that get used with similar arguments are placed close to one another; for example, TOP and AGGREGATE are placed close together (note that `top` is basically a sorted aggregation), and PROJECT and FILTER ON FIELD are placed closed together. Some arguments are somewhat ambiguous; for example, the types of `login` and `level` are hard to guess, and also, many of these arguments could mean different things in different data sets. Many arguments for which a connection is not immediately apparent get placed close together. This could be because they really are all used similarly, or because we have chosen to project onto a two dimensional space for the purpose of visualization, where a higher dimension would have made this distinction. Lastly, some use cases are placed farther away from the main cluster of points. This is generally because such use cases occur more frequently than the others, and also intuitively means that such use cases are distinguished because they represent an important dimension. Often with LSA the two most frequent and distinct use cases will be place on opposite axes from one another.

While some of these observations could be made from examining the raw frequencies, or from reasoning about the nature of the operations performed, the value of LSA is in uncovering similarity that is not explicitly captured in the raw frequencies. The hypothesis put forth in this section is that the intuition underlying the application of LSA to terms and documents carries over to this domain, and the initial results presented here provide evidence that qualitatively supports this. Thus, the intuition here is that the raw frequencies describing which contexts an argument appears in represents a set of constraints that reflect the similarity of meaning among arguments, and the underlying structure of meaning is revealed by LSA. This suggests that we can learn more from LSA than we could from simply examining raw frequencies and operation types.

To better illustrate and provide context to these results, we highlight a subset of arguments for which we can partially reconstruct plausible example data sets. Two such example data sets, which together include 15 of the arguments that we analyzed, are shown in Table 4. These two data sets are similar in that they both fall under the domain of IT data sets. However, they are still distinct data sets from distinct sources. The idea is that we can leverage the similarities between these data sets to apply operations that make sense even if the data is from a new source we that haven't seen yet.

4.3 Discussion

In order for LSA to work, we needed to make several adjustments to the raw frequencies we computed. In particular, because skew in frequencies or skew in document length (in our case, frequency of use case occurrence) causes undesirable behavior, we had to correct this skew. The undesirable behavior is that most points are "crushed down to zero," that is, almost all of the points in the lower-dimensional space get crushed down to a small space around the origin, with a few points placed far apart down opposite axes. To avoid this, we even out the distribution of frequencies by applying common techniques, including removing arguments that occur too little or too often, re-expressing frequencies using logarithms, and re-weighting frequencies using tf-idf. Such re-expression to find a scaling of the data that produces an even distribution – with points level and symmetrical – is a reoccurring point of emphasis in Tukey's original exposition on exploratory data analysis [15]. Tukey emphasizes the need for re-expression to make it easier to comprehend, especially through visualization:

> "A basic problem about any body of data is to make it more easily handleable by minds. The grasping is with the eye and the better job is through a more symmetric appearance."

This is consistent with our experience, in that without re-expressing the frequencies, LSA yielded a difficult-to-interpret lower-dimensional projection, as described above.

5. NEXT STEPS

We can extend LSA further, adapting an extension called latent semantic indexing (LSI) [10]. Using LSI in the standard way, the lower-dimensional space can be queried using a set of terms to find documents most relevant to them. Doing this requires computing the centroid of the terms in the queries, and returning those documents which are closest to that centroid. We can extend this idea for our problem domain in an analogous way. Instead of using the frequency of arguments in certain use cases as done in the analysis described above, we can use the frequency of arguments in specific functions rather than operations. The functions would be more specific than operations and also callable, so, for example, in place of using AGGREGATE like we do here, we would specify specific aggregating functions like `count` or

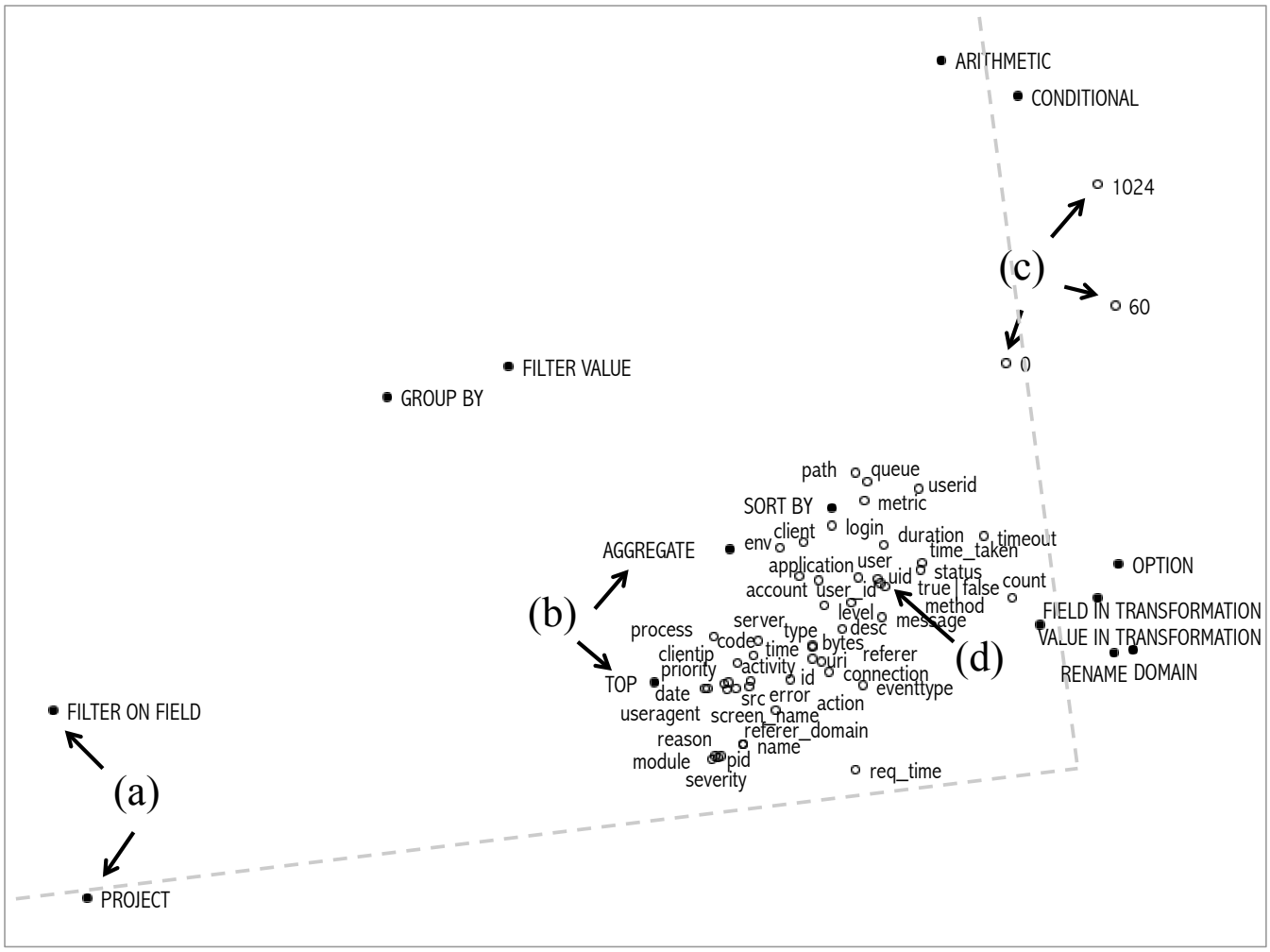


Figure 4: This figure depicts the frequency of arguments (white circles) used in different use cases (black dots) projected onto a two dimensional subspace using SVD. The nearer two points, the more similar they are. We can use this fact to read off some interesting observations about the data from the figure. For example, this figure shows that based on how the arguments are used, the use cases PROJECT and FILTER ON FIELD are similar (a). Likewise, AGGREGATE and TOP are similar, which makes sense as top is a sorted aggregation (b). The numerical arguments `1024` and `0` and `60` are similar, and also are more commonly used as an argument to ARITHMETIC than GROUPED BY (c). Also, the probably-similar arguments `uid`, `user_id`, and `user` get placed close together (d). Lastly, points which get placed further out on an axes, sketched in the figure as dotted lines, are interpreted as being an important dimension, and use cases that are on opposite axes get used with "opposite" arguments, in a way. Thus we see that FILTER ON FIELD is an important use case, and that it tends to get used with different arguments than ARITHMETIC does.

`average`. When a user uploads a data set, we will process that dataset with LSA as well, and then determine which functions to apply to that data set by looking up the column names of that data set using LSI and determining the functions which are closest to them.

This approach suffers from the problem that if a column name does not appear in the term set, it can't be looked up. To get around this, we can extend column names using more context in the form of *fingerprints*. These are extended descriptions of columns that would include, in addition to the the column name, other attributes such as column type, and indicator variables indicating which other columns are present. The fingerprint could also include pipeline context such as previous stages in the query. Then, we will define a new distance function between fingerprints, and use this to lookup the fingerprint or the centroid of fingerprints which are closest to that with the missing column. For example, we would look for matching column types if the queried fingerprint contains a column name that has not been seen before, or look for fingerprints with column names that are near-matches to the column name in the queried fingerprint. A special approach would have to be adopted for operations that take additional arguments, such as filters. One possibility is using partially bound operations for these cases.

Hypothesized data set: Web access logs

sourcetype	_tm	method	stat	useragent	uri
access_*	0	GET	200	Pingdom	/login
access_*	1	GET	404	NewRelic	/apple-touch-
access_*	2	GET	200	Pingdom	/favicon.ico

Hypothesized data set: Mobile metrics

module	metric	device	user
http	memory	ios	foo
mysql	cpu	ios	bar
docs_api	memory	android	bat

Table 4: Two hypothesized data sets partially reconstructed using queries. These are meant as an illustration of how the arguments analyzed relate to the underlying data set.

In addition to extending LSA, there are additional building blocks that we plan as future work to facilitate building such a recommendation system:

- a characterization of the types of operations users perform on system data, to prioritize actions to support for a given domain,
- an identification of common analysis patterns, which suggest what to prioritize so that the common cases for system data analysis can be made fast and intuitive,
- a set of methodologies for analyzing the data analysis process,
- studies examining the behavior of data analysts in various domains, using a variety of tools.

In addition, we plan to evaluate specific recommendation approaches, including algorithms to rank various actions and visualizations that effectively encapsulate the transformations being recommended. This effort will require substantial quantitative and qualitative analysis of queries as well as the data sets to which the queries referred. To this end, we plan to characterize a variety of data sets and evaluate the recommendation system prototype's effectiveness in suggesting useful analysis actions for these.

6. RECOMMENDING VISUALIZATIONS

The analysis presented here are important steps toward creating more intelligent data analysis tools. An effective tool to assist in data exploration via intelligent suggestions requires mechanisms for recommending visualizations, in addition to an intuitive interface to allow the user to provide feedback on suggestions, indicate changes to make, explore data in more detail, or take control over the creation of visualizations. This tool should support modern versions of the techniques articulated by Tukey in his original text on the topic [15], such as histograms, scatter plots, and run charts, which have now become standard practice, in addition to newer types of visualizations. As a sketch of how such a tool might work: the user uploads their data, and initially receives a set of visualizations to browse, ranked in some fashion. Ideally, the tool would be able to identify and prioritize visualizations which conveyed some unexpected information, possibly by using recent techniques for anomaly detection with explanations [16]. The user could choose to inspect certain visualizations further, or provide feedback to the tool about what he or she would like to see. For example, the user might indicate a desire to see more visualizations for a certain subset of the data, or certain type of visualizations for different subsets of the data. The tool should also have some means for inferring which visualizations were most valued by the user to use that information when making future suggestions.

Ranking would be preferable over suggesting all type-compatible visualizations. For example, even if we consider only suggesting two-dimensional scatter plots for a simple data set consisting of n numerical columns, there are $n^2 - n$ possible visualizations. With each additional data type, visualization type or data transformation that we consider, of which there are at least dozens, if not hundreds, the space of possible suggestions increases by an additional $O(n^k)$ term, where k is the dimensionality of the possible visualizations. Thus, for realistic data sets, all possible type-compatible visualizations would number in the tens or hundreds of thousands and would overwhelm the user.

Instead, suggestions should be selected to achieve both breadth and relevance. Breadth of suggestions is important for covering as many different types of questions as possible and giving the user wide set of options for moving forward with the exploration, which will depend in part on their goals. For example, the tool should not just suggest scatter plots at first, or only suggest time series charts. One way to achieve breadth using LSI is to recommend operations that provide good coverage of the space; i.e. that are spread far apart.

Relevance is important in two senses, first in that the suggestions must be compatible with the type of the data. For example, categorical values that happen to be integers should not be treated as numerical values and averaged. Second, the suggestions should make semantic sense for the data. For example, with server performance data, while it is type-compatible to visualize 10th percentile response time over time, users with such data are much more likely to derive value from viewing the 95th or 99th percentile response time over time. One way to achieve relevance using LSI is to recommend operations that are near by fingerprints from semantically similar data sets.

7. CONCLUSION

In this paper, we motivated the need for intuitive, intelligent data exploration tools that reduce the overhead of manual visualization decisions, and instead enable the user to spend more time learning from the data. One possible reason that this vision has not been fully realized is that it is hard to extract the right features from exploratory data analysis examples. To progress towards this vision, it is important to understand the process of data analysis on a deep level. We contribute towards this understanding with a study of data analysis queries from Splunk. We were able to make progress because Splunk programs are concise instances of people analyzing data. The results of this analysis provide some indication that our goal is practical. In particular, we demonstrate promising evidence that what a data set is, semantically, influences what analysis operations are performed, in Section 4. These results suggest that we can recommend similar types of visualizations for similar types of data sets. In other words, we can suggest analysis operations based on measuring the semantic similarity between a given data set and data sets for which "good" analyses are known. In Section 6, we outline ideas for what it would mean to suggest "good" visual analyses and what a tool that did this might look like.

8. REFERENCES

[1] Tableau software. `www.tableausoftware.com`.

[2] Abraham Bernstein, Foster Provost, and Shawndra Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):503–518, 2005.

[3] Ledion Bitincka, Archana Ganapathi, Stephen Sorkin, and Steve Zhang. Optimizing data analysis with a semi-structured time series database. In *SLAML*, 2010.

[4] Stephen M Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics (TOG)*, 10(2):111–151, 1991.

[5] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2011.

[6] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM Human Factors in Computing Systems (CHI)*, 2011.

[7] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Advanced Visual Interfaces (AVI)*, 2012.

[8] Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *ACM International Conference on Management of Data (SIGMOD)*, 2012.

[9] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–141, 1986.

[10] Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[11] Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 112–117. ACM, 1994.

[12] Michael Schiff et al. *Designing graphic presentations from first principles*. University of California, Berkeley, 1998.

[13] Robert St. Amant and Paul R Cohen. Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 7(4):545–558, 1998.

[14] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):52–65, 2002.

[15] John Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

[16] Kiri Wagstaff, Nina Lanza, David Thompson, Thomas Dietterich, and Martha Gilmore. Guiding scientific discovery with explanations using DEMUD. In *Conference on Artificial Intelligence (AAAI)*, 2013.

Methods for Exploring and Mining Tables on Wikipedia

Chandra Sekhar Bhagavatula, Thanapon Noraset, Doug Downey
Department of Electrical Engineering & Computer Science, Northwestern University
{csbhagav, nor.thanapon}@u.northwestern.edu, ddowney@eecs.northwestern.edu

ABSTRACT

Knowledge bases extracted automatically from the Web present new opportunities for data mining and exploration. Given a large, heterogeneous set of extracted relations, new tools are needed for searching the knowledge and uncovering relationships of interest. We present *WikiTables*, a Web application that enables users to interactively explore tabular knowledge extracted from Wikipedia.

In experiments, we show that *WikiTables* substantially outperforms baselines on the novel task of automatically joining together disparate tables to uncover "interesting" relationships between table columns. We find that a "Semantic Relatedness" measure that leverages the Wikipedia link structure accounts for a majority of this improvement. Further, on the task of keyword search for tables, we show that *WikiTables* performs comparably to Google Fusion Tables despite using an order of magnitude fewer tables. Our work also includes the release of a number of public resources, including over 15 million tuples of extracted tabular data, manually annotated evaluation sets, and public APIs.

1 Introduction

Researchers have made significant strides toward automatically extracting massive, open-domain knowledge bases from Web content [1–10]. While a variety of extraction methods have been developed, the important tasks of searching, browsing, and mining the new knowledge bases have remained relatively unexplored.

This paper investigates new methods for searching and mining a knowledge base extracted from Wikipedia data tables. Consider the "electricity consumption" table found in Wikipedia (which lists total Megawatt-hours consumed for each country, and other data).[1] A user viewing this table may be curious how *other* properties of countries (e.g. CO2 emissions, GDP, etc.) are related to electricity consumption. This information need motivates the first task we consider, *relevant join*: the task of automatically identifying, given a query table $\mathbf{T}_q$, which columns from other tables would make relevant additions to $\mathbf{T}_q$. Viewed as a database join, in the *relevant join* task we first identify a column c in $\mathbf{T}_q$ to use for joining—e.g., country names in the electricity consumption table. Then, we find a distinct table $\mathbf{T}_t$ (e.g. a list of countries by GDP) with a column c' similar to c, and perform a left outer join on $c = c'$ to augment $\mathbf{T}_q$ with an automatically selected target column from $\mathbf{T}_t$ (e.g., the GDP amounts). Unlike a standard database join, the *relevant join* task requires automatically selecting the join columns and the target column to make a *relevant* addition to the query table. Automatically joining distinct tables can provide users with unified data views that aren't available in any single table on the Web.

Further, the open-domain nature of tables in Wikipedia presents an opportunity for automatically uncovering interesting or even surprising statistical relationships. The second task we investigate is the novel *correlation mining* task: determining which correlations between numeric columns are likely to be interesting or surprising, rather than trivial or spurious. For example, a user may find it interesting that electricity consumption is strongly correlated with population, and be surprised to learn it is even more strongly correlated with GDP.

In this paper, we introduce *WikiTables*, a prototype information exploration system that allows users to search, join, and mine Wikipedia tables.[2] *WikiTables* focuses on Wikipedia tables, rather than tables across the Web. Our results indicate that Wikipedia's smaller size (1.4M data tables, compared to an estimated 154 million across the Web [1]) is often outweighed by its high quality and more easily extracted semantics. In contrast to Web tables, Wikipedia's tables are explicitly distinguished from page layout, and rarely duplicated across pages.

WikiTables leverages Wikipedia's rich content and link structure as features within machine learners to search and mine the extracted tables. As one example, the semantic relatedness (SR) between Wikipedia pages estimated from the link graph [11] forms an extremely valuable feature for identifying relevant joins. On the *relevant join* task, we show that *WikiTables* more than doubles the accuracy of baselines, and that Semantic Relatedness (SR) measures account for 58% of the increase. Preliminary results on the *correlation mining* task show that our system improves F1-score over the baseline by about 26%. Finally, we also present experimental results on the previously studied *table search*, the task of

[1] `http://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption`

[2] `http://downey-n1.cs.northwestern.edu/public/`

IDEA'13, August 11th, 2013, Chicago, IL, USA.

returning relevant tables in response to a given textual query, and show that *WikiTables* performs comparably to previously published results by Venetis et al. [9], *despite* using an order of magnitude fewer tables.

Our work includes the release of a publicly available prototype for each of the three tasks, along with several data resources, including over 15 million tuples of extracted tabular data, the first manually annotated test sets for the *relevant join*, *table search*, and *correlation mining* tasks, and public APIs for accessing tabular data and computing Semantic Relatedness.[3]

2 Previous Work

A wide variety of recent research is aimed at automatically extracting large bodies of structured data from the Web. Examples include systems that extract relational tuples from Web text [3–7] or from semi-structured Web sources like Wikipedia infoboxes [12, 13] or lists [14, 15]. This paper focuses on the more recent approach of extracting data from *tables* on the Web to power new Web search capabilities [1, 8–10, 16, 17].

Compared to other Web information extraction targets, table extraction offers certain advantages. Each table typically encapsulates a complete, non-redundant set of facts of a given type (e.g., an "electricity consumption" table on Wikipedia lists total Megawatt-hours consumed for each country, per-capita statistics, and the year the information was measured[4]). In extraction from free text, by contrast, systems extract facts individually, and then must deduplicate and synthesize the facts to form a complete set—a very challenging research problem (see e.g. [18]).

One closely related work to ours is found in the Google Fusion Tables system [1, 9, 10]. Our *WikiTables* system was inspired by Fusion Tables and shares similar goals of providing interactive, searchable access to tabular information. However, *WikiTables* is distinct in its focus on Wikipedia tables, and on the mining of tabular data. We introduce the novel *relevant join* task, which shares commonalities with the *schema complement* task studied in [10] but is distinct in that *relevant join* is aimed at automatically identifying specific columns of relevant data, rather than related tables as in [10]. We also present first investigation into the *correlation mining* task on extracted tables. Also, unlike the Fusion Tables work, our table corpus, APIs, and evaluation data with human relevance judgments are all publicly available.

Yakout et. al. introduced the task of Attribute Discovery in [19], which involves finding columns of attributes from tables for a given query set of entities. Our *relevant join* task differs from Attribute Discovery in an important way: in Attribute Discovery, the input is a set of entities, whereas in *relevant join* the input is a table. Thus, unlike in Yakout et. al., the columns returned for *relevant join* by *WikiTables* must be relevant not only to the set of entities, but also to the context in which the set of entities is found. For example, we would expect the most relevant columns to add to a country GDP table would differ from the most relevant columns for an Olympic Medal table, even though the entities are the same in both cases (countries).

Finally, our experiments reveal that a Semantic Relatedness (SR) measure that estimates the relatedness between Wikipedia topics using the link graph [11, 20] is especially valuable for the *relevant join* task. Experimenting with other SR measures that utilize other inputs, beyond Wikipedia links (e.g., [21, 22]), is an item of future work.

3 Task Definitions

In this section, we formally define our three tasks: *relevant join*, *correlation mining* and *table search*. All three tasks utilize a corpus of tables extracted from Wikipedia, so we begin by formally defining the table extraction process.

3.1 Table Extraction from Wikipedia

WikiTables scans all Wikipedia articles for tables. Each table found in an article is extracted and converted into an $m \times n$ matrix of cells (details in Section 4.1). We refer to these matrices as *normalized tables*. Each cell in the matrix holds data from one cell in the table, consisting of all string values and links to Wikipedia concepts contained in the table cell.

The set of all normalized tables extracted from Wikipedia forms our corpus, $\mathcal{T}$. Each normalized table $\mathbf{T}_i \in \mathcal{T}$, having m rows and n columns is represented as a list of column vectors:

$$\mathbf{T}_i = (\mathbf{c}_1^i \mathbf{c}_2^i \dots \mathbf{c}_n^i)$$
$$\text{where } \forall k \in [1, n] \;\; \mathbf{c}_k^i \text{ is a vector of length } m$$

3.2 Relevant Join

Relevant join is the task of finding *columns* that can be added to a given table and ranking them in descending order of relevance to the given table.

Consider a user viewing table $\mathbf{T}_q$, the *QueryTable*, as shown in Figure 1. An informative column to add to this table could be the population data of each country. This data exists in a different table, denoted as $\mathbf{T}_t$ (the *TargetTable*). The "Country" column of $\mathbf{T}_q$ and the "Nation" column of $\mathbf{T}_t$ contain nearly identical data, which suggests the two tables can be joined on these columns. The columns from $\mathbf{T}_q$ and $\mathbf{T}_t$ that contain similar data are called the *SourceColumn* ($\mathbf{c}_s^q$) and *MatchedColumn* ($\mathbf{c}_m^t$) respectively. All columns from $\mathbf{T}_t$ except $\mathbf{c}_m^t$ can be added to $\mathbf{T}_q$. We refer to any column that can be added to a *QueryTable* from a *TargetTable* as a *CandidateColumn*, denoted by $\mathbf{c}_c^t$. The final table, $\mathbf{T}_f$, is created by appending $\mathbf{T}_q$ with the "Population" column from $\mathbf{T}_t$. A *CandidateColumn* that is added to a *QueryTable* is referred to as an *AddedColumn*, $\mathbf{c}_a^t$.

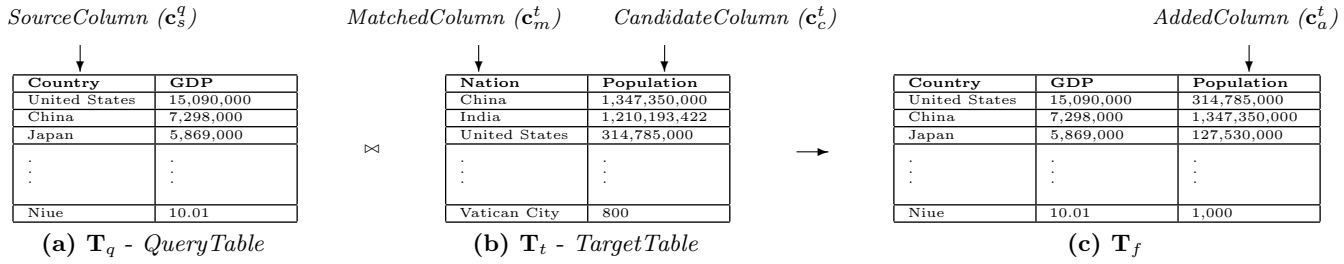
Formally, FindRelevantJoin($\mathbf{T}_q, \mathcal{T}$) takes a query table $\mathbf{T}_q$ and the corpus of tables $\mathcal{T}$ as inputs, and returns a ranked list of triplets denoting relevant joins. Each triplet consists of the *SourceColumn* ($\mathbf{c}_s^q$), the *MatchedColumn* ($\mathbf{c}_m^t$) and the *CandidateColumn* ($\mathbf{c}_c^t$). Triplets $(\mathbf{c}_s^q, \mathbf{c}_m^t, \mathbf{c}_c^t)$ are ranked in decreasing order of relevance of $\mathbf{c}_c^t$ to $\mathbf{T}_q$. In each triplet, $m \neq c$, columns $\mathbf{c}_m^t, \mathbf{c}_c^t \in \mathbf{T}_t$ and $\mathbf{c}_s^q \in \mathbf{T}_q$. Columns $\mathbf{c}_s^q$ and $\mathbf{c}_m^t$ are chosen such that they contain nearly identical data in their cells. The pre-processed set of (*SourceColumn*, *MatchedColumn*) pairs across the corpus $\mathcal{T}$ is referred by $\mathcal{M}$.

Example 1: Figure 2 shows a table from the *"List of countries by GDP (nominal)"* Wikipedia page.[5] This table is the query table, $\mathbf{T}_q$. The column *"GDP (PPP) $Billion"* is the *AddedColumn* $\mathbf{c}_a^t \in \mathbf{T}_t$, where $\mathbf{T}_t$ is the *"List of countries*

[3] http://downey-n1.cs.northwestern.edu/public/

[4] http://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption

[5] http://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)#List



(a) $\mathbf{T}_q$ - *QueryTable* **(b)** $\mathbf{T}_t$ - *TargetTable* **(c)** $\mathbf{T}_f$

Figure 1: Joining a *QueryTable* $\mathbf{T}_q$, (that contains GDP information) with a *TargetTable* $\mathbf{T}_t$, (that contains Population data) to get $\mathbf{T}_f$. Country column from $\mathbf{T}_q$ is the *SourceColumn* and the Nation column from $\mathbf{T}_t$ is the *MatchedColumn*.

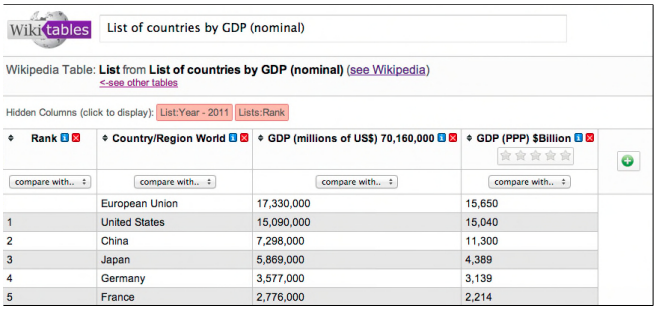


Figure 2: *WikiTables* column addition in action: *WikiTables* shows the table containing list of countries and their GDP (nominal) stats. The first 3 columns belong to this table. *WikiTables* adds the right-most column, "GDP (PPP) Billion", to this table.

by GDP (PPP)" table.[6] The data in the "Country/Region" column in $\mathbf{T}_q$ matches the data from the "Country" column in $\mathbf{T}_t$. These two columns from $\mathbf{T}_q$ and $\mathbf{T}_t$ are the match columns $\mathbf{c}_s^q$ and $\mathbf{c}_m^t$ respectively.

The *relevant join* task involves two primary challenges. First, we wish to select an *AddedColumn* that lists attribute values for the entities in the *SourceColumn*. In practice this can be challenging. Consider a hypothetical table containing columns, "City," "Country," and "Mayor." Here, "Mayor" is a property of "City," and not of "Country." Thus, if we erroneously select "Country" as a *MatchedColumn* and "Mayor" as an *AddedColumn*, this will lead to a poor join. Secondly, we wish to choose *AddedColumns* that are relevant to the query table. For example, the "prisoners" column is relevant to the incarceration table in Figure 2, whereas other attributes of countries (Olympic gold medal winners, for example) are far less relevant. The *relevant join* task requires identifying this distinction automatically, for a given table. Both of the above challenges are addressed by our trained models for the *relevant join*, which are described in the following section.

3.3 Correlation Mining

Correlation Mining is the novel task of determining which correlations between numeric table columns are"interesting."

Formally, FINDCORRELATIONS($\mathcal{M}$) takes the set $\mathcal{M}$, consisting of pairs of matching columns, and generates a set $\mathcal{P}$, of triplets $(\mathbf{c}_s, \mathbf{c}_m, type)$. $\mathbf{c}_s$ and $\mathbf{c}_m$ are numeric columns. Column $\mathbf{c}_s$ belongs to the same table as *SourceColumn* and $\mathbf{c}_m$ belongs to the same table as *MatchedColumn*, where (*SourceColumn*, *MatchedColumn*) is a pair in $\mathcal{M}$.

The third element, *type*, is one of the following three categories: (i) *Unrelated:* Signifying that there could not be a reliable relation between the numeric columns, irrespective of the correlation coefficient, or that the correlation is insignificant, e.g. the correlation of GDP with the years in which a country hosted the World Taekwondo Championships (ii) *Plausible:* If a correlation is intuitive – for example, GDP correlates highly with the imports of a country. (iii) *Surprising:* If a user finds the correlation surprising – for example, the GDP of a country correlates slightly positively with its linguistic diversity. We note that interestingness is inherently subjective, and rating a correlation as *Surprising* is largely dependent on the annotator's background. More extensive user studies that measure how the categorization of attributes may differ across different user groups is an item of future work.

3.4 Table Search

Table search is the task of returning a ranked list of tables for a given textual query. Formally, TABLESEARCH($q, \mathcal{T}$) takes as input a query, q, and the corpus of tables, $\mathcal{T}$ and returns a list of tables in decreasing order of relevance to q. For example, the top three results returned by *WikiTables* for **TableSearch**(`country population`, $\mathcal{T}$) [7] are *1*) List of countries by population *2*) List of sovereign states and dependent territories by population density *3*) World population.

4 WikiTables - System Description

In this section, we describe our *WikiTables* system, and its *relevant join*, *correlation mining* and *table search* capabilities. *WikiTables* performs these tasks using a corpus of tables extracted in advance from Wikipedia. Parsing Wikipedia tables accurately requires some care. We first describe our extraction method and evaluate its accuracy. We then describe how *WikiTables* utilizes the extracted tables for the *relevant join*, *correlation mining* and *table search* tasks.

4.1 Extracting Wikipedia Tables

Wikipedia offers free download of all articles. The data-source for *WikiTables* is the English Wikipedia XML dump available online.[8] This XML dump contains data in Wiki Markup which encases tables between the '{|' and '|}' tags.[9] However, some Wikipedia pages may also contain tables that are *not*

[6] `http://en.wikipedia.org/wiki/List_of_countries_by_GDP_(PPP)#Lists`

[7] `http://downey-n1.cs.northwestern.edu/tableSearch/`

[8] `http://dumps.wikimedia.org/enwiki/`

[9] `http://en.wikipedia.org/wiki/Help:Wiki_markup`

found within these table tags, but get rendered to web users at runtime by expanding *templates*.[10] We found nearly 643,000 Wikipedia pages that contained about 1.4 million tables that were either enclosed within the tags or rendered by expanding templates.

We restrict extraction to tables that belong to the HTML class `wikitable` since it is the only class used to denote data tables on Wikipedia.[11] Filtering on this class attribute allows us to ignore other parts of a page that are rendered as tables – such as infoboxes, table of contents or meta-data boxes – which we do not wish to include in our table corpus.

The resulting set of matrices form our corpus of extracted normalized tables, $\mathcal{T}$.

To evaluate the quality of our table extraction, we picked 50 random pages (using Wikipedia's random page generator[12]) that contained at least 1 table. The total number of tables found on these 50 pages was 111. *WikiTables* extracted 82 out of these 111 tables ($recall = 0.74$). The total number of cells in the 82 tables is 6404, of which *WikiTables* extracted 6383 cells accurately ($precision = 99.7\%$).[13] This is much more precise than table extractors that crawl the full Web, which are faced with lower-quality content and the challenging problem of interpreting when `<table>` tags indicate relational data rather than page layout. In fact, the previous WebTables system achieved a precision of only 41% in extracting relational data tables [23], whereas all tables extracted by *WikiTables* contain relational data.

Hence, we find that *WikiTables* extracts tables at very high precision at the cost of some recall. Most of the loss in recall is due to data tables missing the class attribute `wikitable`. Identifying and fixing such errors is part of future work. Our experiments in Section 5 show that this level of extraction precision and recall is sufficient to build a system that is comparable to other implementations of *table search*.

4.2 Relevant Join Implementation

In this section we describe how *WikiTables* performs the *relevant join* task. *WikiTables*'s implementation of the *relevant join* task requires identifying *CandidateColumns* and computing Semantic Relatedness (SR) for the candidates in real-time. We begin by describing pre-processing which enables efficient execution at query time.

4.2.1 Pre-Processing

Our table corpus $\mathcal{T}$ contains more than 7.5 million columns in all the tables in $\mathcal{T}$ which makes finding joins at runtime prohibitively inefficient. In order to find relevant joins in realtime, we pre-process tables in $\mathcal{T}$ to identify which tables can be joined with a given table. This pre-processing allows *WikiTables* to quickly identify *CandidateColumns* for a given query table efficiently, and then use machine learning models (described in the following section) to rank these *CandidateColumns*.

We pre-compute pairs of matched columns, $(\mathbf{c}_s^i, \mathbf{c}_m^j)$ for columns $\mathbf{c}_s^i \in \mathbf{T}_i$ and $\mathbf{c}_m^j \in \mathbf{T}_j$. Here $\mathbf{c}_s^i$ is the *SourceColumn* and $\mathbf{c}_m^j$ is the *MatchedColumn*. The percentage of values in $\mathbf{c}_s^i$ found in $\mathbf{c}_m^j$ is called the *MatchPercentage*. It must be noted that *MatchPercentage* is not a commutative property of a pair of columns, $(\mathbf{c}_s^i, \mathbf{c}_m^j)$.

To reduce the number of columns that are considered as candidates for joins, we use heuristics to select only those columns that (i) are non-numeric (ii) have more than 4 rows of data (iii) have an average string length greater than 4. This prevents joining tables on columns that contain serial numbers, ranks, etc. In our experience, using these heuristics improves the quality of joins dramatically.

After filtering out columns with the heuristics, we are left with about 1.75 million columns. We compute these matches exhaustively, to obtain the *MatchPercentage* for each pair of columns. Matches that have a *MatchPercentage* greater than 50% are added to our corpus of matches $\mathcal{M}$.

We have nearly 340 million matches in $\mathcal{M}$, the set of matching pairs. Section 4.2.3 explains how $\mathcal{T}$ and $\mathcal{M}$ are used at runtime to find candidate columns to add to a table and rank them by relevance.

4.2.2 Semantic Relatedness in *WikiTables*

A *Semantic Relatedness* (SR) measure takes as input two concepts, and returns a numeric relatedness score for the concept pair. Our experiments demonstrate that preferring joins between page concepts with higher SR results in significant performance improvements on the *relevant join* task.

In *WikiTables*, we use the $MilneWitten$ Semantic Relatedness measure [11], which estimates the relatedness of Wikipedia concepts (each Wikipedia page is treated as a "concept"). $MilneWitten$ is based on the intuition that more similar Wikipedia pages should share a larger fraction of inlinks. We use the $MilneWitten$ implementation from Hecht et al. [20], an enhancement to the original measure that emphasizes prominent links within the Wikipedia page "gloss" and learns parameters of the measure based on hand-labeled relatedness judgments.

Utilizing SR at runtime within *WikiTables* requires quickly computing the relatedness between the page containing the query table and all other pages containing candidate columns. Naively computing in-link intersections with every candidate column at query time would be intractable. To solve this performance issue, we pre-compute all non-zero relatedness scores for all concepts on Wikipedia, and store these in memory. The data store is compressed (by quantizing SR scores to 256 bins and using simple variable-byte encoding), and occupies about 30GB of memory. We have provided access to this data store through a public API.[14]

4.2.3 Finding Relevant Joins

In *WikiTables*, the user begins by selecting a table to view. This table is the input to the FindRelevantJoins algorithm illustrated in Algorithm 1. FindRelevantJoins takes a query table $\mathbf{T}_q$ and the table corpus $\mathcal{T}$ as inputs, and returns a set of triplets. A triplet contains a *SourceColumn*, a *MatchedColumn* and a *CandidateColumn*, denoted $\left(\mathbf{c}_s^q, \mathbf{c}_m^t, \mathbf{c}_c^t\right)$. These triplets are ranked on the basis of the estimated relevance of adding $\mathbf{c}_c^t$ to $\mathbf{T}_q$. From the top ranked triplet, $\mathbf{c}_c^t$ can be added to $\mathbf{T}_q$ through a left outer join between $\mathbf{T}_q$ and $\mathbf{T}_t$ ON $\mathbf{c}_s^q = \mathbf{c}_m^t$.

These triplets are formed by first getting all matches from $\mathcal{M}$, in decreasing order of *MatchPercentage* such that one of the columns from $\mathbf{T}_q$ is the *SourceColumn* (the function GetMatchPairs in Algorithm 1). All columns from the

[10] `http://en.wikipedia.org/wiki/Help:Template`
[11] `http://en.wikipedia.org/wiki/Help:Wikitable`
[12] `http://en.wikipedia.org/wiki/Special:Random`
[13] `http://downey-n1.cs.northwestern.edu/public/`
[14] `http://downey-n2.cs.northwestern.edu:8080/wikisr/sr/sID/69058/langID/1`

Algorithm 1 Pseudo-code to find relevant joins for a given table

```
function FindRelevantJoins(T_q, 𝒯)
    C_Matches ← ∅                                   ▷ Set of (SourceColumn , MatchedColumn) pairs
    for all c ∈ GetColumns(T_q) do                  ▷ GetColumns(T_q) returns all columns in T_q
        C_Matches ← C_Matches ∪ GetMatchPairs(c)    ▷ Get Pairs from ℳ in descending order of match percentage, for a source column
    end for
    Sort_MatchPercent(C_Mc)                         ▷ Sort (SourceColumn , MatchedColumn) pairs in decreasing order of MatchPercent
    C_candidates ← {}                               ▷ Set of (SourceColumn , MatchedColumn, CandidateColumn) triplets
    for all (c_s^i, c_m^j) ∈ C_Matches do
        for all x ∈ GetColumns(TableOf(c_m^j)) ∧ x ≠ c_m do   ▷ TableOf(c_m^t) returns table, T_t to which c_m^t belongs
            C_candidates ← C_candidates ∪ (c_s^i, c_m^j, x)
        end for
    end for
    C_F ← ClassifyJoins(C_candidates)               ▷ Classify triplets as relevant or non-relevant using a trained classifier
    C_R ← RankJoins(C_F)                            ▷ Rank triplets in C_F using a ranking model
    return C_R
end function
```

matched column's table except the matching column are considered as candidate columns.

Candidate columns are first classified as either *relevant* or *non-relevant* using a trained classifier model, and then ranked using a ranking model. The classifier discards low quality column additions. We used the Weka implementation [24] of the Logistic Regression model as our classifier, and a linear feature based Coordinate Ascent ranking model, implemented by RankLib.[15] Both models use features listed in Table 1. The models were trained on small set of hand labeled examples as described in Section 5.1.1.

Entities in the *SourceColumn* may be present more than once in the *MatchedColumn*. In this case, all values from the *AddedColumn* that correspond to a single entity in the *MatchedColumn* are grouped together and added to the query table, i.e. we display all values that we join to within a single cell. A feature ("avgNumKeyMap" in Table 1) measures how many values each cell in the added column contains, on average. Our results (Section 5.1) show that one-to-many joins are not preferred by users—there is a strong negative correlation between the avgNumKeyMap feature and the relevance rating of a column.

4.3 Correlation Mining Implementation

WikiTables uses the set of column matches $\mathcal{M}$ to find correlations between numeric columns. For each pair $(\mathbf{c}_s^i, \mathbf{c}_m^j) \in \mathcal{M}$, *WikiTables* finds all pairs of numeric columns $(\mathbf{n}^i, \mathbf{n}^j)$, such that $\mathbf{n}^i \in$ TableOf$(\mathbf{c}_s^i)$ and $\mathbf{n}^j \in$ TableOf$(\mathbf{c}_s^j)$, and calculates Pearson correlation coefficient between the numeric columns. Table 2 lists the features used to build machine learning models to classify each pair of correlated columns into one of three types described in Section 3.3.

4.4 Table Search Implementation

The dataset used for *table search* is the set of normalized tables, $\mathcal{T}$, extracted from Wikipedia. Our goal is to find the most relevant tables from $\mathcal{T}$ for a given textual query q, using contents of the table and the text around the table. To incorporate textual features of the page containing the table, we use an existing search service.[16] At runtime, a user's query is first sent to the search service, and result URLs restricted to the *en.wikipedia.org* domain are retrieved. Using $\mathcal{T}$, we obtain tables found in each of the top 30 Wikipedia pages returned for a query. A trained linear ranking model (learned with Coordinate Ascent [25]) ranks these tables using four types of features described in Table 3. We train our ranker on a small set of hand-labeled examples, as described in Section 5.3.1.

[15] `http://people.cs.umass.edu/~vdang/ranklib.html`

[16] `http://developer.yahoo.com/boss/search/`

Name	***Description***
matchPercentage	*MatchPercentage* between *SourceColumn* and *MatchedColumn*
avgNumKeyMap	Average number of times a key in the *SourceColumn* is found in the *MatchedColumn*
srRelate	SR measure between the page containing the *SourceColumn* and the page containing the *MatchedColumn*
srcAvgLen	Average length of data in the *SourceColumn*
srcDistinctValFrac	Fraction of distinct values in the *SourceColumn*
srcNumericValFrac	Fraction of columns in *SourceTable* that are numeric
tgtAvgLen	Average length of data in the *CandidateColumn*
tgtIsNum	Boolean, 1 if *CandidateColumn* is numeric
tgtDistinctValFrac	Fraction of distinct values in the *CandidateColumn*
inLink	Number of inlinks to the page to which *TargetTable* belongs
outLink	Number of outlinks from the page to which *TargetTable* belongs
srcTableColIdx	Boolean feature: 1 if *SourceColumn* is among the first three columns of the table
matchedTableColIdx	Boolean feature: 1 if *TargetColumn* is among the first three columns of the table

Table 1: Features used by *WikiTables* for the *relevant join* task

Name	***Description***
corel	Correlation Coefficient between numeric columns
sr	Semantic Relatedness measure between the pages containing the matched pair of columns
se	Standard error of the correlations coefficient
numSamples	Number of samples considered in the calculation of correlation coefficient
zScore	zScore of correlations for a given source column

Table 2: Features used by *WikiTables* for the *correlation mining* task

5 Experiments

In this section, we present our evaluation of *WikiTables* on the three tasks: *relevant join*, *correlation mining* and *table search*.

Name	*Description*
Page Features	
yRank	Rank in Yahoo!'s results for query
inLinks	Number of in-links to page
outLinks	Number of out-links from
pageViews	Number of page views
Table Features	
numRows	Number of rows
numCols	Number of columns
emptyCellRatio	Fraction of empty cells
Table+Page Features	
sectionNumber	Wikipedia Section index the table occurs in
tableImportance	Inverse of number of tables on page
tablePageFraction	Ratio of table size to page size
Query Features	
qInPgTitle	Ratio: Number of query tokens found in page title **to** total number of tokens
qInTableTitle	Ratio: Number of query tokens found in table title **to** total number of tokens

Table 3: Features used by *WikiTables* for the *table search* task

We describe the datasets, comparison methods, and metrics used for this evaluation, along with results obtained. We also present an analysis of the results obtained for each task. Experiments show that *WikiTables* beats baseline methods on the *relevant join* and *correlation mining*. *WikiTables* also performs comparably to previous work on the *table search* task, despite using an order of magnitude fewer tables. We also demonstrate that Semantic Relatedness measures defined over Wikipedia concepts are especially valuable for the *relevant join* task.

5.1 Relevant Join

5.1.1 Datasets

We evaluated *WikiTables* on a dataset that consists of tables to which columns can be added. To provide a non-trivial number of columns to rank, we performed a weighted random selection of 20 Wikipedia articles that contained at least one table, where weight of a page is the number of times a column from that page occurs in our match corpus $\mathcal{M}$ as a *SourceColumn*. The random list of articles used for evaluation are listed here.[17]

The WikiTables UI allows users to rate automatically joined columns on a scale of 1 to 5. To train the machine learning models, for *relevant join*, described in Section 4.2, 4 users posted 593 ratings on 82 distinct tables, mutually exclusive from the test set. For the classifier, all ratings of 3 and above were considered *relevant* (positive class of the classifier). Input for the ranker is the actual rating given by the user. For evaluation, two annotators rated the top 4 columns added to each table from pages in the test set, on a scale of 1 to 5, indicating relevance of an added column to the query table. Inter annotator agreement, measured by Spearman's rank correlation coefficient between ratings of the two annotators for each query, was found to be 0.73.

5.1.2 Comparison Methods

To our knowledge, *WikiTables* is the first system that finds and ranks relevant columns that can be added to a given table. Previous work has looked into using *Schema Complement* as an indicator of relatedness of tables and using this relatedness to improve table search [10]. Google Fusion Tables allows adding columns to a table from other tables, but a user selects the *subject* column manually and also selects the column(s) to be added. Thus, to evaluate *WikiTables* on the *relevant join* task, we created a baseline method (Base) and compared it with our implementation, *WikiTables*. Base simply ranks *CandidateColumns* in decreasing order of *MatchPercentage*. In an ablation study to evaluate the impact of *Semantic Relatedness* on this task, we created a system (*WikiTables−SR*) which ignores the *srRelate* feature from both the classifier and ranking models.

5.1.3 Metrics

To evaluate *WikiTables* on this task, we used the standard information retrieval metrics of Discounted Cumulative Gain (DCG) [26] and Normalized Discounted Cumulative Gain ($nDCG$). Both of these metrics are designed to give greater importance to occurrences of more useful results higher in a ranked list. DCG at a particular rank position p is given by:

$$\mathbf{DCG_p} = \sum_{i=1}^{p} (2^{rel_i} - 1)/(\log_2(i+1)) \quad (1)$$

where rel_i is the rating of the i^{th} ranked result table. $nDCG$ is defined as the ratio between DCG of the ranked list retrieved and DCG of the ideal ranking possible. This is given by:

$$\mathbf{nDCG_p} = (DCG_p)/(IDCG_p) \quad (2)$$

$nDCG$ metric is calculated for the top 4 columns added by a method to each query table. We choose to evaluate on the top 4 as this is a typical number of added columns that fits in a single browser window. The normalizing factor in this case is the ideal ranking of the union of columns added by the three methods that are being compared. Performance of each method can be estimated by taking the average of $nDCG$ values across all queries.

We also measure the accuracy of columns that are added to a table by a method. We define this metric as:

$$\mathbf{Acc_m} = \frac{\#\text{ columns with rating} \geq 3}{\#\text{ columns added by method m}} \quad (3)$$

5.1.4 Results and Analysis

The results of our evaluation are summarized in Table 4, which lists the average accuracy of the four columns added by each method, and the average DCG and $nDCG'$ values of results retrieved for all query tables. Both Base and *WikiTables−SR* added columns to 106 tables and *WikiTables* added columns to 103 tables. Results show that compared to Base, *WikiTables* more than doubles the fraction of relevant columns retrieved. Further, Semantic Relatedness measures defined over Wikipedia concepts account for approximately 58% of this improvement (over *WikiTables−SR*).

Model	**cols added**	**acc.**	**DCG@4**	**nDCG′@4**
Base	423	0.29	11.38	0.43
WikiTables−SR	414	0.43	13.71	0.48
WikiTables	410	**0.62**	**30.71**	**0.76**

Table 4: Results of comparison of three methods for the *relevant join* task. *WikiTables* more than doubles the accuracy of added columns as compared to the baseline Base. Using Semantic Relatedness feature results in a significantly improved performance in *WikiTables* as compared to *WikiTables−SR*

Table 5 lists the correlation between features and user rating for the *relevant join* task. Results show that columns

[17] `http://downey-n1.cs.northwestern.edu/public/randomQSet.html`

containing numeric data make more relevant additions to a table than other non-numeric ones. A greater value of Semantic Relatedness, more distinct values in the source column, and a higher match percentage between matched columns leads to higher-quality of added columns.

Feature	Correlation with rating
tgtIsNum	0.285
srRelate	0.253
avgNumKeyMap	-0.231
srcDistinctValFrac	0.209
srcTableColIdx	-0.202
matchPerc	0.167
outLink	-0.161
inLink	-0.157
srcNumericValFrac	0.134

Table 5: Spearman's rank correlation coefficient of *relevant join* features with column rating. Only features with significant correlations ($p < 0.01$) are listed. Correlation is calculated over 485 pairs.

The results described above show that the *Semantic Relatedness* measure between Wikipedia pages is a very valuable indicator of relatedness of which column should be added to a given table. For example, for a table of the top 10 largest power producing facilities, *WikiTables−SR* adds an unrelated column containing the number of Olympic medals won by countries in swimming. On the other hand, *WikiTables*, which uses the Semantic Relatedness features, adds highly relevant information about the annual CO_2 emissions of countries.

5.2 Correlation Mining

5.2.1 Datasets

We evaluated *WikiTables* on the *correlation mining* task using 100 randomly sampled pairs of numeric columns from the set of correlations $\mathcal{P}$. These 100 pairs were manually classified into one of three categories: *Unrelated*, *Plausible*, or *Surprising* correlations.

5.2.2 Comparison Methods

We tried different models and compared their performance on our dataset. The models used are: ZeroR for baseline, Logistic Regression, SVM Classifier, 1-Nearest Neighbor and 3-Nearest Neighbor classifiers. Each classifier is used to predict one of the three classes: *Unrelated*, *Plausible* or *Surprising*.

5.2.3 Metrics

We performed 10-fold cross validation over these 100 pairs of labeled columns. We choose the standard F1 score as our metric for evaluation.

5.2.4 Results and Analysis

Table 6 shows the accuracy and F1 score of each model. We find that 1-vs-All Logistic Regression model has the best performance.

Using our best model, i.e. Logistic Regression, we performed an ablation study by removing each of the five features. Results are tabulated in Table 7.

5.3 Table Search

We compare *WikiTables* with previously published work by Venetis et al. [9] (referred to further as TABLE). We define metrics that are used to measure relative performance of table search methods and present results that show that

Model	Accuracy	F1
ZeroR (Baseline)	66%	0.525
SVM Classifier	65%	0.52
1-Nearest Neighbor	64%	0.62
3-Nearest Neighbor	71%	0.657
Logistic Regression	**71%**	**0.661**

Table 6: Preliminary Results on the Correlation Mining task. All classifiers classify data into 4 classes (using 1-vs-All classification where applicable)

Feature removed	Accuracy	F1
corel	70%	0.649
sr	68%	0.626
se	70%	0.639
numSamples	71%	0.661
zScore	72%	0.66

Table 7: Ablation Study

WikiTables outperforms TABLE in the *table search* task. The results of the *table search* experiments also show that even with a much smaller dataset of tables, *WikiTables* satisfies more user queries aimed at retrieving tables from the Web.

5.3.1 Datasets

As we compare *WikiTables* with Web-based extractors, it is important that we do not bias the evaluation in favor of information from Wikipedia. To prevent this bias, all of our *table search* evaluation query workloads are drawn from previous work on searching tables from the Web at large.

Our first test set `D1` consists of a set of 100 queries of the form (C, P), where C is a string that denotes a *class* of instances, and P is also a string that denotes a *property* associated with these instances. This dataset is taken from previous work by Venetis et al. [9]. For each query in the test set of 100 (C, P) pairs, Venetis et al. rated the top 5 tables retrieved by a table search method as either *right on* (if it had all information about a large number of instances of the class and values for the property), *relevant* (if it had information about only some of the instances, or of properties that were closely related to the queried property) or *irrelevant*. Some systems of table search (e.g. GOOG , GOOGR - described in the following section), used by [9] to compare with TABLE, return documents and not just tables. Thus, annotators also annotated if a result was found *in a table*. On the other hand, *WikiTables* only returns tables as results of a search query.

We replicated this evaluation method on *WikiTables*, hand-labeling results for each of the 100 queries. Because *WikiTables* takes a single string as its input rather than a (C, P) pair, we simply concatenate the class and property strings to form the textual query for our system. More sophisticated handling of classes and properties within *WikiTables* is an item of future work.

We measured inter-annotator agreement on a held out set of 8 queries (from Sarma et al. [10]). Two raters annotated each table returned for queries in this set on a scale of 0 to 5. The average Spearman's correlation coefficient between ratings of the annotators across all queries was 0.71, which we believe is sufficiently high for such a subjective task.

For training the *WikiTables*'s table search ranker (Section 4.4), we utilized a training set of about 350 labeled examples on 5 queries, disjoint from the test set queries.

Method	All Ratings				Ratings by Queries				Metrics		
	Total	(a)	(b)	(c)	Some Result	(a)	(b)	(c)	$\mathbf{P_{tq}}$	$\mathbf{R_{tq}}$	$\mathbf{F1_{tq}}$
WikiTables	500	54	88	142	100	**35**	**63**	**63**	0.63	**0.63**	**0.63**
TABLE	175	69	98	93	49	24	41	40	**0.82**	0.4	0.54
DOCUMENT	399	24	58	47	93	13	36	32	0.34	0.32	0.33
GOOG	493	63	116	52	100	32	52	35	0.35	0.35	0.35
GOOGR	156	43	67	59	65	17	32	29	0.45	0.29	0.35

Table 8: Comparing results of TABLE and *WikiTables*. *WikiTables* outperforms TABLE with a much higher recall and $F1$ score. Results of DOCUMENT, GOOG and GOOGR have been re-printed from [9]. The columns under All Ratings present the number of results that were rated to be (a) *right on*, (b) *right on* or *relevant*, and (c) *right on* or *relevant* and *in a table*. The Ratings by Queries columns aggregate ratings by queries: the sub-columns indicate the number of queries for which a table in the top 5 results got a rating with (a) *right on*, (b) *right on* or *relevant*, and (c) *right on* or *relevant* and *in a table*.

5.3.2 Comparison Methods

We compare *WikiTables* with previous work by Venetis et al.. Our primary evaluation compares performance on data set `D1` with previously published results from [9]. The best-performing method from that work, TABLE, automatically annotates all tables in the corpus with labels, which indicate what a table is *about*, and ranks these labels on importance. The table search system within TABLE takes queries of the form (C, P), where C is a class name and P is a property, and returns a ranked list of tables. To create this ranked list, TABLE considers tables in the corpus that have a class label C in the top-k class labels in its annotations. TABLE then ranks these tables based on a weighted sum of a number of signals, some of which are derived from the property P from the query. The weights were determined using a training set of examples. In their published results, Venetis et al. compared performance of TABLE with three other methods: 1. GOOG : the results returned by www.google.com, 2. GOOGR: the intersection of the table corpus (from TABLE) with the top-1,000 results returned by GOOG, and 3. DOCUMENT: document-based approach proposed in [1]. We include results of the performance of all methods compared in [9], for completeness.

5.3.3 Metrics

The primary metric we use to compare *WikiTables* with previously published results on data set `D1` is simply the number of queries for which a given method returned a result in its top 5 that was *relevant* or *right on* and *in a table*. Venetis et al. also evaluated *precision* on the table search task [9]. We consider precision to be a secondary metric for this task (it is rare in information retrieval tasks to distinguish between returning poor results and returning no results), but following [9] we also evaluate on precision and F1 metrics:[18]

$$\mathbf{P_{tq}} = \frac{\#\text{ queries with } \textit{right on} \text{ or } \textit{relevant} \text{ tables}}{\#\text{ queries for which results returned}} \quad (4)$$

$$\mathbf{R_{tq}} = \frac{\#\text{ queries with } \textit{right on} \text{ or } \textit{relevant} \text{ tables}}{\#\text{ queries}} \quad (5)$$

$$\mathbf{F1_{tq}} = (2 * \mathbf{P_{tq}} * \mathbf{R_{tq}})/(\mathbf{P_{tq}} + \mathbf{R_{tq}}) \quad (6)$$

5.3.4 Results and Analysis

Table 8 shows the results of our experiment comparing *WikiTables* with TABLE using the Dataset `D1`. While TABLE only produced a result for 49 of the 100 queries, *WikiTables* retrieved tables for all 100 queries. Furthermore, TABLE returned a *right on* or *relevant* table in the top 5 results for only 40 queries, whereas *WikiTables* returned a relevant table for 63 queries—a 58% increase.

[18]Our metrics are similar to but differ slightly from those of [9]; the precise metrics they use depend on which methods retrieved accurate tables for which queries, which is not known to us.

Feature	Correlation with rating
tablePageFraction	0.41
numRows	0.40
tableCaption ContainsQuery	0.31
sectionNumber	-0.19
inlinks	-0.16
numCols	0.156

Table 9: Spearman's rank correlation coefficient between *table search* features and user ratings on tables retrieved for text queries. Only features with Significant correlations ($p < 0.01$) are listed.

Our results show that on query workloads from previous work on Web table search, the higher quality of Wikipedia data combined with our machine learning approach results in *WikiTables* outperforming Web table search systems. The text surrounding a table is more relevant for table search when tables are drawn from high-quality, topically-focused pages like those on Wikipedia.

Table 9 lists the Spearman's rank correlation coefficients between features and user ratings of tables. The features in bold represent statistically significant correlations. From the table, it can be observed that tables that cover larger fraction of a page tend to be better and that bigger tables are better. One surprising observation is that the number of inlinks has a significant negative correlation with result quality. This is because prominent Wikipedia pages (with more inlinks) do not necessarily contain better tables. For example, the "List of counties in Maryland" page, which has only 928 inlinks, contains much higher-quality tabular data than the "Maryland" page that has about 29,000 inlinks. The "List of prolific inventors" page, which has 67 inlinks, contains much better tabular data than the "Alexander Graham Bell" page that has about 2600 inlinks.

In our evaluation of *WikiTables* on Dataset `D1`, there were 37 queries for which *WikiTables* did not retrieve any *right on* or *relevant* tables in the top 5 results. We analyzed the types of queries for which *WikiTables* failed to retrieve relevant tables. The query classes for which *WikiTables* did not retrieve accurate tables are the ones that are open-ended, i.e. the number of entities that belong to these classes is very large. Examples include *movie stars*, *guitars*, *football clubs*, *beers*, *clothes*, etc. Since these classes contain large numbers of entities, they are generally not found in just one single table. They rather tend to be sub-categorized into smaller

groups. Such *open-ended* queries accounted for 17 out of the 37 queries in which *WikiTables* did not retrieve any *right on* or *relevant* tables.

6 Conclusions and Future Work

In this paper, we introduced *WikiTables*, a system that extracts, searches, and automatically joins Wikipedia tables. We showed that leveraging high-quality textual and tabular content on Wikipedia is valuable for the *relevant join*, *correlation mining*, and *table search* tasks. On the novel *relevant join* and *correlation mining* tasks, our machine learning methods are found to significantly outperform baselines. Results on *table search* show that the smaller number of tables on Wikipedia is outweighed by their higher quality and more easily extractable semantics.

In future work, we plan to investigate ways to combine *WikiTables* with the broader-coverage Google Fusion Tables to improve the precision and recall of each. We also wish to investigate richer interactive data mining over the table corpus, that goes beyond the two-column correlations studied in this paper. Wikipedia tables and Web tables include an unprecedentedly broad collection of entity properties, presenting a novel opportunity to generate insights from data and make predictions. For example, one could ask whether the myriad of properties of countries found across Wikipedia tables might aid in predicting how a country's unemployment rate will change over time.

7 Acknowledgments

This work was supported in part by NSF Grant IIS-1016754 and DARPA contract D11AP00268.

8 References

[1] M.J. Cafarella, A. Halevy, D.Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 2008.

[2] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007.

[3] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th WWW*, 2004.

[4] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts-step one: the one-million fact extraction challenge. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[5] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. *Open information extraction for the web*. PhD thesis, University of Washington, 2009.

[6] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Annual meeting-association for Computational Linguistics*, 2007.

[7] Y. Fang and K.C.C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *Proceedings of the fourth WSDM*, 2011.

[8] S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Eng. Bull*, 2010.

[9] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 2011.

[10] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding related tables. In *Proceedings of the 2012 SIGMOD*. ACM, 2012.

[11] I.H. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, 2008.

[12] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2009.

[13] F. Wu and D.S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM CIKM*, 2007.

[14] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2009.

[15] R.C. Wang and W.W. Cohen. Iterative set expansion of named entities using the web. In *Data Mining, 2008. ICDM'08. Eighth ICDM*, 2008.

[16] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[17] B. Chan, L. Wu, J. Talbot, M. Cammarano, and P. Hanrahan. Vispedia: Interactive visual exploration of wikipedia data via search-based integration. *IEEE TVCG*, 2008.

[18] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of NAACL HLT*, 2007.

[19] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 international conference on Management of Data*, pages 97–108. ACM, 2012.

[20] B. Hecht, S.H. Carton, M. Quaderi, J. Schöning, M. Raubal, D. Gergle, and D. Downey. Explanatory semantic relatedness and explicit spatialization for exploratory search. In *Proceedings of the 35th international ACM SIGIR*, 2012.

[21] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th IJCAI*, 2007.

[22] J. Hoffart, S. Seufert, D.B. Nguyen, M. Theobald, and G. Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM CIKM*, 2012.

[23] M.J. Cafarella, A.Y. Halevy, Y. Zhang, D.Z. Wang, and E. Wu. Uncovering the relational web. *WebDB*, 2008.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 2009.

[25] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 2007.

[26] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM (TOIS)*, 2002.

One Click Mining— Interactive Local Pattern Discovery through Implicit Preference and Performance Learning*

Mario Boley, Michael Mampaey, Bo Kang, Pavel Tokmakov, and Stefan Wrobel
Fraunhofer IAIS and University of Bonn
Schloss Birlinghoven, Sankt Augustin, Germany
{mario.boley,stefan.wrobel}@iais.fhg.de
{michael.mampaey,bo.kang,pavel.tokmakov}@uni-bonn.de

ABSTRACT

It is known that productive pattern discovery from data has to interactively involve the user as directly as possible. State-of-the-art toolboxes require the specification of sophisticated workflows with an explicit selection of a data mining method, all its required parameters, and a corresponding algorithm. This hinders the desired rapid interaction—especially with users that are experts of the data domain rather than data mining experts. In this paper, we present a fundamentally new approach towards user involvement that relies exclusively on the implicit feedback available from the natural analysis behavior of the user, and at the same time allows the user to work with a multitude of pattern classes and discovery algorithms simultaneously without even knowing the details of each algorithm. To achieve this goal, we are relying on a recently proposed co-active learning model and a special feature representation of patterns to arrive at an adaptively tuned user interestingness model. At the same time, we propose an adaptive time-allocation strategy to distribute computation time among a set of underlying mining algorithms. We describe the technical details of our approach, present the user interface for gathering implicit feedback, and provide preliminary evaluation results.

1. INTRODUCTION

Productive pattern discovery from data is known to be an iterative process that ideally requires a tight interaction between a discovery system and a user who is an expert of the data domain [Fayyad et al., 1996]. State of the-art data analysis suites (e.g., Rapid Miner [Mierswa, 2009], WEKA [Hall et al., 2009], KNIME [Berthold et al., 2008]) rely on an explicit construction of a discovery workflow including selection of a discovery method along with its parameters, a corresponding mining algorithms, and post-processing of results. The resulting high number of alternative formalizations of a single analysis task poses a substantial burden on creating and iteratively refining these workflows—especially for users that lack deep technical understanding of data mining methods—and ultimately it hinders the desired rapid interaction [Cao, 2012]. Previous research tried to alleviate this problem by assisting the user in constructing or selecting a workflow (see Morik and Scholz [2004] or Bernstein et al. [2005]) or by providing direct active user-involvement for individual parts of the workflow such as for the post-processing [Xin et al., 2006] or even the mining itself [Goethals et al., 2011]. However, all these approaches still expose the user to the complexity of the discovery worklow and/or require technical knowledge about its components that goes way beyond the semantic knowledge about the data domain.

In this paper, we present a fundamentally new approach towards user involvement that for the first time requires neither an explicit formalization of analysis goals in terms of a workflow or another specification language nor any technical data mining knowledge that goes beyond the pure data semantics. Instead, the approach relies exclusively on the implicit feedback available from the natural analysis behavior of the user when he investigates the data and mining results. At the same time, the approach allows the user to work with a multitude of pattern classes and mining algorithms simultaneously without even knowing the details of each algorithm. In particular, the approach avoids all method selection and configuration steps from standard processes, and instead an individual mining step is started by pressing once a single dedicated mine button. Hence we refer to this process as *one-click mining*.

Naturally, the goal of this process is to produce patterns that are relevant to the latent user interest as fast as possible. We show how this goal can be achieved by the interplay of two appropriately designed online learning/optimization components. On the one side, there is model of the hidden user interest based an a suitably designed feature representation of all pattern types that are included in the range of the analysis system. The learning of the corresponding model parameters is based on the recently proposed co-active learning model [Shivaswamy and Joachims, 2012, Raman et al., 2012]. On the other side, there is a time-allocation strategy that distributes the computational time-budget available in each discovery round among a set of underlying mining algorithms. We model this task as a multi-armed bandit explo-

*This article is a short version containing only preliminary experimental results. Complete evaluation will be available in a future full version.

IDEA'13, August 11th, 2013, Chicago, IL, USA.

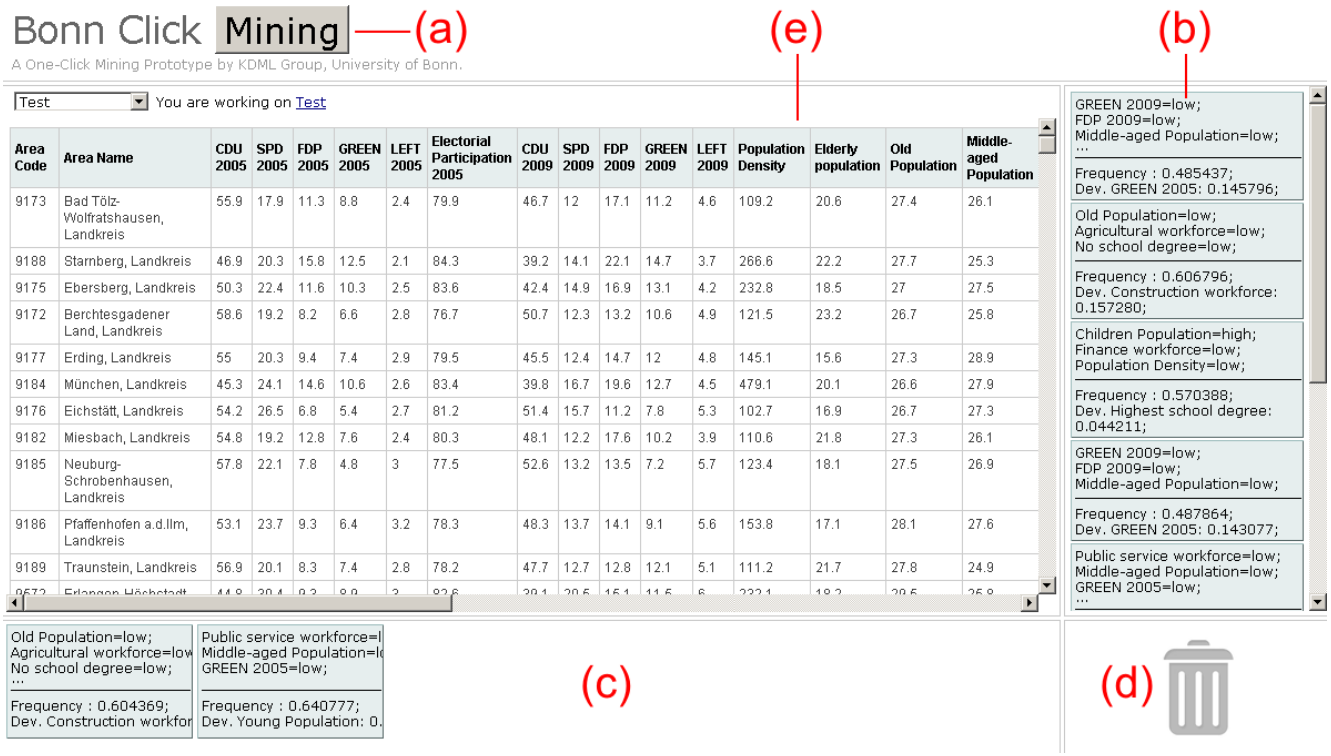


Figure 1: Visual layout of one-click mining prototype that contains (a) mine-button, (b) result candidate area, (c) result analysis board, trash can (d), and data view (e).

ration/exploitation problem where the payoffs correspond to the utility of the mined patterns. Since this utility is measured by an evolving approximation of the user interest, we address this problem by a bandit algorithm suitable for shifting payoffs [Cesa-Bianchi and Lugosi, 2006]. Overall, we end up with a very general method that can aggregate any combination of data mining tasks, for which results can be mined by parameter-free anytime algorithms and be represented in a suitable joint feature space.

After recapping basic pattern discovery concepts from a unifying perspective (in Sec. 2), we present in detail the one click mining framework along with visual components that support its user/system dialog (Sec. 3). We then give a proof of concept based on an exemplary instantiation that combines subgroup discovery and association discovery (Sec. 4). The resulting prototypical one-click mining system, called *Bonn click mining* (see Fig. 1), is demonstrated in the context of a socio-economic data analysis task. We document that the system is able to quickly provide interesting patterns corresponding to latent analysis goals that have been learned implicitly.

2. PATTERN DISCOVERY

In this section we provide and repeat general formal definitions for pattern discovery starting from pattern classes (or languages), over scoring functions that assess pattern interestingness, up to mining algorithms that aim to find interesting patterns. As illustrative examples, we recall subgroup discovery [Klösgen, 1996] and association discovery [Webb, 2011], which are also used in our one-click mining prototype.

As notational convention, throughout this paper we denote by $[n]$ for a positive integer $n \in \mathbb{N}$ the set $\{1, \ldots, n\}$, and by $\mathbb{B}$ the set of truth values $\{\text{true}, \text{false}\}$.

2.1 Pattern Languages

Standard approaches to local pattern discovery that are fully automatized usually rely on a specific pattern (descriptor) language along with a single choice of a measure for assessing the utility or interestingness of a descriptor. In contrast, one-click mining can aggregate a mixture of different pattern discovery methods and interestingness measures. Hence, we have to introduce an explicit notion of pattern that combines a descriptor with the information of why it is supposed to be interesting (i.e., wrt what measure).

We assume a given fixed **dataset** $D = \{d_1, \ldots, d_m\}$ of m **data records** $d \in D$, each of which is described by a set of n **attributes** $A = \{a_1, \ldots, a_n\}$. All attributes a_i assign to each data record a value from their **attribute domain** V_i, i.e., $a_i \colon D \to V_i$. Here, we assume that attributes are either **numerical**, i.e., $V_i \subseteq \mathbb{R}$ and we use $\leq$ to compare attribute values, or **categorical**, i.e., $|V_i|$ is finite and its values are incomparable. A **pattern language** $\mathcal{L}$ is a set of **pattern descriptors** $s \in \mathcal{L}$ to each of which we can associate a local **extension** $D(s) \subseteq D$ in the data.

For example, in **association discovery**, where one aims to find attribute/value combinations that show a high co-occurrence in the data, one usually considers the language $\mathcal{L}_{\text{cnj}}$ of **conjunctions** of constraints on individual attribute values. That is, $\mathcal{L}_{\text{cnj}}$ contains descriptors s of the form

$$s = c_{i_1}(\cdot) \wedge \cdots \wedge c_{i_s}(\cdot)$$

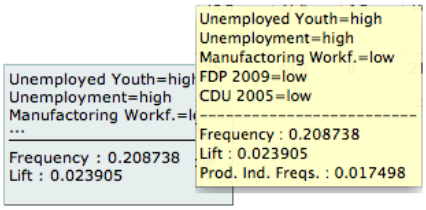


Figure 2: Association pattern with a descriptor containing five attribute constraints and a rationale that contains two elementary interestingness functions and one function derived from them.

such that $c_{i_j} : V_{i_j} \to \mathbb{B}$ is a binary constraint on attribute a_{i_j} for all $j \in [i_s]$. Correspondingly, the extension of s is defined as

$$D(s) = \{d \in D\colon\ c_{i_1}(a_{i_1}(d)) \wedge \cdots \wedge c_{i_k}(a_{i_k}(d))\} \ .$$

In particular, for a categorical attribute a_i we consider **equality constraints** $c(v) \equiv v = b$ with $b \in V_i$, and for numerical attributes we consider **interval constraints** $c(v) \equiv v \in [l, u]$ with $l, u \in V_i$. We refer to $\mathrm{cnst}(s) = \{c_{i_1}, \ldots, c_{i_s}\}$ as the constraints and to $\mathrm{attr}(s) = \{a_{i_1}, \ldots, a_{i_s}\}$ as the attributes **contained** in s, respectively. We assume that each attribute is contained at most once and call $|\mathrm{cnst}(s)| = |\mathrm{attr}(s)| = i_s \in \mathbb{N}$ the **size** of s.

A further example of a pattern language is provided by **subgroup discovery** where one is interested in pattern descriptors that are at the same time fairly general (have a large extension) and that show an unusual distribution of one specific **target attribute** $a_t \in A$ in their extension. The corresponding pattern language is $\mathcal{L}_{\mathrm{sgd}} = \mathcal{L}_{cnj} \times [n]$, i.e., it contains descriptors $s = (c, t)$ with a conjunctive descriptor c annotated by the index t of a target attribute. The conjunction c also defines the extension of the subgroup descriptor s, i.e., $D(s) = D(c)$.

2.2 Interestingness and Patterns

In order to assess how interesting a pattern descriptor is as an observation in the data, there exists a wide range of **interestingness functions** $f : \mathcal{L} \to \mathbb{R}_+$ that have been developed across different pattern discovery methods (see Geng and Hamilton [2006]). A basic example is the **frequency** $f_{\mathrm{frq}}(s)$ of a pattern descriptor s, which is defined as its generality measured as the fraction of data records that are part of the pattern's extension, i.e., $f_{\mathrm{frq}}(s) = |D(s)| \,/\, |D|$. Another general example is the **relative shortness** of a pattern defined by $f_{\mathrm{sho}}(s) = (n - |\mathrm{attr}(s)|)/n$.

Specifically for **subgroup discovery interestingness**, corresponding to its semantics, one typically uses functions of the form

$$f^b_{\mathrm{sgd}}(s, t) = f_{\mathrm{frq}}(s)^b f_{\mathrm{dv}}(s, t) \ , \qquad (1)$$

i.e., a multiplicative combinations of frequency (weighted by a real-valued parameter $b \in [0, 1]$) and a **target deviation function** f_{dv}. In this work, we choose f_{dv} to be the total variation distance between the distribution of the target attribute in the pattern extension $S = s(D)$ and the distribution in the complete data, i.e.,

$$f_{\mathrm{dv}}(s, t) = \sup_{X \subseteq V_t} |\mathbf{p}^t_S(X) - \mathbf{p}^t_D(X)| \ .$$

Here, $\mathbf{p}^t_S$ and $\mathbf{p}^t_D$ are a (fitted) distribution of attribute values of a_t in the pattern extension and the complete data, respectively (see Appendix A for computational details). This function provides a uniform interpretation of interestingness for categorical and numerical target variables.

Association interestingness is usually quantified as the difference between the frequency of the pattern and its expected frequency if we assume that some of its parts are satisfied independently. Here, we use a first order approximation to the leverage measure [Webb, 2010] that can be computed efficiently. That is, we consider the following **additive lift** measure defined by

$$f_{\mathrm{lft}}(s) = \left(f_{\mathrm{frq}}(s) - \prod_{c \in \mathrm{cnst}(s)} f_{\mathrm{frq}}(c) \right) / 2^{|\mathrm{cnst}(s)|-2} \ .$$

Thus, conceptually this measure assumes as null hypothesis that all individual constraints of the descriptor are satisfied independently.

A **pattern** as a pair $(s, F) \in \mathcal{L} \times 2^{\mathcal{F}}$ where $s \in \mathcal{L}$ is a descriptor and $F \subseteq \mathcal{F}$ a rationale consisting of one *or more* interestingness measures with an appropriate domain. As we will see below, it is useful to potentially have more than one function in the interestingness rationale, because standard measures often are a function of several elementary functions—like in subgroup discovery interestingness, which is a function of frequency and target deviation. By giving feedback on a pattern annotated by elementary measures, a user implicitly provides insight into his preferences about all other measures that can be computed from these elementary measures. Fig. 2 shows an example of a pattern as displayed in our one-click mining prototype. By $\mathcal{P} = \mathcal{L} \times 2^{\mathcal{F}}$ we denote the **set of patterns** defined by $\mathcal{L}$ and $\mathcal{F}$.

2.3 Mining Algorithms

For the actual generation of patterns, let us denote by $\mathcal{M}$ a set of k **mining algorithms** that is at our disposal. For notational convenience we sometimes identify $\mathcal{M} = [k]$. From our perspective an individual mining algorithm $m \in \mathcal{M}$ can simply be treated as producing a random set of result patterns $m(t) \subseteq \mathcal{P}$ with an unknown distribution that depends on the time $t \in R_+$ that the algorithm is running. Of course, we usually know more about a given mining algorithm such as the pattern language it uses and the interestingness measure it optimizes. Yet from the perspective of one click mining it is sufficient to treat the algorithms as black boxes as long as they satisfy the following two requirements.

We assume that all algorithms are *parameter-free*. In practice this can mean that a single algorithm either uses a specific parameter assignment of a mining algorithm or it is in fact a meta-algorithm that includes a parameter-selection procedure. Moreover, all algorithms should be *anytime algorithms*, i.e., conceptually at every moment in time after they are started they maintain a current solution that can be retrieved when the algorithm is terminated preemptively. This is necessary, because in one click mining the time budget available for a given run of an algorithm is determined by the user ad hoc. These requirements can (or are automatically) satisfied by a wide range of modern pattern discovery algorithms that provide various pattern types. Examples are Slim [Smets and Vreeken, 2012], pattern sampling [Boley et al., 2012], or beam search approaches [van Leeuwen and Knobbe, 2011].

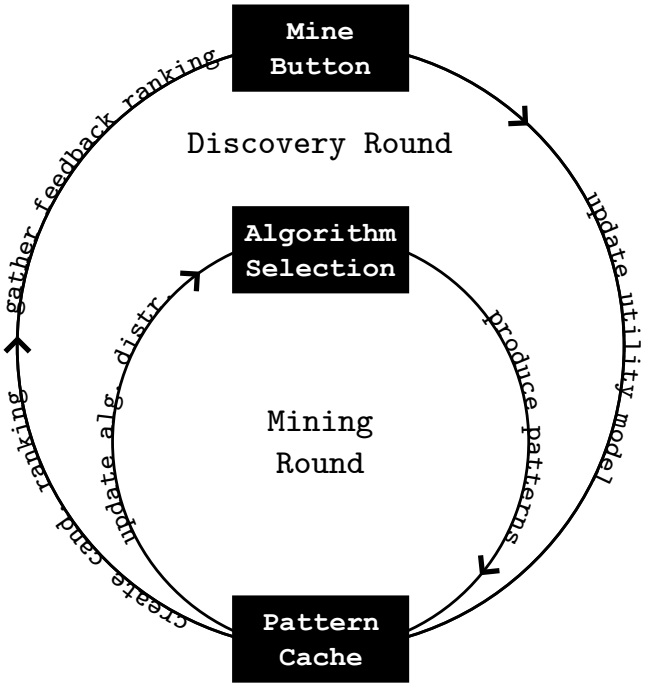


Figure 3: Temporal structure of discovery process: one discovery round can contain several mining rounds.

3. ONE-CLICK MINING

We now show how different pattern discovery methods and algorithms can be combined within a one-click mining system. After outlining the general process dynamics and the visual components that support the user/system interaction, we describe in detail the two online learning components involved in that process: the ranking mechanism of the mined results and the exploitation/exploration strategy for controlling the underlying mining algorithms.

3.1 Discovery Process and Visual Elements

The key idea of one-click mining is to not occupy the user with the technicalities of the mining workflow, but instead to let her focus on investigating the produced results on a semantic level. In this subsection we present visual elements that support this investigation and show how the interaction with these elements can be incorporated into a sensible discovery process that produces useful patterns.

The most characteristic visual element is the **mine button** (Fig. 1 (a)), which the user presses in order to see new mining results. Implicitly, pressing this button also indicates that the user is done with inspecting the results that were displayed before. Moreover, there is a result **candidate area** (Fig. 1 (b)), which is used to present ranked mining results. From here the user can investigate results, delete those that she considers useless by drawing them to the **trash can** (Fig. 1 (c)), and move those which she wishes to use for further analysis to the result **analysis board** (Fig. 1 (d)). These elements support an interactive *discovery process*, which is defined as follows (see Fig. 3).

The main temporal unit of the process are **discovery rounds** $t \in \mathbb{N}$ that correspond to the period between two consecutive activations of the mine button. As further, more refined, temporal structure, during each discovery round there is a number of **mining rounds** $l \in \mathbb{N}$, each of which corresponding to a background execution of a mining algorithm. Here, the last mining round within a given discovery round is terminated preemptively in order to end synchronously with this discovery round. Hence, every mining round can be associated to a unique discovery round. We count mining rounds consecutively and denote by $t(l)$ the discovery round in which mining round l occurs and conversely by $l(t)$ the first mining round within the discovery round t.

An individual mining round l consists of first selecting a mining algorithm m_l at random according to a **distribution for algorithm selection** $\boldsymbol{\pi}_l : \mathcal{M} \to [0, 1]$, running m_l and fetching its result patterns P_l. All mining results are then stored in a **pattern cache**, for which we denote by $C_l \subseteq \mathcal{P}$ the state before the results of mining round l are added. The cache has a finite **cache capacity** $c \in N$ such that at all times l it is enforced that $|C_l| \leq c$. Finally, the performance of m_l is assessed by comparing C_l and C_{l+1} (using a current approximation of the pattern utility defined below). Based on this information the selection distribution for the next round $\boldsymbol{\pi}_{l+1}$ is determined and automatically started.

In the beginning of a discovery round t, a **candidate ranking** r_t of size c is constructed from the current state of the pattern candidate cache $C_{l(t)}$ and displayed in the result candidate area. Formally, a **ranking** of patterns is an ordered list $r = \langle r_1, \ldots, r_k \rangle \in \mathcal{P}^*$ such that $r_i \neq r_j$ for all $i, j \in [k]$. Let us denote by $\{r\} = \{r_1, \ldots, r_k\}$ the (unordered) set of patterns contained in the ranking, by $r^i = \langle r_1, \ldots, r_i \rangle$ the i-**prefix** of r for $i \leq k$, and by $|r| = |\{r\}| = k$ the size of r. The **set of rankings** is denoted by $\mathcal{R}$ and the set of all rankings of some fixed size c is denoted $\mathcal{R}^c$.

The ranking r_t is based on a current **utility approximation** $\hat{u}_t$ of the user-subjective pattern utility u. After the user indicates that she is done with investigating the candidate ranking (by clicking mine as described above), a feedback ranking $\overline{r}$ can be constructed from her actions. For the definition of this feedback ranking, let us denote by T_t and B_t all the patterns that have been deleted to the trash can and promoted to the result board until the end of round t, respectively. The feedback ranking consists of all patterns that were promoted in that round to the result board followed by all patterns in the candidate ranking that have been inspected by the user and were neither deleted nor promoted. Let

$$x = \max\{i \in [c] \mid x_i \notin (B_{t+1} \setminus B_t) \cup (T_{t+1} \setminus T_t)\}$$

be the maximal index of a pattern that has been either promoted or deleted during round t. Formally, the **feedback ranking** is defined by

$$\overline{r}_t = \langle b_1, \ldots, b_k, r_{i_1}, \ldots, r_{i_l} \rangle$$

where $\{b_1, \ldots, b_k\} = B_{t+1} \setminus B_t$ in the order of their promotion and $\{r_{i_1}, \ldots, r_{i_l}\} = \{r^x_t\} \setminus (B_{t+1} \cup T_{t+1})$ with $i_j < i_{j'}$ for $j < j' \leq x$. At the end of the discovery round, a new utility approximation $\hat{u}_{t+1}$ is inferred by comparing $\overline{r}_t$ with r and the next discovery round starts.

3.2 Learning and Construction of Rankings

We now turn to the precise details of the ranking mechanism that is used in the beginning of a discovery round

in order to compute a candidate ranking from the content of the pattern cache. With this mechanism we aim for two goals: firstly, we want to allow the user to find new patterns that are maximally *relevant* in terms of her specific utility preferences, and, secondly, we want to provide her a sufficient amount of *diversity* in the displayed mining results. The latter goal is important in a user-based search process, because if the user is not able to express previously unspecified aspects of her preferences the whole discovery process can get stuck in a local maximum.

In order to achieve these goals, we adapt the co-active learning process proposed in Shivaswamy and Joachims [2012] and Raman et al. [2012] for maintaining the parameter vector $\mathbf{w}_t$ of a **ranking utility function**

$$\hat{u}_t(r) = \langle \mathbf{w}_t, \varphi(x_t, r) \rangle$$

over the discovery rounds $t \in \mathbb{N}$. This function is defined by a joint **feature map** $\varphi \colon X \times \mathcal{R} \to \mathbb{R}^{\mathcal{F}}$ that maps a pattern ranking together with the current system state to an $|\mathcal{F}|$-dimensional real-valued feature vector, i.e., the feature representation is determined by the set $\mathcal{F}$ of interestingness functions. The **system state** $x_t \in X$ in discovery round t is given by the contents of the pattern cache, the result board, and the trash can, respectively, i.e., $x_t = (C_{l(t)}, B_t, T_t)$. The component of φ corresponding to function $f \in \mathcal{F}$ is defined as a discounted aggregate of the individual patterns' contribution to feature f, i.e.,

$$\varphi_f(x_t, r) = \left\| \left(\frac{\delta(x_t, r_i)\varphi_f(r_i)}{\log(i+1)} \right)_{i=1}^{|r|} \right\|_d$$

where $\varphi_f(r_i)$ is given by the feature map for individual patterns (defined below) and $\delta(x_t, r_i)$ is an indicator function that is 1 if pattern r_i neither already present on the result board B_t or in the trash can T_t and 0 otherwise. The choice $d \in \mathbb{N} \cup \{\infty\}$ of the norm is a **diversity parameter** that determines the trade-off between relevance and diversity when evaluating a ranking.

The feature map for the individual patterns is designed to allow a maximal cross-pattern inference and at the same time to only require minimal attention of the user: while the actual pattern p can only contain the values for some base interestingness functions, the feature vector of p also contains values for all interestingness functions that the user can infer from these base functions. For example the rationale of the pattern in Fig. 2 contains only the interestingness functions f_{frq}, and f^t_{dv}. However, this is enough to infer also the values for the multiplicative combinations $f^b_{\text{frq}} f^t_{\text{dv}}$, and, hence, we can also use these in the feature reperesntation of the pattern. Formally, the individual components of the **pattern feature map** are defined by

$$\varphi_f(s, X) = \begin{cases} f(s) & \text{, if } f \in \hat{X} \\ 0, & \text{, otherwise} \end{cases}$$

where $\hat{X}$ denotes the set of all feature functions in $\mathcal{F}$ that can be computed based on functions in the rationale X, i.e.,

$$\hat{X} = \{ f \in \mathcal{F} \colon f_1, \ldots, f_k \in X, \\ f(s) = g(f_1(s), \ldots, f_k(s), s) \} \ .$$

This means the feature representation of a pattern consists of all function values of interestingness functions in the rationale X and those that can be inferred from these values. Other features—that are not in X or that cannot be inferred from X—are set to zero. Note that feature functions that only depend on the pattern descriptor (such as the relative shortness f_{sho}) are always part of $\hat{X}$. Hence, if $\mathcal{F}$ contains features such as descriptor length and indicator features for the presence of the specific constraints, then these features are relevant for all patterns.

Algorithm 1 Greedy ranking

Require: Patterns $P \subseteq \mathcal{P}$, size $c \in \mathbb{N}$, utility fct $u \colon \mathcal{R} \to \mathbb{R}$
Ensure: Ranking $r^c_{\text{grd}}(P)$ s.t. $u(r^c_{\text{grd}}(P))/u(r^c_{\text{opt}}(P)) \geq 1/3$
1. **for** $i = 1, \ldots, c$ **do**
2. **set** $r_i \in \arg\max_{p \in P \setminus \{r_1, \ldots, r_{i-1}\}} u(\langle r_1, \ldots, r_{i-1}, p \rangle)$
3. **return** $\langle r_1, \ldots, r_c \rangle$

With the definition of the ranking utility we can now specify the candidate ranking r_t that is displayed to the user at the beginning of every discovery round t. Naturally, one would want this to be the **optimal ranking** of length c (cache capacity) with respect to the current model, i.e.,

$$r^c_{\text{opt}}(C_{l(t)}) \in \arg\max \{ \hat{u}_t(x_t, r) \colon r \in \mathcal{R}^c(C_{l(t)}) \} \ .$$

Unfortunately, using a reduction from the max-k-cover problem (see, e.g., [Feige et al., 2011]), one can show that it is **NP**-hard to compute this optimal ranking and even to approximate one within a ratio larger than $(1 - 1/e)$. This holds already for very simple feature spaces $\mathcal{F}$, in particular for the one used in our prototype (see Sec. 4). On the other hand, a **greedy ranking** $r^c_{\text{grd}}(P)$ can be constructed efficiently by Alg. 1, which iteratively grows a solution by adding in each step to the current partial ranking the pattern that maximizes the utility. For all pattern sets $P \subseteq \mathcal{P}$ this solution can be computed in time $O(c\,|P|)$ and satisfies the approximation guarantee

$$\hat{u}_t(r^c_{\text{grd}}(P))/\hat{u}_t(r^c_{\text{opt}}(P)) \geq 1/3 \ .$$

This result can be proven by observing that the space of partial rankings can be represented by the intersection of two Matroids and that $\hat{u}$ is *sub-modular* with respect to that set system. The approximation guarantee then follows from a general performance theorem for the greedy algorithm from Fisher et al. [1978].

Finally, we can specify how to update the parameter vector of the ranking utility at the end of each discovery round. Following Raman et al. [2012], we can update by the following multiplicative **utility update rule**

$$\mathbf{w}_{t+1,f} = \mathbf{w}_{t,f} \exp(\theta_t(\varphi_f(\overline{r}_t) - \varphi_f(r_t)))/Z \qquad (2)$$

where Z is a normalization factor that ensures that $\|\mathbf{w}_t\|_2 = 1$ and $\theta_t = 1/(2S\sqrt{2^{\lfloor \log t \rfloor}})$ is a decreasing **utility learning rate** depending also on a bound $S \geq \max_{r,B} \|\varphi(r)\|_\infty$ on the max-norm of all rankings (in our prototype we can, e.g., use $S = c^{1/d}$; see Sec. 4). The approximation guarantee of the greedy algorithm and a certain guarantee on the quality of the user feedback imply that this update mechanism has a controlled regret over an optimal weight vector.

3.3 Online Control of Mining Algorithms

It remains to specify the algorithm selection distribution $\boldsymbol{\pi}_l$ that is used in mining round l. As mentioned earlier, we consider the output of mining algorithms as random variables following a distribution that depends on the available

running time. In order to asses the mining performance, the system can only observe the output of algorithms that it actually uses and initially the performance of all algorithms are unknown. Thus, the system is facing an exploitation/exploration problem of the multi-armed bandit style (see, e.g., Cesa-Bianchi and Lugosi [2006]). In order to apply known strategies for this kind of problem, we first have to model the reward that is generated by a mining algorithm when it is executed.

Let us P_l denote the **result pattern set** returned by the mining algorithm executed in round l (denoted by $m_l \in \mathcal{M}$) and by c_l the **computation time** it used to produce these results. Then the mining performance of round l can be quantified by the utility gain per time of the ranking that can be constructed from the old and the new patterns together, i.e., by

$$(u(r_{\text{opt}}(P_l \cup \{r_{t(l)}\})) - u(r_{t(l)}))/c_l \ .$$

Of course, the system has no access to the true utility function and cannot compute an optimal ranking efficiently. Hence, it has to rely on its current approximation $\hat{u}_{t(l)}$ and the greedily approximated ranking to estimate the performance, i.e., it has to use the estimated relative **utility gain**

$$g_l = (\hat{u}_{t(l)}(r_{\text{grd}}(P_l \cup \{r_{t(l)}\})) - \hat{u}_{t(l)}(r_{t(l)}))/c_l \ .$$

Thus, the observed reward generated by a mining algorithm depends not only the current system state but also on the current approximation of the user utility, both of which evolve over time.

This means that we need an exploitation/exploration strategy that is robust to *non-stationary rewards*. To this end, we employ an algorithm of Cesa-Bianchi and Lugosi [2006, p. 160] that has an optimally bounded regret. Throughout all mining rounds $l \in \mathbb{N}$, it maintains **performance potential weights** $\mathbf{v}_l \in \mathbb{R}_+^k$ starting with $\mathbf{v}_1 = (1, \ldots, 1)$. The algorithm m_l to run in mining round l is then chosen at random according to the **algorithm selection distribution** $\boldsymbol{\pi}_l \in [0,1]^k$, which is a mixture of the distribution given by $\mathbf{v}$ and the uniform distribution, i.e., it is given by

$$\boldsymbol{\pi}_{l,i} = ((\gamma_l - 1)\mathbf{v}_i)/V + \gamma_l/k$$

where V normalizes the sum of the entries of $\mathbf{v}$ to one. The **bandit mixture coefficient** γ_l depends on the mining round and will be specified below. After the result of a mining round is observed the potentials are updated multiplicatively by the **bandit update rule**

$$\mathbf{v}_{l+1,i} = \mathbf{v}_{l,i} \exp(\eta_l \overline{g}_{l,i})$$

where η_l is the **bandit learning rate** $\eta_l = \gamma_l/(2k)$ and $\overline{g}_{l,i}$ an optimistic estimator of the performance of algorithm i in round l that is defined by

$$\overline{g}_{l,i} = \begin{cases} (g_l + \beta_l)/\boldsymbol{\pi}_{m_l}), & \text{if } i = m_l \\ \beta_l/\boldsymbol{\pi}_{m_l}), & \text{otherwise} \end{cases} \ .$$

By choosing

$$\beta_l = \sqrt{\ln(10k)/(k2^{\lfloor \log l \rfloor})}$$

one can make sure that the bias of the performance estimates is not too large while still being optimistic with high probability. Depending on β_l one can also chose the bandit mixture coefficient as $\gamma_l = 4k\beta_l/(3 + \beta_l)$.

This concludes the formal description of the one-click mining framework. All algorithmic ideas are summarized again in Alg. 2. Note that this is a compressed listing that needs a slight addition in order to avoid concurrency issues: When the mine-click procedure terminates the currently running mining algorithm, this triggers a call of the algorithm-end procedure. In this case the algorithm-end procedure should only be carried out until step 6 and the new mining algorithm is only started after the remaining steps 5-7 of the mine-click procedure are finished.

Algorithm 2 One-click Mining

Initialization:
1. **init** utility weights $\mathbf{w}_1 \leftarrow (1, \ldots, 1)/|\mathcal{F}|$
2. **init** performance weights $\mathbf{v}_1 \leftarrow (1, \ldots, 1)$
3. **init** discovery and mining round $t, l \leftarrow 1$
4. **draw** algorithm $m \in \mathcal{M}$ uniformly at random
5. **run** m blocking for time c_{init} (result patterns P)
6. **init** candidate buffer $C_1 = P$ and present $r_{\text{grd}}(C_1)$

On Algorithm End:
1. **update** candidate buffer $C_{l+1} = C_l \cup P_l$
2. **asses** $g_l = (\hat{u}_{t(l)}(r^c_{\text{grd}}(C_{l+1})) - \hat{u}_{t(l)}(r^c_{\text{grd}}(C_l))/c_l$
3. **for all** $i \in \mathcal{M}$ **do**
4. $\quad \overline{g}_{l,i} \leftarrow \begin{cases} (g_l + \beta_l)/\boldsymbol{\pi}_{m_l}, & \text{if } i = m_l \\ \beta_l/\boldsymbol{\pi}_{m_l}, & \text{otherwise} \end{cases}$
5. $\quad \mathbf{v}_i \leftarrow \mathbf{v}_i \exp(\eta_l \overline{g}_{l,i})$
6. $l \leftarrow l + 1$
7. **run** algorithm $m_l \sim \boldsymbol{\pi}_l$ in background where

$$\boldsymbol{\pi}_{l,i} = (1 - \gamma_l)\mathbf{v}_i/V + \gamma_l/k$$

On Mine Click:
1. **assess** feedback ranking $\overline{r}_t$
2. **for all** $f \in \mathcal{F}$ **do**
3. $\quad \mathbf{w}_{t+1,f} = \mathbf{w}_{t,f} \exp(\theta_t(\varphi_f(\overline{r}_t) - \varphi_f(r_t)))$
4. **terminate** current algorithm m_l
5. **construct** and show greedy ranking $r_{t+1} = r_{\text{grd}}(C_{l-1})$
6. **reset** $C_l = \{r_{t+1}\}$
7. $t \leftarrow t + 1$

4. PROOF OF CONCEPT

In order to provide a proof of concept for the one-click-mining approach, we present an exemplary pattern discovery session performed by the prototypical one-click mining implementation called *Bonn click mining*. In this session, we deal with the pattern discovery use case of election analysis (see Grosskreutz et al. [2010]).

4.1 Prototype Configuration

The prototype is configured to combine association and subgroup discovery; both with a range of interestingness functions that is able to express different trade-offs between pattern frequency and lift or target deviation, respectively. In addition there are functions that express certain preferences on the form of the pattern only, i.e., its descriptor. On the algorithm side, there is a mixture of 8 deterministic beam-search and randomized pattern sampling algorithms.

More precisely, the pattern language $\mathcal{L}_{bcm}$ of the prototype is the combination of association and subgroup pat-

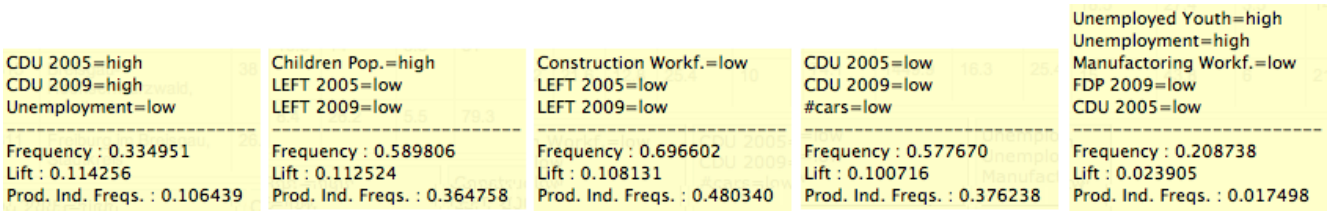


Figure 4: Five association patterns found in analysis phase 1 of proof of concept experiment; patterns were found between discovery rounds 5 and 15 after promoting some simpler (more trivial) associations and deleting some subgroup patterns.

terns, i.e., $\mathcal{L}_{bcm} = \mathcal{L}_{asd} \cup \mathcal{L}_{sgd}$. The feature functions $\mathcal{F}_{\text{bcm}}$ can be separated into three groups, i.e.,

$$\mathcal{F}_{\text{bcm}} = \mathcal{F}_{\text{sgd}} \cup \mathcal{F}_{\text{asd}} \cup \mathcal{F}_{\text{dsc}}$$

where $\mathcal{F}_{\text{sgd}}$, $\mathcal{F}_{\text{asd}}$, and $\mathcal{F}_{\text{dsc}}$ are sets of subgroup discovery, association discovery, and descriptor functions, respectively. The subgroup discovery features $\mathcal{F}_{\text{sgd}}$ contain the functions given by Eq. (1) for the three choices of b equal to 0, 1/2, and 1. Analogously, the association functions contain the same trade-offs with frequency, but with the target deviation measure replaced by the lift measure f_{ass}. Also pattern frequency is included for both pattern classes. Finally, the descriptor features contain the relative shortness f_{sho} along with binary indicator functions f^d_{cns} that signal whether attribute d is present in the descriptor or not, i.e., $f^d_{\text{cns}}(s) = 1$ if $d \in \text{attr}(s)$ and $f^d_{\text{cns}}(s) = 0$ otherwise. For subgroup patterns there are in addition similar features that can express affinity for a specific target attribute, i.e., for all $t \in [n]$ the feature $f^t_{\text{trg}}(s, t')$ that takes on the value 1 if and only if $t = t'$. For the resulting feature space $\mathcal{F}_{\text{bcm}}$ we have for all patterns $p \in \mathcal{P}$ that $\varphi_f(p) \in [0, 1]$. Hence, we can use the bound $S = c^{1/d}$ for setting the learning rate for the utility updates as defined in Eq. (2) where c denotes as usual the capacity of the pattern cache.

The employed set of algorithm $\mathcal{M}_{\text{bcm}}$ consists of 4 direct pattern sampling and 4 beam search algorithm such that from each group there are two algorithms for each discovery task. Direct pattern sampling produces random pattern collections as the outcome of fast appropriately biased random experiments without constructing auxiliary parts of the pattern space (see Boley et al. [2012] from which we also use the complementary pattern sampling library[1]). All algorithm preprocess the data by discretizing numerical attributes into high and low bins. In the case of subgroup discovery all algorithm draw the target attribute at random according to the distribution given by the current weights for the target preference features. The beam search algorithms then directly optimize either subgroup or association interestingness; finding the top-10 patterns with a beam size of 5 or 10, respectively. For the sampling algorithms appropriately constructed pattern distributions are chosen: For association discovery we use distributions that are biased towards patterns with a high frequency on the one side but that contain individual constraints with a low frequency on the other side. This favors the production of patterns with a high lift. For subgroup discovery we split the dataset into to parts corresponding to high and low values of the target attribute and sample patterns discriminating these parts.

4.2 German Socio-economical Data

For this proof of concept we used data from the domain of socio-economics and politics, which can be used to investigate a diverse set of understandable and interpretable analysis questions. Specifically, we constructed a table from publicly available database provided by the German Federal Office of Statistic[2]. This database provides a wide range of statistical variables mapped to regional units of Germany.

For our table, we let the *data records* correspond to the 413 administrative districts of Germany (Landkreise), which is the second finest spatial resolution provided in the database. Each district is described by 39 *attributes* that can be roughly grouped into socio-economical and political variables. In terms of socio-economic attributes we selected variables from the following categories: age structure and education of the population, economic indicators (e.g., GDP growth, unemployment), and structure of the labor market (workforce in different sectors such as production, public service, etc.). In terms of political attributes, we added the election results of the five major political parties for the federal elections in 2005 and 2009, respectively: CDU (conservative), SPD (center-left), GREEN (center-left), FDP (liberal), and LEFT (left-wing).

4.3 Results

We report some results of an exemplary analysis session to illustrate the system behavior. For this we assume that the user starts at first with a very general purely exploratory analysis intent on order to get an overview before she turns to attack more specific questions. That is, we assume the following analysis question for phase 1.

Phase 1—general question:
What attribute/value combinations show a strong correlation in the data?

While following this intent, the user investigates and promotes mostly fairly general association patterns, while she deletes too specific and in particular subgroup patterns. During the first discovery rounds the produced matching candidate patterns are dominated by those that reflect the well-known fact that political parties have relatively stable regional strongholds. This means there are a lot of patterns of the form "party 2005=high/low, party 2009=high/low". Then, after a few rounds, more space in the candidate area is devoted to association patterns, and consequently a higher

[1] `http://www-kd.iai.uni-bonn.de/index.php?page=software_details&id=23`

[2] `www.regionalstatistik.de`

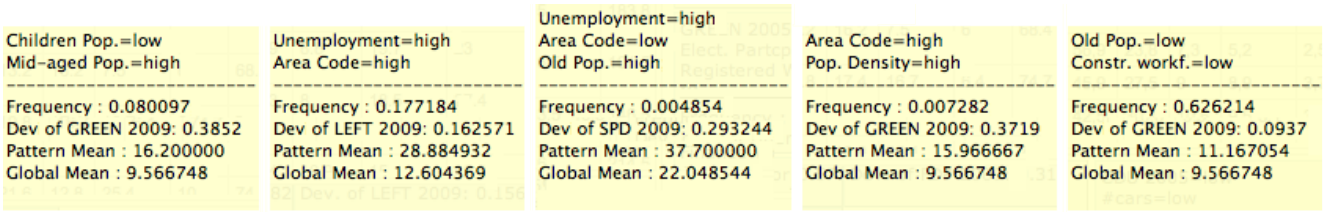


Figure 5: Subgroup patterns found in analysis phase 2 of proof of concept experiment; patterns were found between discovery rounds 10 and 30 after promoting specific party targets and demoting sufficiently many subgroup patterns with other parties in descriptor.

diversity of them with more non-trivial correlation are offered. For instance, the patterns shown in Fig. 4 have been produced between discovery rounds 5 and 15. These patterns confirm some known associations between attributes. CDU is strong in the economically strong regions of the south of Germany that show low unemployment. Conversely, CDU also and its economically liberal coalition partner FDP are weaker in economically problematic areas with high unemployment and little industry. Other patterns in the set appear to be plausible and could be interesting for further investigation.

After this initial exploratory phase, we now turn to a much more specialized analysis question, in which we try to find explanations for the 2009 election results of some specific parties.

Phase 2—specific question:
What socio-economic and regional factors favored parties from the left spectrum in the German 2009 federal election?

This question implies that we are interest in subgroup patterns with the three targets SPD, GREEN, and LEFT. At the same time, we are interested in socio-economic and regional explanations only, which means do not want to see descriptor elements of other parties (or the same party in the 2005 election). Correspondingly, while following this analysis question, the user deletes general association patterns and those subgroup patterns that do comply to the descriptor restrictions mentioned above. Again, just as in phase 1, in the beginning the produced candidates are dominated by the obvious correlations between party attributes. Additionally, now we also have a much more narrow focus of suitable pattern forms, because we are not interested just in any subgroup pattern but only in those with having a target from a set of 3 out of 39 attributes. Consequently, this time it takes longer until suitable patterns show up in the result area. The patterns shown in Fig. 5 have been produced between discovery rounds 10 and 30. Again the patterns partially confirm some known party preferences (note that a high area code corresponds to the south/east and a low area code to north/west regions). For instance it is known that SPD has a relatively high result among older voters, while GREEN is strong in the densely populated urban areas. Finally, LEFT is known to be strong among areas in the east and particularly in economically weak areas with high unemployment.

5. CONCLUSION

We presented a general framework for combining different pattern discovery methods and algorithm into a single one-click mining system. As a proof of concept of these ideas, we constructed a prototype that includes association and subgroup discovery. In a preliminary evaluation we saw that the resulting system is able to produce patterns corresponding to certain simple analysis goals without exposing the user to the technical side of the pattern mining method and its algorithms. The system is now ready to advance to the next stage of evaluation with a full scale empirical user study.

One limitation of the current approach is related to the expressiveness of the user utility model. Currently, the model weights as well as the feature functions are all positive and the learning takes place in primal space without kernel-based learning. On the one hand, this results in a sub-modular utility function for rankings, which can be efficiently optimized by the greedy algorithm within a reasonable approximation ratio. On the other hand, it prevents the learning of negative influence of some base features. Also the user is currently not able to learn, e.g., conjunctive conditions on the utility. For example she can not express that she is interested in a certain descriptive attribute only in the context of a specific target and otherwise not.

Another direction for future research is to closer investigate and extend the feedback options of the user. For instance, it appears to be desirable for negative feedback to differentiate between the cases of patterns that are simply invalid for the current analysis task and between patterns that are interesting in a vacuum but not novel. One approach to realize this is to introduce an explicit model of the user's background knowledge. This would allow to include subjective interestingness functions (see De Bie [2011]) into the utility model.

Acknowledgment

This work was supported by the German Science Foundation (DFG) under the reference number 'GA 1615/2-1'.

APPENDIX

A. SUBGROUP DEVIATION MEASURE

The subgroup deviation measure can be computed differently depending on the nature of the target attribute. For a categorical targets a_t with domain $V_t = \{v_1, \ldots, v_k\}$ the measure is given as

$$f^t_{\mathrm{dv}}(s) = \sum_{i=1}^{k} |\mathbf{p}^t_S(i) - \mathbf{p}^t_D(i)|$$

where $S = s(D)$ is the extension of s and $\mathbf{p}^t_X \in [0,1]^k$, defined by

$$\mathbf{p}^t_X(i) = |\{d \in X \colon a_t(d) = v_i\}| \,/\, |X| \ ,$$

is the empirical distribution of target values with respect to some subset of the data $X \subseteq D$. Hence, Eq. (1) includes well-known subgroup discovery measures such as Weighted Relative Accuracy and the Binomial Test. In case of a numerical target, the deviation can be computed by fitting a continuous distribution and then approximating the total variation distance. Here, for simplicity we assume a normal distribution with mean $\mu = \sum_{d \in D} a_t(d) / |D|$ and variance

$$\sigma_X^2 = \sum_{d \in X} (a_t(d) - \mu_X)^2 / (|X| - 1)$$

for the global data; and for the pattern a normal distribution with $\mu_S = \sum_{d \in s(D)} a_t(d) / |s(D)|$ and the same variance as in the global data. The target deviation can then be expressed much simpler as

$$f_{\mathrm{dv}}^t(s) = 2 \left| \mathrm{erf}\left(\frac{\mu_S - \mu_D}{2\sqrt{2\sigma_D^2}} \right) \right| ,$$

where erf denotes the Gauss error function. This is equal to twice the probability mass of $\mathcal{N}(\mu_D, \sigma_D^2)$ that is at most $(\mu_S - \mu_D)/2$ standard deviations away from the mean μ_D.

References

Abraham Bernstein, Foster Provost, and Shawndra Hill. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):503–518, 2005.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. *KNIME: The Konstanz information miner*. Springer, 2008.

M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2012.

Longbing Cao. Actionable knowledge discovery and delivery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):149–163, 2012.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

Tijl De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery*, 23(3):407–446, 2011.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

Bart Goethals, Sandy Moens, and Jilles Vreeken. Mime: a framework for interactive visual pattern mining. In *Machine Learning and Knowledge Discovery in Databases*, pages 634–637. Springer, 2011.

Henrik Grosskreutz, Mario Boley, and Maike Krause-Traudes. Subgroup discovery for election analysis: a case study in descriptive data mining. In *Discovery Science*, pages 57–71. Springer, 2010.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996.

Ingo Mierswa. Rapid miner. *KI*, 23(2):62–63, 2009.

Katharina Morik and Martin Scholz. The miningmart approach to knowledge discovery in databases. In *Intelligent Technologies for Information Analysis*, pages 47–65. Springer, 2004.

K. Raman, P. Shivaswamy, and T. Joachims. Online learning to diversify from implicit feedback. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 705–713. ACM, 2012.

Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. In *Proceedings of the 29th International Conference on Machine Learning, (ICML 2012)*, 2012.

Koen Smets and Jilles Vreeken. Slim: Directly mining descriptive patterns. In *Proceedings of the Twelfth SIAM International Conference on Data Mining (SDM 2012)*, pages 236–247, 2012.

Matthijs van Leeuwen and Arno J. Knobbe. Non-redundant subgroup discovery in large and complex data. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III*, pages 459–474, 2011.

Geoffrey I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *TKDD*, 4(1), 2010.

Geoffrey I. Webb. Filtered-top-k association discovery. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 1(3):183–192, 2011.

Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 773–778. ACM, 2006.

Lytic: Synthesizing High-Dimensional Algorithmic Analysis with Domain-Agnostic, Faceted Visual Analytics

Edward Clarkson[1], Jaegul Choo[2], John Turgeson[1], Ray Decuir[1] and Haesun Park[2]

Georgia Tech Research Institute
925 Dalney St., Atlanta, GA 30332-0834
{edward.clarkson, john.turgeson, ray.decuir}
@gtri.gatech.edu

School of Computational Science & Engineering
Georgia Institute of Technology
266 Ferst Drive, Atlanta, GA 30332-0765
{jaegul.choo, park}@cc.gatech.edu

ABSTRACT

We present *Lytic*, a domain-independent, faceted visual analytic (VA) system for interactive exploration of large datasets. It combines a flexible UI that adapts to arbitrary character-separated value (CSV) datasets with algorithmic preprocessing to compute unsupervised dimension reduction and cluster data from high-dimensional fields. It provides a variety of visualization options that require minimal user effort to configure and a consistent user experience between visualization types and underlying datasets. Filtering, comparison and visualization operations work in concert, allowing users to hop seamlessly between actions and pursue answers to expected and unexpected data hypotheses.

Categories and Subject Descriptors

H.4.2 [Information Systems]: Types of Systems–Decision Support

General Terms

Design, Human Factors

Keywords

Visual analytics, scientific intelligence, infovis

1. INTRODUCTION

General-purpose data analysis tools—spreadsheets—were at the forefront of the personal computer revolution in the late 1970s and early 1980s. As a so-called "killer app" that showcased PC utility, the benefits of digitized data helped to drive widespread corporate adoption of personal computing [7]. In the 2010s, the wide availability of large amounts of data from a variety of sources has shown the potential for a similar revolution centered on data, and more importantly the *knowledge* that can be extracted from it by algorithms and human ingenuity.

And yet, there are relatively few general-purpose tools for such large-scale data analysis, particularly for exploratory purposes. Broadly, visual analytics addresses this problem through the synthesis of sophisticated computational algorithms with human initiative, and especially our high-functioning visuospatial perception capabilities.

This paper presents *Lytic* (from the word Ana***lytic***), a flexible, domain-independent visual analytic (VA) system for interactive exploration of large datasets. Its design focuses on facilitating key data analysis tasks that occur across disciplines: filtering, transformation, comparison, and visualization. Data can be formatted in convenient (character-delimited flat files) formats, with the UI adapting to the data characteristics (# of items, data types, etc.). Users create visualizations (scatter plots, line charts, parallel coordinates, etc.) simply by choosing variables from dropdown lists.

Lytic also integrates dimension reduction and clustering (DR/C) algorithms to facilitate analysis of high-dimensional data (e.g., text document term frequencies). It thus contributes to the literature a scalable system suitable for domain experts (rather than visualization or algorithm experts) to explore a wide variety of datasets[1].

IDEA '13, August 11th, 2013, Chicago, IL, USA.

2. RELATED WORK

Several related lines of research have informed the Lytic system. Faceted classification is a library science concept derived from colon classification [24], which in turn has been independently replicated in database systems (cf. dimensional modeling [19]) and information visualization (e.g., in the Attribute Explorer [26]). Its core concept is that of classifying items of interest along multiple independent axes, allowing more powerful and descriptive classifications than strict hierarchical organizations. When implemented in software, this pattern (which is both a UI and a data model pattern) is commonly referred alternately as *faceted search*, *browsing,* or *navigation.*

This design pattern has also become common in e-commerce sites, for example: customers can browse for clothes by filtering or comparing available items based on their size, style, color, price, manufacturer, etc. Because users can easily determine what characteristics (or *facets*) are available and hop between them to perform filters, this method has proven to be powerful, flexible and user-friendly to a wide audience. This commercial popularity was preceded by their use in many research systems [5,30,31]. Notably, faceted UI systems commonly display information about the values within each facet (e.g., the number of items matching a given category) and dynamically update it with successive queries [12].

There has long been recognition of the importance of the exploratory search and its distinction from targeted information seeking [22]. Information visualization (*infovis*) systems in particular are concerned with that distinction, as much of their value comes not just from allowing users to answer specific questions, but from harder-to-quantify attributes such as allowing

[1] A 64-bit windows binary release is publicly available from http://tmt.gtri.gatech.edu/lytic.html

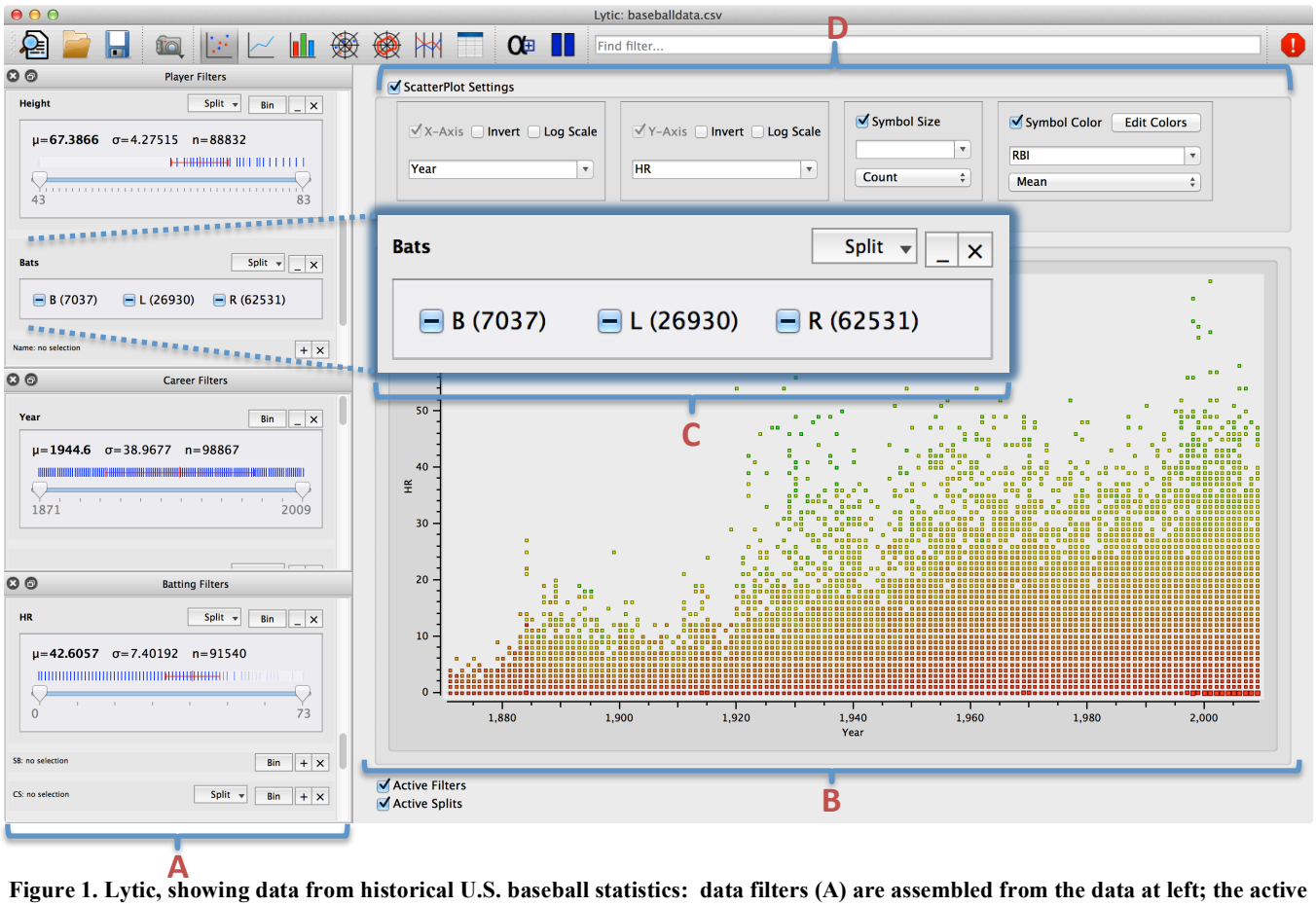


Figure 1. Lytic, showing data from historical U.S. baseball statistics: data filters (A) are assembled from the data at left; the active visualization (B) is at right. Filter choices update the active visualization and the choices available in other filter widgets. Inset (C) shows detail of a filter for the nominal variable *Bats* indicating a player's handedness. The view shows the number of home runs hit by players over time, configured by the common view controls (D).

users to ask *better* questions, or prompting users to ask questions they did not know they had in the first place [14].

Because software must first be derived from some implicit or explicit set of requirements, infovis work has therefore been concerned with the difficult question of what kind of end-user tasks it should support in order to help users achieve ill-defined, mutable, vague goals. One of the best known is Shneiderman's Visual Information Seeking mantra: "Overview first, zoom and filter, then details-on-demand" [25]. While the mantra concerns software, other work in both the infovis and VA communities characterizes the nature of both high- and low-level human activities and goals (Yi et al. provide a recent survey [32]). But informally, such activities include data filtering, comparisons, transformation, and visualization.

Infovis and VA systems have sought to support these kinds of tasks implicitly or explicitly in general-purpose analysis tool. Systems like Jigsaw [27], Apolo [8], and IN-SPIRE [29] all provide varying levels of support for these kinds of tasks in service of exploratory search goals. The overlapping domain of data warehousing within the database community also contributes to this space and commonly refers to these kinds of tools as a part of a *business intelligence* software environment.

Polaris [28] and its commercial successor Tableau exemplify this type of work, extending the Excel pivot table concept to provide a wide variety of filtering and visualization capabilities for arbitrary data. TIBCO Spotfire also began as an academic work [1] before becoming a commercial products. Many other purely commercial products also exist, such as SAS JMP, Pentaho, and many others.

DBMS-centric visualization environment tools generally act as an SQL abstraction layer, reducing the amount of programming expertise required. The Tioga-2 visualization environment [2] was an early example of providing a programming abstraction (boxes-and-arrows) specifically for visualization purposes. Zloof proposed a rendering-by-example [20] environment (building off his well-known query-by-example work [33]) that would allow users to build interactive renderings via a direct manipulation style interaction. More recent system include specialized designs such as building queries within a radial visualization [13] and querying electronic health record (EHR) data [15].

Influential infovis systems often employ brushing and linking interaction facilities to assist interactive exploration. The attribute explorer [26] consists of a set of linked histograms that show the distribution of the dataset along many dimensions, and dynamically update with filtering operations. Brushing scatterplots [4] visualize multidimensional data by forming a scatterplot matrix of arbitrarily many dimensions, and dimensional relationships can be explored by brushing items in the visualization. Both of these systems tightly couple a particular visualization technique with data exploration (filtering, comparison, interaction) actions: in contrast, Lytic decouples such operations from the visualization because 1) as the number of dimensions grows large, such small multiples-style approaches

become problematic and 2) it allows for operations on dimensions that are not being visualized.

Other systems have approached data exploration and visual analytics from a more algorithmic perspective, using machine learning or other data mining-oriented computation. The FODAVA Testbed system provides a library of old and new dimension reduction and clustering algorithms and provides a visualization environment to evaluate their performance on different datasets [10]. The VisIRR system uses the same library to implement a system for dynamic exploration of large-scale document corpora [9].

Lytic is distinct from previous work in two ways. First, it approaches its UI design from a faceted navigation rather than a RDBMS or pivot table orientation. Second, it directly integrates a scalable, general-purpose data exploration-and-visualization tool with algorithmic (DR/C) preprocessing.

3. DESIGN

Lytic is based on the data analysis component of a modeling and simulation management framework [11], and thus its origins are in analyzing large, high-dimensional simulation datasets for scientific and engineering applications. As a result, we think of it as a sort of *scientific intelligence* tool, complementing the existing business intelligence tools. It is implemented in C++ using the Qt framework, which provides the ability to compile a desktop application for Windows, Linux and OS/X.

As such, Lytic is designed as a generic data analysis tool to support common analytic activities that exist across application domains (e.g., filter, compare, transform, visualize). Its design imperative is to make common tasks as easy and fast as possible to complete, while providing facilities for more complex exploration through extension and customization. The focus on optimizing common tasks accelerates the human-in-the-loop analytic cycle, providing more opportunities for opportunistic and goal-driven insights.

Figure 1 shows the high-level organization of the tool, which is populated by an initial **data ingest and processing** step. **Data filter** widgets, which are created for each facet/dimension, are grouped at left. **Data view** choices (e.g., scatterplot visualization) at right. Filter operations update the backing **data model**, which is structured as a SQLite database, and in turn update the data view as well as other filters. **Data comparisons** notify the view that it should provide a partitioning of its representation along some dimension. The user may perform several types of **data transformations** on the data, including unit conversions and customized formulations of new variables. The following sections provide details of each of these topics.

3.1 Data Ingest and Processing

In keeping with this lowest-common-denominator support, Lytic reads row/column-oriented character-separated value (CSV) files. The convention used by Lytic and this paper is to treat columns as data dimensions or variables, and rows as distinct data items. A key component of Lytic is the integration of the algorithmic suite developed for the FODAVA testbed [10], which provides a variety of dimension reduction and clustering algorithms[2]. An end user can specify one or more algorithms to selected dimensions so that the use can leverage the algorithms to better analyze the data or use the tool to analyze the algorithms themselves. For very high-dimensional data (to us, more than ~1000 dimensions), the user can also augment the main data file with sparse matrix syntax auxiliary (high-dimensional data can also be specified in the CSV file).

Integrating DR/C algorithms directly helps analysts deal with high-dimensional data more effectively, which often occurs from automated processing of some source data, such as term frequencies in document data or image characteristics. DR/C algorithms thus summarize those dimensions efficiently, reducing the complexity of the data space for the analyst.

High-cardinality data (i.e., large number of data items/rows) also presents a scalability challenge for most DR/C algorithms, which are commonly $O(N^2)$ or worse in processing time, memory space, or both. Lytic is intended as an interactive end-user tool on commodity PC hardware, so it is important that ingest and load be limited to a reasonable amount of time—for us, a few minutes in the common case and no more than a few hours. As such, "high-cardinality" is a relative term for DR/C scalability; we have found

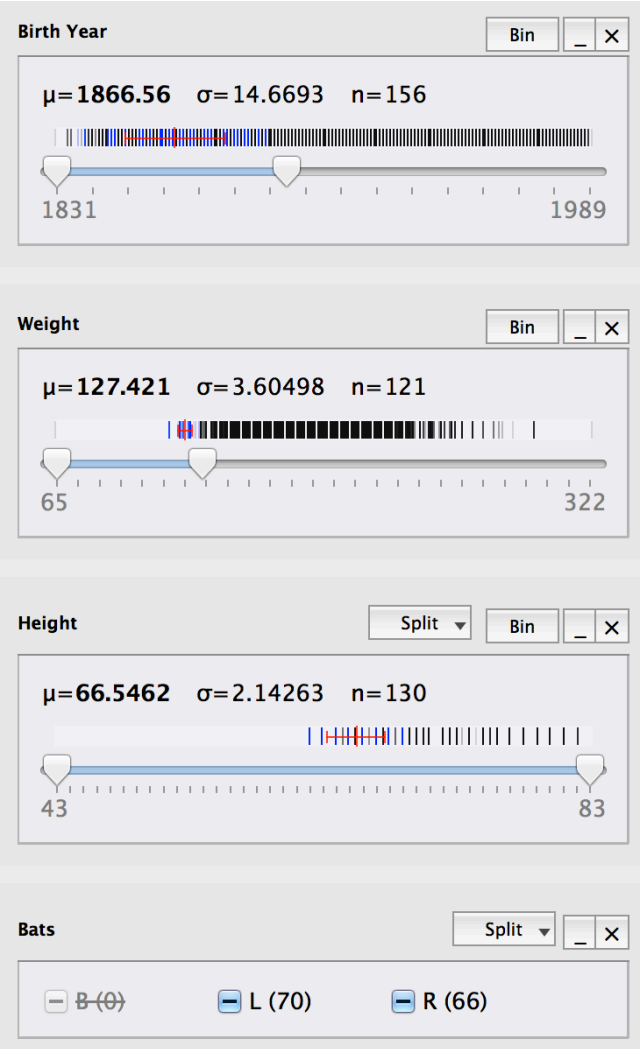
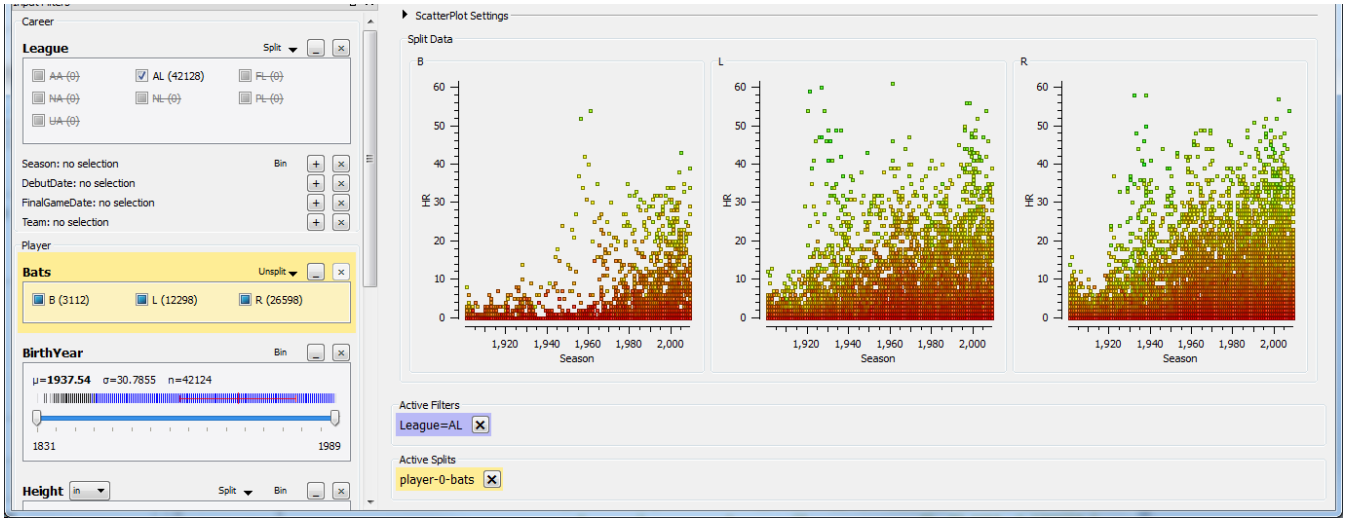


Figure 2. Three filter widgets for scalar data, which use double-ended sliders rather than checkboxes and render visualizations of the data distribution for that variable. Actions which would yield no data are disabled (Bats = 'B').

[2] The FODAVA suite includes newer techniques like linear discriminant analysis (LDA) and non-negative matrix factorization (NMF) as well as older algorithms like principal component analysis (PCA) or K-means clustering.



on modern laptop processors with SSD storage and 8-16 GB RAM this threshold to be ~2000 items.

Lytic itself can scale 2-3 orders of magnitude larger (see the following section), so we attempt to deal with this disparity by sampling and interpolation [3]: we randomly select a designated number (1000 by default) of rows and use only that data as input to the DR/C algorithms. Those results are then extrapolated to the extra-sample data items.

3.2 Data Model

The Lytic data model is similar to that of many OLAP-style tools. The combined CSV and synthesized DR/C data are imported into a single SQLite table, trading space efficiency for speed. All columns are indexed. This monolithic construction deviates from data warehousing and previously-described norms [12] because Lytic was initially targeted at scientific datasets with largely numeric, non-hierarchical data.

We define the data *scope* as the items/rows in the data model that have not been filtered out (see the following section): initially, all data are in scope. Filter operations are essentially SQL query generators the successively limit the scope of the data to smaller subsets. Since all columns are indexed, queries are executed efficiently, and become more efficient the more filters are put in place since the scope is necessarily reduced. Fields generated by DR/C algorithms are treated as any other filter, thus integrating synthetic and first-order characteristics of data items.

Lytic's scalability is dependent on the memory and speed of the user's workstation as well as the nature of the test data (the type and number of variables). On modern workstation-level hardware, we have found practical limits with reasonable interaction rates [6] to be between 100,000 and 1 million items with up to about 300 dimensions (after dimension reduction).

Lytic maintains a clean separation (with one minor exception, see §3.5) between its model and view, so other SQL implementations can easily replace the current Sqlite implementation (e.g., a column-oriented store like MonetDB), or even replaced by a non-SQL alternative.

3.3 Data Filtering and Comparison

Lytic's interaction design is focused on minimizing actions necessary to accomplish common operations: filtering, comparing, and visualizing. A faceted navigation-style UI makes as much data visible as possible and removing opportunities to make

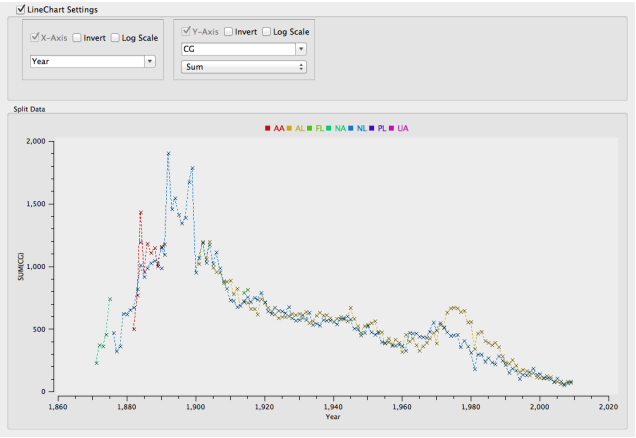


Figure 3. Comparison of split scatterplot (top) and line chart (bottom) views.

unproductive choices. Streamlining these operations reduces the barriers to iterative investigation through visual hypothesis testing and analysis. Filtering operations are accomplished by single actions: clicking a checkbox or dragging a slider handle. Dragging a slider handle dynamically queries the data view (subject to performance constraints).

Lytic forms a filter UI element for each field/column in the data model adapted to the nature of the data in the field: high-variation numeric data use a slider GUI element; nominal and low-variation scalars have a series of checkboxes (cf. Figure 1 inset) with parenthetical values indicating the number of data items matching that filter value. In Figure 1, there are 7037 B (baseball players who hit with **B**oth hands) players—many fewer than either **L**eft or **R**ight handed players.

For numeric (scalar) data, basic statistics (mean, std. dev., count) are calculated about the variable distribution. If there are sufficient distinct values, the UI includes a double-ended slider element to make range-based filters and renders a simple visualization of the data distribution). Blue marks indicate items that are in scope; black marks have been filtered out. Figure 2 shows three scalar filters (and one nominal); the user has filtered out all items with Weight greater than 135 and Birth Year greater than 1899, so all marks above those values are black in their respective distribution visualizations. However, there are black marks in the height filter as well, indicating how the filters from

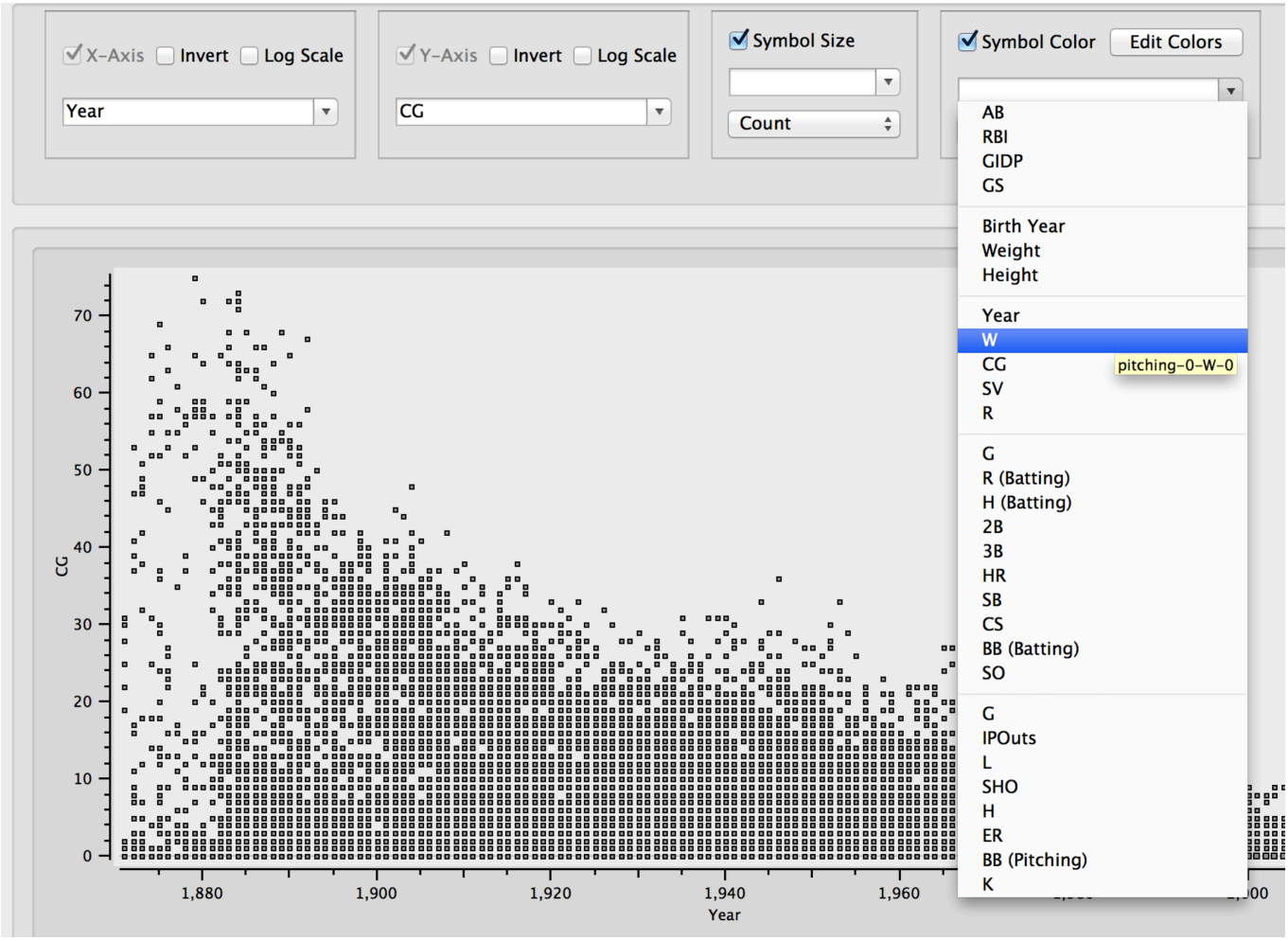


Figure 4. Plot slot detail view. Different slots have different options, and variable selection dropdowns are populated only with appropriate types.

one variable affect the distribution of data in another: as one might expect, lightweight players born in the 19th century are relatively short. Furthermore, the user is prevented from making unproductive UI choices: there are no switch hitting players left in scope, so the *B* value in the Bats filter is disabled.

The *Split* operator—available on nominal filters—compares data by partitioning it on the filter values. Data views are responsible for visualizing the comparison as appropriate for each view type (see Figure 3 and the following section).

3.4 Data Views

A *data view* is a fundamentally a method of mapping data to the characteristics of a visual *glyph*. We term these characteristics *slots,* similar to the Polaris/Tableau *shelf* concept). Examples of glyph slots are the two position dimensions of a mark on an X/Y Cartesian coordinate system, a mark's color, size, orientation, etc. Thus, a data view is described by how glyph characteristics are used in the data mapping, and a user creates a specific view instantiation by choosing what actual data to map to those slots. Lytic is currently limited to a single view at a time, and there are currently no brushing operations from the view to filter widgets. The lack of brushing operations is due to computational obstacles: recent work shows promise for scalable brushing using precomputed "data tiles", but even so scales only to modest numbers of dimensions [21].

All data views reflect current filters and splits. Views only display data within the current filter scope, and any split comparisons are handled in a view-appropriate manner: line charts use color to denote different split values within the same chart (cf. Figure 3, top), while scatter plots display multiple instances of the view with the distinct data partitions (cf. Figure 3, bottom). The latter allows the easy construction of small multiple views of the dataset. There are currently five major view types implemented: scatterplots, line charts, bar charts, parallel coordinates [16], and a tabular data view, with polar variants of the scatter and line charts.

For example, the Lytic scatterplot view can map data to the glyph X/Y position, color and size; a user defines a specific scatter plot view by selecting variables from her dataset to map to each slot: time to the X-axis; distance to the Y-axis, etc. Other views have fewer (e.g., line chart has only X and Y axis slots) or different slots (the parallel coordinates view has a dynamic number of slots, all for parallel Y axes).

Different slots have different properties: scalar-only slots can be inverted or log-scaled; some have custom transformations (e.g., color mapping); some views have dynamic slots that can be added or removed. All data views list what slots are used at the top of the visualization area, and within each slot a drop-down list of variables is available to choose which to map to that slot. Thus, to

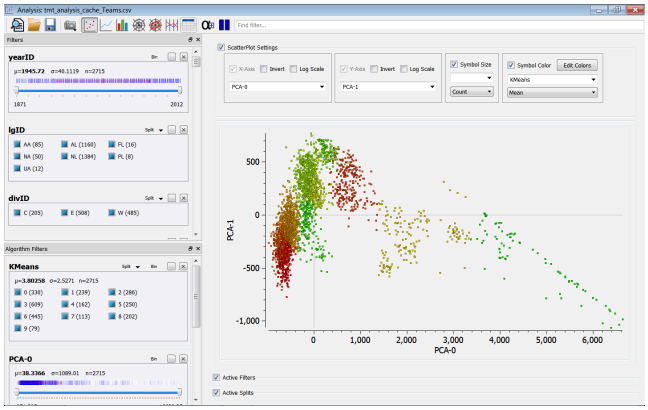


Figure 6. PCA visualization of baseball team statistics.

create a new visualization, the user simply selects the desired variable from the list, and the view updates with the new data.

Only valid selections are shown or allowed from the view drop-down boxes: variables that are not the appropriate data type for a slot (e.g., nominal variables for a numeric axis) are not included. Some slots are *aggregate*, meaning the view will combine multiple data points (grouping by the same value the non-aggregate slots in that view); for those slots a second drop-down box is shown with the available aggregate functions (e.g., mean, standard deviation, etc.).

In Figure 4, the user has already selected to visualize Year vs. CG (Complete Games), and is preparing to make a selection for the Color slot. Variables such as Team or League are not present since they are nominal. As the user makes selections in the slots, the view updates immediately; the current view shows that over time, the number of pitcher Complete Games has decreased dramatically. If the user selects W (Wins) from the dropdown, she could investigate if the number of Complete Games is correlated with Wins in some way.

3.5 Data Transformation

Other analysis capabilities include data transformations via automated unit conversions, creating new computed variables, and custom analysis via a simple plugin API. Unit conversions are specified in an XML file containing relevant units and conversion factors. In the data ingest step, the user can specify units for a variable, and any valid conversions are automatically presented within the filter widgets. Users can also set global preferences for metric or imperial unit systems, and any non-conforming units will be transformed.

New computed variables can be created by the Bin dialog (which bins scalar variables per user input) or in the more general case, arbitrary SQL formulas using existing variables (this is the only Lytic feature that directly exposes the SQL data model underpinnings).

Lytic can be configured to launch external programs ("data handlers"), passing any subset of the current data scope (as a file) for customized analysis. Data is selected from the active view (e.g., lasso-selecting scatterplot points, or selecting rows in the tabular view), and a right-click context menu shows all available handlers. A built-in default shows the entire record for each selected data item, but generally the fields can be processed for any purpose. For example, a handler could read the contents of a file path that contains 3D scene data and render it, or could implement a customized statistical analysis with an R script.

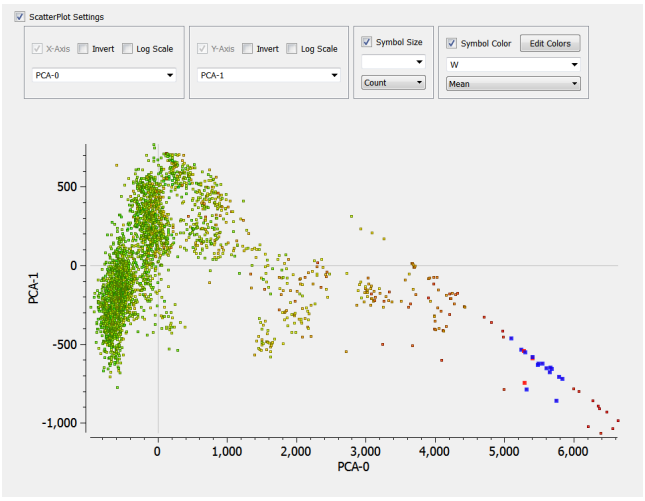


Figure 5. The same view as Figure 6, but with color mapped to Wins instead of cluster label. The user has selected a group of items in the lower left (bold blue items).

4. USAGE SCENARIO

To illustrate Lytic's end-to-end utility, we provide a high-level usage scenario prior to providing more details on system design and implementation. Sports data analysis is an interesting domain for visual analytics that has seen some recent attention [23], providing motivated users (fans, teams, and media) and a rich set of qualitative and quantitative data.

4.1 Narrative

Consider the perspective of a writer, Sally, doing research for a book on the general history and evolution of the game of baseball. She is aware of the Lahman Baseball Archive[3], which contains pitching, batting, fielding, managerial, and other statistics from U.S. professional baseball.

Sally starts Lytic and selects the Teams CSV file from the several dozen different collations in the archive, which she assumes has summary statistics from each team season. Aside from general research, she is interested in seeing if a machine learning tool can suggest any interesting new ways of categorizing or looking at historical teams, so she also adds a K-Means clustering and a PCA (principal component analysis) dimension reduction algorithm in the load dialog using the default parameters (10 clusters and 2 dimensions, respectively).

Lytic processes the data and presents the initial view of the team

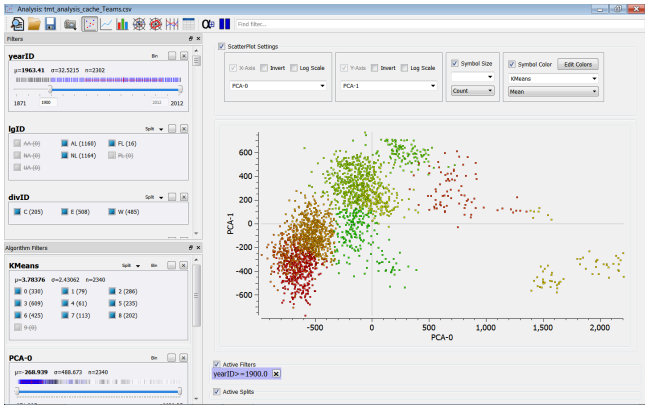


Figure 7. Same view as Figure 6 and Figure 5, but with a filter in place for seasons 1900 and after.

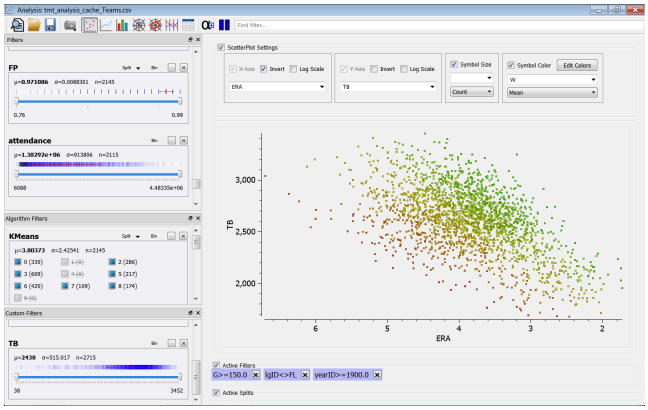


Figure 8. Scatterplot view of Earned Run Average (ERA) vs. Total Bases, with color mapped to Wins, showing the tradeoff between offense and defense and the advantage of being near the Pareto front.

data. Sally is first interested in looking at what makes a successful baseball team, but she first takes stock of the various filters built from the file. She first notices that there is data dating from 1871, when professional baseball was in its infancy, and that there is also data from older defunct professional leagues. There are also filters for the K-Means clusters and PCA dimensions. That kind of data is new to her, so she decides to see what it looks like using the default scatterplot view: she selects the PCA-0 and PCA-1 variables in the X- and Y-axis slots, and selects the K-Means variable in the color (see Figure 5).

She is intrigued by the arrangement of the teams in the plot, and wonders whether the algorithm results have any bearing on team quality or success, so she changes the color slot to Wins (see Figure 6). This seems to have some relationship with the reduced space, primarily in the lower right that all appear to be low-win teams (the default color mapping is a stoplight pattern). She lasso-selects several points in the lower-right section, and right-clicks to examine their details. She finds that they are all from the 19th century, so she immediately goes to confirm her guess by changing the color slot to Year.

That appears to confirm her suspicion, so she filters out all seasons prior to 1900 (see Figure 7). Doing so dynamically filters the visualization, and when she releases the year slider it updates the other filters—whose distribution is visible by default—indicating that (as she already knew) almost all the historical professional leagues were gone by 1900, and more importantly

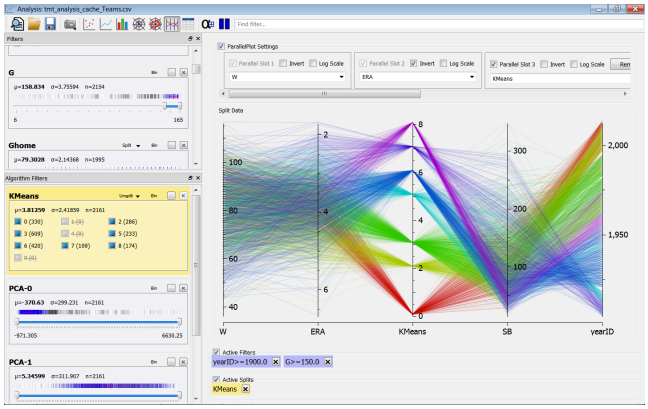


Figure 9. A parallel coordinates view of several different variables. Color is mapped to cluster label.

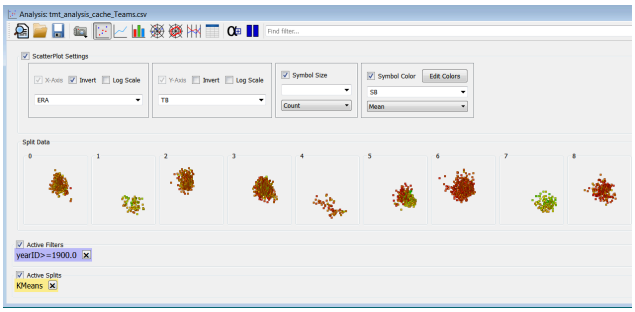


Figure 10. A small multiples scatterplot view of ERA vs. Total Bases (cf. Figure 8), but with color mapped to Stolen Bases and the view split on cluster value. This shows two clusters (#1, 2nd from left; #7, 2nd from right) that show similar teams having high stolen base totals, low ERAs and low Total Bases.

shows that one of the clusters has been removed from the focus data.

Sally then reverts to her original line of inquiry: do the clusters have any relationship to team quality as expressed by Wins? She maps color to Wins—there is still no obvious relationship on the high end, but again the clusters on the lower right are consistently on the lower end. She again lasso-selects and examines their details, finding that one sub-group are teams from 1994 and one from 1981. Sally might recognize these seasons as ones impacted by player strikes; otherwise, this would clearly prompt additional investigation outside Lytic.

This prompts Sally to wonder how much of the PCA components and cluster values are related to the number of games played in the season—she switches the color slot to G, and finds that a great deal of the first component is due to the number of games in the season; this leads her to wonder if the algorithmic data would be better applied to rate statistics (e.g., attendance/game, adjusted for population). She decides to investigate a few more items and leave the re-processing for later.

Still interested in team success, she uses the scatterplot view to look at team wins versus several statistics and finds some interesting tradeoffs between offensive variables (e.g., Home Runs and Triples are slightly negatively correlated). This causes her to wonder about overall team construction: is there a tradeoff between offense and defense? To test this, she creates a custom variable, Total Bases, which is calculated from Home Runs, Triples, Doubles and Hits; she plots total bases versus Earned Run Average (a measure of pitching/defensive effectiveness), and sets the color to wins (see Figure 8). Interestingly, there does appear to be a tradeoff at work—the more offense a team has, the worse its defense, and that the better teams are closer to the Pareto front. Sally realizes that teams are constructed under resource constraints (player supply, salaries), so this makes some sense

Piqued by this realization, she then uses the parallel coordinates view to look at several variables at once (see Figure 9). She happens to place the GHome (home games) variable next to the year, finding that there are no teams from 2006-2010 with any recorded home games. She suspects this must be a data collection problem, since she is certain baseball was played during those years.

She eventually backtracks to looking at the K-Means cluster data (also splitting on that variable to color-code the view), and notices a few clusters do have an association with low Stolen Base totals. That leads her to wonder: is the offensive/defensive tradeoff (as

measured by Total Base) mitigated by Stolen Bases, which are another way to generate offense?

She returns to the scatterplot view of ERA vs. Total Bases, but this time colors nodes by Stolen Bases and splits the view on the cluster value, creating a small multiples-style view (see Figure 10). She is excited to see that some of the clusters *do* have some additional meaning outside of the number of games or the year: two of the clusters appear to represent teams that steal a lot of bases but otherwise are successful due to better pitching.

4.2 Discussion

Sally demonstrated several benefits of visual analytics as supported by Lytic. First, outlier detection is a common task (either intended or serendipitous), and in the context of end-user data analysis, often takes the form of investigating data quality and provenance. If a data point is anomalous, in different contexts it is often the case that the data collection is faulty rather than encountering a truly novel case—a useful outcome in and of itself. Conversely, outliers that are really novel are the canonical case of sidetracking for opportunistic insight, regardless of the original goal or lack thereof.

The benefits of combining a general purpose VA tool with even basic DR/C algorithms are also significant, simply because their data provide a new jumping-off point for further investigation, potentially leading to insights only a human might not have considered.

A few points of Sally's story also pointed out areas for potential improvement. Sally is a writer or journalist, with no background in algorithms—as a result, it is somewhat of a stretch to assume that she would be asking for "PCA" or "K-Means" "algorithms", much less perform even basic configuration, such as selecting an appropriate *k* value. Revising the end-user nomenclature—e.g., *data map* for 2-dimensional DR results; *automatic categories* for cluster results—may be beneficial for some user audiences, as well as simplifying or eliminating configuration options. Conversely, if we have a user-friendly method of employing DR/C algorithms, it would be useful to have inline processing to either include new or exclude spurious features from within the analytic environment: in Sally's case, including more rate statistics or remove year as features.

Finally, although the scenario did not address truly high-dimensional data such as those from a document term frequency matrix, DR is crucial to enabling a manageable navigation environment for those types of datasets.

5. LIMITATIONS AND FUTURE WORK

Lytic has a number of general limitations that we should acknowledge and are implicit future work. Hierarchical facets and similar data—particularly that which can be automatically generated, such as a hierarchy of time intervals or geographic location granularities—are a key omission from the current tool. Similarly, Lytic has no first-class notion of either time or location data types (e.g., Date/Time or latitude/longitude information), and the current view have more options for numeric vs. categorical data. Using Lytic to monitor streaming data would be useful as well, but there is no current support for updating the data model once it is built.

Working on Lytic has also convinced us of one of the conclusions of a recent survey of data analysts [18]: that "data wrangling" is a significant barrier to making full use of tools like Lytic, and that more work [17] is necessary to solve end-to-end visual analytic problems.

Mixed-initiative visual analytic tools that respond to users previous interaction with the same or similar datasets. For example, learning from user's data ingest choices to make better default selections in the future. Finally, we hope to make Lytic a more open tool—including a visualization plugin system—for the visual analytic community as both a research and practitioner platform.

6. ACKNOWLEDGMENTS

The work of these authors was supported in part by the National Science Foundation grant CCF-0808863. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

1. Ahlberg, C. Spotfire: an information exploration environment. *SIGMOD Rec. 25*, 4 (1996), 25–29.
2. Aiken, A., Chen, J., Stonebraker, M., and Woodruff, A. Tioga-2: A Direct Manipulation Database Visualization Environment. (1996), 208–217.
3. Bae, S.-H., Choi, J.Y., Qiu, J., and Fox, G.C. Dimension reduction and visualization of large high-dimensional data via interpolation. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM (2010), 203–214.
4. Becker, R.A. and Cleveland, W.S. Brushing Scatterplots. *Technometrics 29*, 2 (1987), 127.
5. Capra, R.G. and Marchionini, G. The relation browser tool for faceted exploratory search. *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, (2008), 420–420.
6. Card, S.K., Robertson, G.G., and Mackinlay, J.D. The information visualizer, an information workspace. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (1991), 181–186.
7. Ceruzzi, P. and Grad, B. Guest Editors' Introduction: PC Software–Spreadsheets for Everyone. *IEEE Annals of the History of Computing 29*, 3 (2007), 4–5.
8. Chau, D.H., Kittur, A., Hong, J.I., and Faloutsos, C. Apolo: making sense of large network data by combining rich user interaction and machine learning. *Proceedings of the 2011 annual conference on Human factors in computing systems*, (2011), 167–176.
9. Choo, J., Lee, C., Clarkson, E., et al. VisIRR: Interactive Visual Information Retrieval and Recommendation for Large-scale Document Data. *In submission to IEEE Transactions on Visualization and Computer Graphics (TVCG)*, .
10. Choo, J., Lee, H., Liu, Z., Stasko, J., and Park, H. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. *Proceedings SPIE 8654, Visualization and Data Analysis*, (2013).
11. Clarkson, E., Hurt, J., Zutty, J., Skeels, C., Parise, B., and Rohling, G. Supporting robust system analysis with the test matrix tool framework. *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ACM (2013), 23–34.
12. Clarkson, E., Navathe, S.B., and Foley, J.D. Generalized formal models for faceted user interfaces. (2009), 125–134.
13. Draper, G.M. and Riesenfeld, R.F. Who votes for what? A visual query language for opinion data. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1197–1204.
14. Fekete, J.-D., Wijk, J.J., Stasko, J.T., and North, C. The Value of Information Visualization. In A. Kerren, J.T. Stasko, J.-D.

Fekete and C. North, eds., *Information Visualization: Human-Centered Issues and Perspectives*. Springer-Verlag, Berlin, Heidelberg, 2008, 1–18.
15. Huser, V., Narus, S.P., and Rocha, R.A. Evaluation of a flowchart-based EHR query system: a case study of RetroGuide. *Journal of biomedical informatics 43*, 1 (2010), 41–50.
16. Inselberg, A. The plane with parallel coordinates. *The Visual Computer 1*, 2 (1985), 69–91.
17. Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J. Wrangler: interactive visual specification of data transformation scripts. *Proceedings of the 2011 Conference on Human Factors in Computing Systems (CHI)*, ACM (2011), 3363–3372.
18. Kandel, S., Paepcke, A., Hellerstein, J.M., and Heer, J. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2917–2926.
19. Kimball, R. A dimensional modeling manifesto. *DBMS 10*, 9 (1997), 58–70.
20. Krishnamurthy, R. and Zloof, M. RBE: Rendering by example. *Proceedings of the Eleventh International Conference on Data Engineering, 1995*, (1995), 288–297.
21. Liu, Z., Jiang, B., and Heer, J. imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum 32*, 3pt4 (2013), 421–430.
22. Marchionini, G. and Shneiderman, B. Finding Facts vs. Browsing Knowledge in Hypertext Systems. *Computer 21*, 1 (1988), 70–80.
23. Pileggi, H., Stolper, C.D., Boyle, J.M., and Stasko, J.T. SnapShot: Visualization to Propel Ice Hockey Analytics. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2819–2828.
24. Ranganathan, S. *Colon Classification*. Madras Library Association, Madras, 1933.
25. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, (1996), 336–343.
26. Spence, R. and Tweedie, L. The Attribute Explorer: information synthesis via exploration. *Interacting with Computers 11*, 2 (1998), 137–146.
27. Stasko, J., Görg, C., and Liu, Z. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization 7*, 2 (2008), 118–132.
28. Stolte, C., Tang, D., and Hanrahan, P. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65.
29. Thomas, J.J., Cowley, P.J., Kuchar, O., Nowell, L.T., Thompson, J., and Wong, P.C. Discovering knowledge through visual analysis. *Journal of Universal Computer Science 7*, 6 (2001), 517–529.
30. Wilson, M., Russell, A., and Smith, D.A. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM 49*, 4 (2006), 47–49.
31. Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2003), 401–408.
32. Yi, J.S., Kang, Y., Stasko, J.T., and Jacko, J.A. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1224–1231.
33. Zloof, M.M. Query-by-example - Operations on hierarchical data bases. *Managing Requirements Knowledge, International Workshop on*, IEEE Computer Society (1976), 845.

A Process-Centric Data Mining and Visual Analytic Tool for Exploring Complex Social Networks

Denis Dimitrov
Georgetown University
Washington DC, USA
dd322@georgetown.edu

Lisa Singh
Georgetown University
Washington DC, USA
singh@cs.georgetown.edu

Janet Mann
Georgetown University
Washington, DC, USA
mannj2@georgetown.edu

ABSTRACT

Social scientists and observational scientists have a need to analyze complex network data sets. Examples of such exploratory tasks include: finding communities that exist in the data, comparing results from different graph mining algorithms, identifying regions of similarity or dissimilarity in the data sets, and highlighting nodes with important centrality properties. While many methods, algorithms, and visualizations exist, the capability to apply and combine them for ad-hoc visual exploration or as part of an analytic workflow process is still an open problem that needs to be addressed to help scientists, especially those without extensive programming knowledge. In this paper, we present Invenio-Workflow, a tool that supports exploratory analysis of network data by integrating workflow, querying, data mining, statistics, and visualization to enable scientific inquiry. Invenio-Workflow can be used to create custom exploration tasks, in addition to the standard task templates. After describing the features of the system, we illustrate its utility through several use cases based on networks from different domains.

1. INTRODUCTION

More and more social scientists and observational scientists have a need to understand and analyze complex network data sets. They need tools and methods that give them the opportunity to explore these data sets in an ad hoc manner or as part of an analytic workflow process. Kandel et al. [19] reiterate this need, saying that

> "little visualization research addresses discovery, wrangling or profiling challenges...Visual analytics tools that enable efficient application and assessment of these data mining routines could significantly speed up the analysis process."

Our work looks to help fill this gap by improving the capabilities of observational scientists to apply and assess data mining methods during data analysis.

IDEA'13, August 11th, 2013, Chicago, IL, USA.

This paper presents a tool (Invenio-Workflow) that supports exploratory analysis of network data by integrating workflow, querying, data mining, statistics, and visualization to enable scientific inquiry. Invenio-Workflow allows scientists to conduct a workflow process that adds data into the tool, queries the data, conducts a data mining task using one or more algorithms, and compares the results of the algorithms in a table or as an interactive graph visualization.

Some exploratory tasks that this tool supports include: finding communities/clusters that exist in the data using different algorithms, e.g. modularity and betweenness, predicting node labels using node/edge features and graph structure, identifying regions of similarity or dissimilarity between two data sets, querying and analyzing uncertain graphs, running network analysis using R, and finding and highlighting nodes with important centrality properties.

As an example, suppose a biologist wants to better understand group structure in an animal observational data set using Invenio-Workflow. This biologist may decide to create a workflow that compares the results of different clustering/community detection algorithms for the animal population's social network. Since the biologist may not be familiar with different community detection algorithms, she may want to compare the outputs of different methods to see which one matches intuition. Invenio-Workflow helps the biologist accomplish this by letting her setup a workflow that gets data from a file or a database, runs different clustering algorithms, and allows for visual exploration of the results. Figure 1 shows an example workflow for this scenario. While some standard task templates exist in Invenio-Workflow, a user can drag different widgets to create custom workflows to support exploration of data or existing analytic processes.

The contributions of this work include: 1) the development of a prototype process-centric, visual analytic tool; 2) a workflow process that includes visual, data mining, and graph query widgets for custom, exploratory analysis of network data; 3) integration of a graph query engine into the workflow process; and 4) an empirical demonstration of the utility of the proposed workflow design using a complex dolphin observation network and a citation network.

2. RELATED LITERATURE

2.1 Network/Graph Visual Analytic Tools

A number of excellent tools have been developed for exploring network data. Some of them are more general systems or toolkits [3, 4, 6, 7, 9, 12, 15, 18, 27, 26, 32, 14, 16, 34], while others are specialized for a specific task or analysis

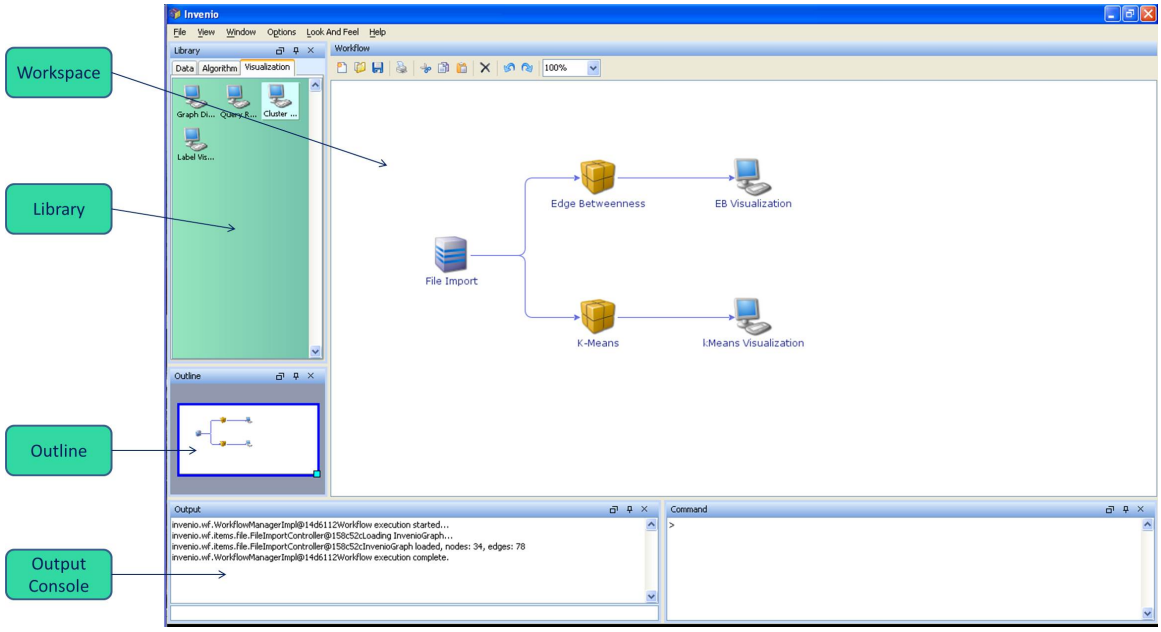


Figure 1: Clustering workflow in the context of Invenio-Workflow UI.

[8, 20, 22]. NodeXL [14] is an exploration tool that is integrated into Excel. Jung [27] is a toolkit containing a number of data mining algorithms. Prefuse [15] and Gephi [6] have more extensive visualization support. Guess [4] has a query language for manipulating graph visualization. While all of these tools have filtering, coloring, and interactive exploration capability, we view these tools as *data-centric* tools. In other words, they are stand-alone applications that focus on graph visualization and interactive manipulation of these graphs. In contrast, Invenio-Workflow is designed to be *process-centric*. Instead of focusing on manipulating and analyzing a single graph or network, our tool focuses on the entire process of data analysis and scientific inquiry, allowing the user to design and execute a workflow process that maps to a particular task of interest.

2.2 Workflow Tools

Orange [10], RapidMiner [24], and WEKA [13] are all examples of data mining / machine learning tools that consider the workflow process in their design. Orange is the most sophisticated in terms of workflow design, allowing users to create their own data mining workflow processes. It is the closest in design to Invenio-Workflow. Similar to Invenio-Workflow, widgets are used to setup an analytic workflow process. RapidMiner is the most sophisticated in terms of data mining support and visualizations. Its workflow support is more limited. The difference between these two tools and Invenio-Workflow is the goal of the tools. The developers of Orange have created a data mining tool that lets users setup different data mining processes. Invenio-Workflow is a scientific inquiry tool for graphs that lets users focus in on analyses specific to graphs of varying sizes. Its emphasis on graph data also distinguishes Invenio-Workflow from such mature, extensive software platforms as Amira [5] and Voreen [35], which incorporate the concept of dataflow for creating highly flexible, interactive visualizations of volumetric data.

Viztrails [31] is a tool that is designed with a similar purpose as our tool. It also combines databases, workflow systems, and visualization tools that are individually incomplete into an interactive scientific inquiry engine. The tool's emphasis use-case is provenance. In contrast, our tool incorporates extensive graph mining and statistics components. The graph mining includes a number of custom algorithms for community detection, bias analysis, etc. For statistical support, Invenio-Workflow incorporates *R* [30]. Many of these analyzes are not supported by Viztrail.

3. TOOL DESCRIPTION

Invenio-Workflow provides an interactive, process-centric environment for graph exploration. Users visually define a process choosing from the available widgets and connecting them to designate the desired data flow. Each widget represents a logically separate piece of functionality, such as loading a graph from a file or executing a specific algorithm.

The Invenio-Workflow interface, shown in Figure 1, consists of several panels. The *Workspace* serves as an editor for constructing workflows, i.e. inserting, connecting, and re-arranging widgets. In addition, it has capabilities for zooming, copying and pasting widgets, and saving workflows for later loading and reuse.

Desired widgets are inserted by dragging from the *Library* panel into the *Workspace*. The *Library* contains several categories of widgets. Data import widgets are used to load data from different sources and formats. Algorithm widgets encompass data processing functionality: specific clustering algorithms, node labeling algorithms, and graph query execution are the main ones in our tool. Visual widgets include a general graph visualization widget, as well as several visualizations best suited for analyzing results of specific algorithms.

The *Outline* panel facilitates navigating large workflows. Sliding the workflow focus window over the workflow outline and adjusting its size brings the corresponding workflow area

into the *Workspace* editor. The *Output* panel is a logging console, to which Invenio-Workflow and its components post messages, warnings, and errors.

We will demonstrate the Invenio-Workflow interface using the previously mentioned motivational example of clustering / community detection algorithms in the well-known karate social network [36]. In this network data set nodes correspond to members of a karate club at a US university and edges represent friendships. Sociologists have used this network for a number of different studies, including ones related to group structure and models of conflict. Sociologists may be interested in seeing which algorithm captures the expected group structure most accurately, or in cases where they do not know the group structure, compare the outputs of the different clustering algorithms to identify group structure.

The *Workspace* in Figure 1 shows the workflow created for our example task - comparing clustering algorithms and visually exploring the results. The File Import widget loads the data set graph from a text file in one of the supported formats. This graph is forwarded to two clustering algorithms: Edge Betweenness Clusterer and K-Means Clusterer. The output of each algorithm is connected to an interactive Cluster Visualizer. This visualizer allows the researcher to see which nodes are member of different clusters, as well as compare the similarities and differences between the two clusterings.

Before a workflow can be executed, some widgets may need to be configured. For example, the File Import widget needs to be supplied with the location of the data set file(s), by opening up its configuration editor. Some clustering algorithms also need parameters specified. Figure 2 shows the configuration panel for the Edge Betweenness Clustering algorithm. After configuring the widgets and executing the workflow, the results at different steps can be observed by opening up the result view of the corresponding widget. Depending on its functionality, a widget may offer a configuration editor, a result view, both, or possibly neither. They open up in separate frames, which the user can dock or float anywhere over the workspace. They can remain open, as long as the widget is present in the workflow, so that the user can observe the configuration and results of any number of widgets simultaneously.

Executing the demonstrated workflow, we obtain the Cluster Visualization result views in Figure 3a and Figure 3b. The information panel provides the user with basic statistics about the generated clusters, such as the number of clusters and the minimum, maximum, and average number of nodes in a cluster. Selecting one or more clusters from the list highlights the corresponding nodes in the graph view. There is also an option to hide the remaining nodes. By viewing the resulting clusterings from the two algorithms, a researcher can analyze the clusters that are similar in both algorithms, and those that differ.

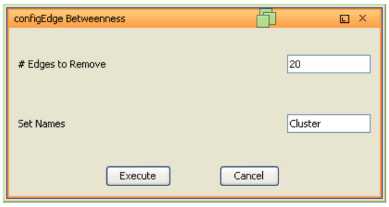


Figure 2: Configuration Panel for Edge Betweenness Cluster widget.

The process-centric approach has several advantages. It provides for ease and flexibility for visually defining, changing, and executing analytical tasks as opposed to writing code or using monolithic, pre-defined tools. Saving and loading workflows makes it possible to repeat the analysis in the future, possibly on a different data set, as well as to share it with other users.

Our implementation is an initial proof of concept, containing widgets that handle only a small subset of possible graph manipulation and visualization tasks. Therefore, one of our design priorities was an open architecture that allowed for ease when adding new widgets. In the straightforward case, a new widget is added by implementing a simple interface and registering the widget with Invenio-Workflow. Additionally, during deployment the widget may be associated with a configuration editor and / or result view. It may declare the expected number and type of input data. For more complex validation, the widget may specify validators to be invoked by the framework at runtime.

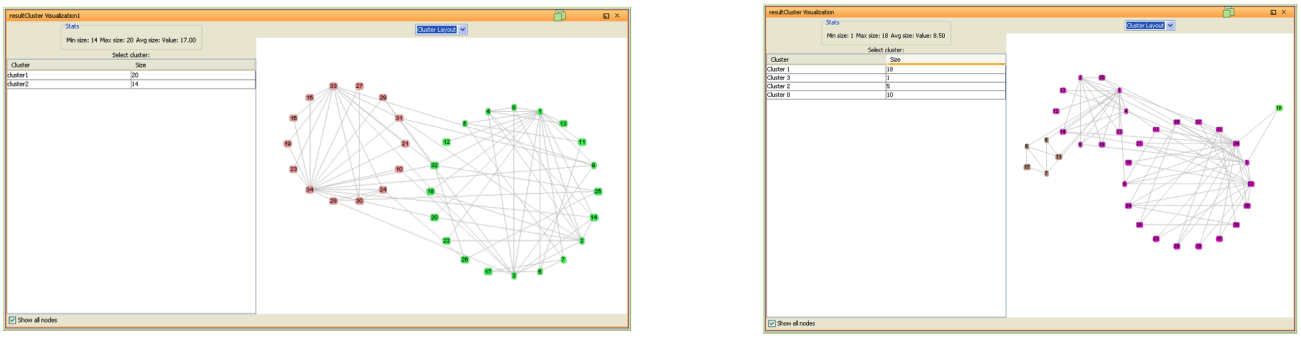
Invenio-Workflow is written in Java. For most flexibility, the workflow processing engine and the individual widgets are a custom implementation. However, Invenio-Workflow relies on several toolkits to support various other aspects of its functionality. Workflow diagrams are based on the open-source JGraph framework [1]. JIDE Docking Framework [2] is used for dockable window support. Graph visualizations in the corresponding widgets build upon the Prefuse visualization toolkit [15, 28]. The uncertain graph data model is implemented as an extension of the JUNG graph data model [17, 27]. Node labeling algorithm implementations are provided by GIA [25]. R [29, 30] is integrated for statistical analysis. Graph query processing is delegated to a query engine that was developed by our group.

4. USE CASES

This section presents use cases based on two real-world data sets, introducing several widgets and demonstrating how they can be customized and combined for accomplishing the desired analytical tasks. For these use cases, we use a personal computer with dual core 2.9 GhZ processor and 8 GB of memory. The graphs analyzed both fit into main memory.

4.1 Dolphin observation network use case

This use case considers a dolphin social network based on approximately 30 years of study of a dolphin population in Shark Bay, Australia [23]. The data set includes demographic data about approximately 800 dolphins, represented as graph nodes and social interactions between these dolphins, captured as approximately 29,000 edges. A researcher observing a particular animal may be uncertain about its identification, features, or behavior. This uncertainty can be expressed as existence probabilities between 0 and 1, associated with nodes and/or edges and as categorical uncertain attributes, representing discrete probability distributions over the set of possible attribute values. Dolphin vertices in the data set include certain attributes (id, confidence, dolphin_name, birth_date) and uncertain attributes (sex_code, location, mortality_status_code), while edges have attributes (id, confidence).



(a) K-Means.

(b) Edge Betweenness.

Figure 3: Result view of 2 different Cluster Visualization Widgets.

A scientist can obtain a general idea about the size, density, and connectivity of the network by creating a very simple workflow that loads the graph and feeds it into a standard graph visualization widget. The latter displays the graph along with basic statistics, such as number of vertices / edges and average degree. To give the users the capability for in-depth exploration and comparison of uncertain graphs, we have designed a prototype SQL-like query language that combines elements of relational, uncertain, and graph databases [11]. In this work, we have incorporated our query engine implementation into a widget that allows users to write their own ad-hoc uncertain graph comparison queries.

Our team met with observational scientists who work with the dolphin population and developed a list of typical queries that they would like the capability to issue when analyzing this dolphin social network and its inherent uncertainty, including:

- Selecting the number of associates and sex composition of associates for male and female dolphins, respectively, using the most probable value of the *sex_code* attribute.
- Visualizing the union, intersection, difference, and bi-directional difference between the ego-networks of a particular dolphin during two different years, where the confidence of relationship existence is above a specified threshold.
- Finding the common associates (friends) of two specific dolphins with a relationship confidence above a certain threshold.
- Calculating a measure of structural and semantic similarity between ego-networks of two particular dolphins.

These and many other queries can be expressed in the proposed language and executed using the Query widget. Because the query result represents a relation whose tuples may contain graphs, vertices, edges, attributes, and / or primitive types, depending on the query, the Query Result widget helps the user to visually explore the result. Figure 4 shows the result of executing a query that returns union, intersection, and difference between ego-networks of a particular dolphin (JOY) during years 2010 and 2009, respectively[1]. The bottom panel contains the executed query. The table on the left is the resulting relation, in this example consisting of a single tuple. As specified in the query, the tuple's columns contain the ego-networks for each year, and their union, intersection, and difference. The main panel visualizes the value, selected in the result table: in this case, the union of ego-networks between the two years, which represents a graph of dolphins that were observed together with JOY during at least one of the years. By selecting the "intersection" and "difference" columns, the researcher can visualize JOY's repeat friends and new friends, respectively, and discover that the two sets, although disjoint, are approximately the same size (due to space limitations, these results are not shown). Selecting a vertex in columns n1 (node 1 attributes) and n2 (node 2 attributes) changes the display to present the vertex attributes instead of the graph visualization.

By connecting several instances of the Query Result widget to the output of the same Query widget instance, the researcher can manipulate them independently to obtain simultaneous different views of the same query result. For example, it may be helpful to display the ego-network for each year (columns ego1 and ego2) and visually compare them side-by-side using union, intersection, and difference operators.

While we do not have space to discuss our query language or present the related queries in more detail, when executing these queries we made several observations. We can visually observe that dolphins who are most probably males are seen together more often than any of the other combinations of sex. Furthermore, a simple query that calculates the average number of associates for male and female dolphins, confirms that male dolphins are more social on average: 51.2 associates compared to 32.6 for female dolphins. Using one of our similarity operators, we identified potentially similar ego-networks to JOY's ego-network. As expected, ego-networks from dolphins who are observed in the same area as JOY had a higher similarity score, since dolphins are likely to have similar associates if they have the same primary location.

4.2 Citation network use case

The second use case is based on the Cora document citation data set [21]. It contains 2708 nodes, representing machine learning papers, and 5429 edges representing citations to the papers. Each publication is described by a 0/1-valued word vector indicating the absence/presence of

[1]the order is important for the difference operator

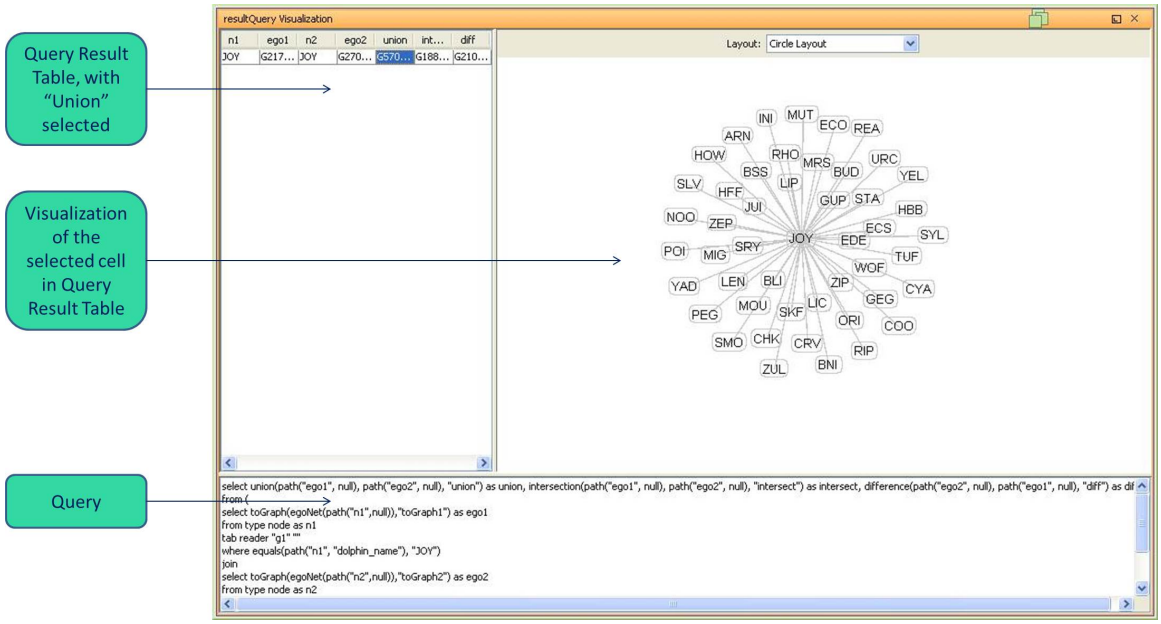


Figure 4: Visualizing query results.

the corresponding word from the dictionary of 1433 unique words. The goal of this use case is to use some or all of the network structure information and the word vectors of the papers to determine the topic of the paper. Each paper can be classified into one of seven possible topics ("Case Based", "Genetic Algorithms", "Neural Networks", "Probabilistic Methods", "Reinforcement Learning", "Rule Learning", "Theory").

More specifically, this use case involves examining and comparing the output of two different node labeling algorithms, which use a partially observed citation data set to predict the probability distribution of the label, i.e. the topic attribute. The workflow in Figure 5a shows two of the several algorithms offered by Invenio-Workflow, where the choice and configuration of the particular algorithm is performed within the Node Label widget. The Majority Rule algorithm simply calculates the label distribution using the labels of the adjacent vertices. The Naïve Bayes classifier, on the other hand, disregards the graph structure and predicts the document topic based on the other attributes, in this case occurrence of words in the paper. We ran the algorithms using a 2-fold training / testing split. Each algorithm partitions the original data into two sets, trains on each set to produce predictions for the other, and outputs a copy of the graph with the predicted probability distribution of the paper topic for each node (paper) in the graph.

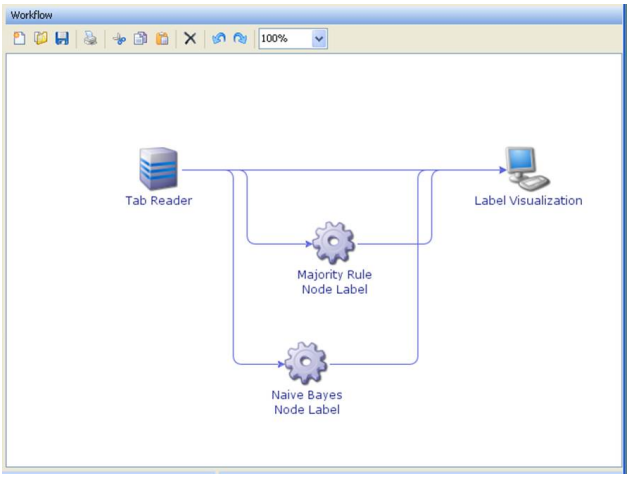
Comparing and contrasting these uncertain graphs to each other or to a ground-truth graph allows researchers to analyze the performance of different node labeling algorithms, experiment with a single algorithm under different assumptions, and examine the graph data set, by highlighting parts of data where algorithms disagree in their predictions or perform poorly. To that end, we created the Node Label Visualization widget. It takes as input the two predicted uncertain graphs, as well as the ground truth graph.

The widget's result view is shown in Figure 6. The basic visual analytic concept is borrowed from the G-Pare visual analytic tool [2] [33]. The statistics pane determines that the simple Majority Rule classifier has better overall accuracy. The table above it provides a side-by-side comparison at vertex granularity. Its columns show the vertex identifier and the most probable label from the ground truth graph and each of the two predicted graphs, respectively. The last three columns show the color-coded histogram representing the probability distribution over all possible labels - again, based on the ground truth and the two node labeling algorithms. The height of the bar corresponds to the probability of the particular value being the actual label. Hovering the mouse over a distribution cell brings up a tooltip with the exact probability for each of the possible label values. Choosing a column highlights the corresponding vertex in the graph view on the right, allowing users to visually identify the node and its neighbors.

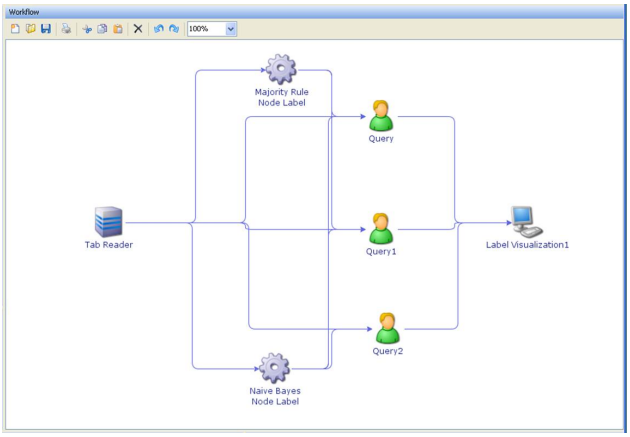
Unlike the detail table, which focuses on examining predictions for each node, the table in the lower left corner provides a higher-level view, showing the areas where the node labeling algorithms agree and disagree in their predictions. As described in [33], the table represents a confusion matrix between the pair of graphs, selected in the combo box, with the option to compare each graph against the ground truth or between each other. Each cell in the table contains the count of nodes with the corresponding combination of most probable label values between the two chosen graphs. Therefore, the cells along the main diagonal represent the vertices, whose most probable labels match between the graphs.

Through heatmap coloring ranging from green through yellow to red, the confusion matrix emphasizes the relative frequency of nodes in each cell. Hovering the mouse over a particular cell brings up the percentage of its node count relative to the total number of nodes in the graph. Furthermore, the confusion matrix supports selection and filtering. Choosing a cell eliminates all the nodes belonging to other

[2]G-Pare is a tool focused on analyzing and comparing machine learning algorithms related to the node labeling task.



(a) Workflow for comparing node labeling algorithm results.



(b) Workflow that includes queries for filtering node labeling algorithm results.

Figure 5: Node labeling workflows.

cells from the detail table. It also highlights the corresponding nodes in the graph view on the right. There is an option to hide the nodes that are not selected in either the confusion matrix or the detail view. For example, the graph view in figure 6 shows the graph subset that includes only "Neural Network" papers misclassified as "Theory" by the Naïve Bayes algorithm.

The graph view, displaying the whole graph or its selected subset, assists the user in visually identifying the differences in predicted labels between the chosen pair of graphs. When the two graphs have the same most probable value, the nodes are colored in different intensity of green through blue, depending on the difference in probability between the graphs. Likewise, the nodes for which the label does not match are colored in shades of yellow through red.

Applying these capabilities to the Cora data set, we can see that for both algorithms, the counts along the main diagonal of the confusion matrix are significantly higher than in the remaining cells, confirming the relatively high accuracy reported in the statistics panel. In particular, "Neural Networks" stands out as the category that occurs by far the most both in the ground truth and in the predicted graphs. It is also the category, in which the Naïve Bayes classifier under performs in comparison with the other categories, in which the counts between the two models are relatively close. In this category, the algorithm misclassified a higher number of papers as either "Probabilistic Methods" or "Theory". Visually examining the detail table and then filtering for nodes misclassified as "Theory", we can conclude that most Naïve Bayes predictions are inaccurate with high probability. The Majority Rule classifier, while being less certain in many cases, is able to suggest the correct topic.

Inserting a query into the node labeling workflow is a flexible way to complement the basic filtering capabilities embedded in the Node Label visualization widget. Such enhanced workflow (Figure 5b) lets the analyst select and concentrate on some areas of particular interest. For example, we can write a query selecting the subgraph that includes only the vertices, for which both models give wrong predictions with high probability (greater than 0.75) and the edges between these vertices.

The Query widget has the capability of selecting a particular cell from the query result relation and sending the selected object downstream to the connected widgets, instead of sending the whole table. Supplying the Node Label visualization with the desired subgraph of papers misclassified with high confidence yields the visualization in Figure 7.

The statistics panel shows the user that the subset under examination is relatively small. With only 89 nodes, it does not provide enough information to draw reliable conclusions about the models on bigger scale, only to make observations and suggest directions for further examination. One of these observations is that, with only 15 edges between the 89 vertices, the papers under consideration are mostly unrelated to each other. There are several exceptions, consisting of 2 or 3 papers, as observed in the graph view. As expected, when the accuracy is 0, there are no entries along the main diagonal, and all nodes in the graph are red, indicating strong label mismatch.

Examining the three confusion matrices leads to several observations (Figure 8). Model 1 misclassified a number of Neural Networks papers as Probabilistic Methods papers and vice versa. This is mostly consistent with model 2 predictions, but in addition, model 2 also assigned Theory label to a number of Neural Networks papers. Furthermore, comparing model 1 vs model 2, the user can see that the majority of entries are along the main diagonal (especially in Theory category), and most nodes in the graph view are colored blue. This leads to the conclusion that whenever both models were wrong with high probability, they made the same predictions, showing the common limitation that both classifiers have or the possible noise in the data set.

Clicking along the main diagonal between the two models and observing the changes in the detail table provides a different perspective. The user may notice that in most cases, there is no obvious correlation between the actual topic and the topic simultaneously chosen by both models (Figure 9). The exception, however, is the case shown in Figure 10, where the majority of papers collectively misclassified as Probabilistic Methods are in fact Neural Networks, and vice versa (not shown). These findings reinforce the previously made observations.

This query represents a single example of examining the results of two node labeling algorithms over a particular subset of interest. Additional queries can be introduced into the

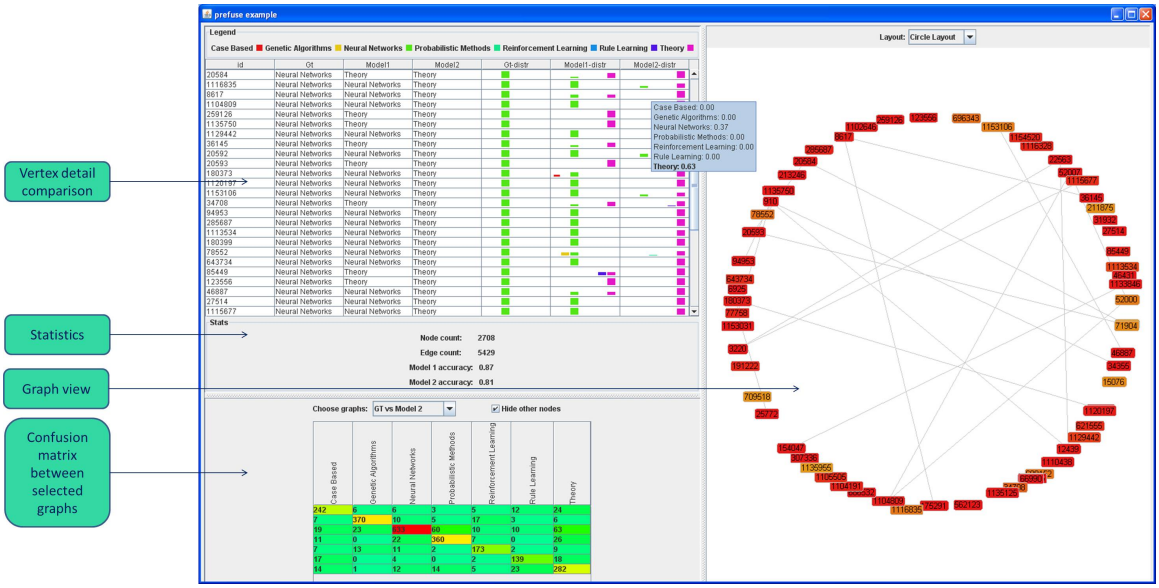


Figure 6: Visualization of node labeling algorithm results.

workflow to further investigate reasons for the wrong predictions. Or, by creating similar queries or modifying these queries in-place, researchers can evaluate other subsets.

5. CONCLUSIONS

In this paper, we have demonstrated how an analyst may use Invenio-Workflow to create workflows with widgets for importing data, executing algorithms, running queries, and interacting with visualizations. Our prototype has focused on three tasks, clustering, node labeling, and graph comparison. We have shown that the widgets associated with these tasks can be synergetically combined in different ways that solve a variety of analytical tasks beyond the capabilities of each single widget. Furthermore, building / executing workflows, interactively analysing their results, and modifying / re-running the workflow in an iterative, ad-hoc manner is a valuable capability for analysts dealing with complex network data, particularly observational scientists. By bringing together elements from areas that include workflow processing, uncertain graph queries, data mining, and graph visualization, we believe that we have created a unique tool with abilities to examine, analyze, and visualize a wide range of graph data from different domains. As future work, we hope to increase the number of data mining and other exploratory tasks supported by the tool, optimize performance for graphs that do not fit into main memory, and develop more sophisticated widgets related to time-evolving networks and information diffusion.

Acknowledgments

This work was supported in part by the National Science Foundation Grant Nbrs. 0941487 and 0937070, and the Office of Naval Research Grant Nbr. 10230702.

6. REFERENCES

[1] Jgraph - open source (bsd) java graph visualization and layout component. http://www.jgraph.com/jgraph.html.

[2] The jide docking framework - a very powerful yet easy-to-use dockable window solution. http://www.jidesoft.com/products/dock.htm.

[3] J. Abello, F. van Ham, and N. Krishnan. ASK-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[4] E. Adar. Guess: a language and interface for graph exploration. In *International Conference on Human Factors in Computing Systems*, 2006.

[5] Amira. Software platform for 3d and 4d data visualization, processing, and analysis. http://www.amira.com/.

[6] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[7] V. Batagelj and A. Mrvar. Pajek-analysis and visualization of large networks. In P. Mutzel, M. Junger, and S. Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*. Springer, 2002.

[8] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. In *IEEE Symposium on Visual Analytics Science and Technology*, 2006.

[9] U. Brandes and D. Wagner. Visone - analysis and visualization of social networks. In *Graph Drawing Software*, 2003.

[10] J. Demšar, B. Zupan, G. Leban, and T. Curk. Orange: from experimental machine learning to interactive data mining. In *European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2004.

[11] D. Dimitrov, L. Singh, and J. Mann. Comparison queries for uncertain graphs. In *(to appear) 24th*

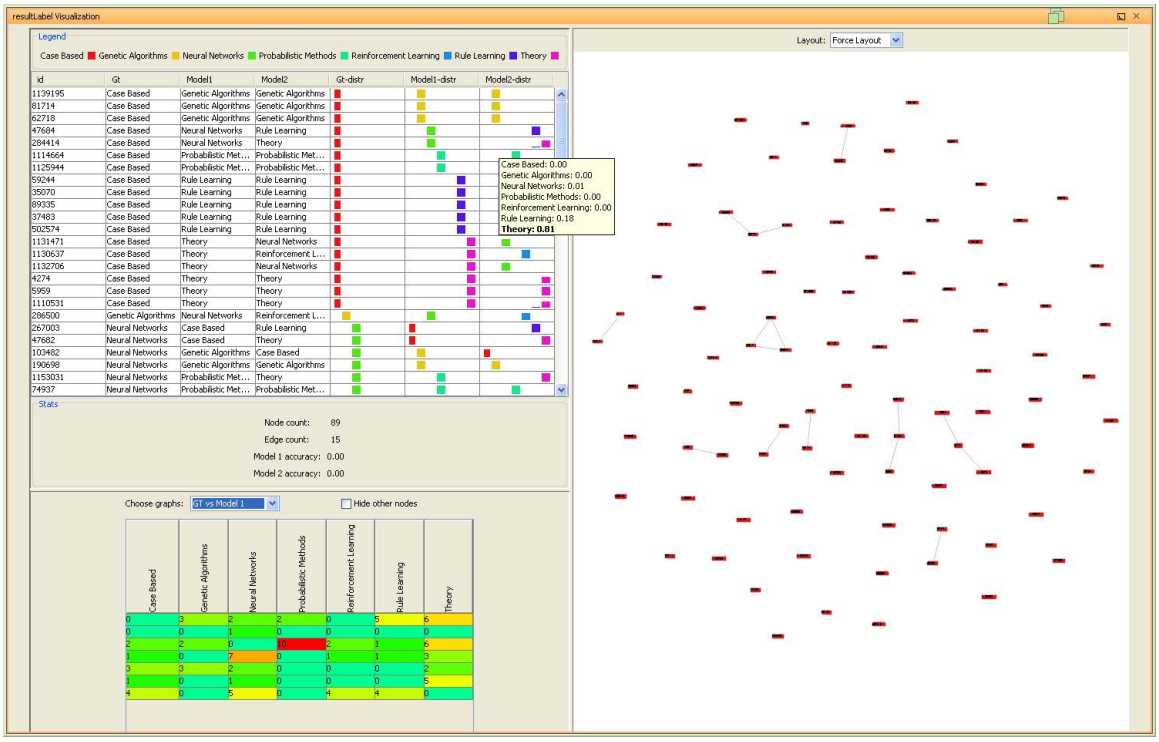


Figure 7: Visualization of query-filtered node labeling algorithm results.

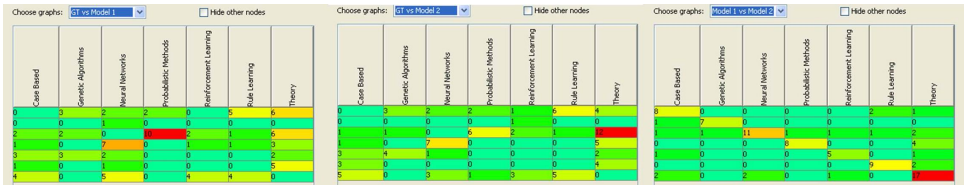


Figure 8: Confusion matrices for query-filtered nodes.

International Conference on Database and Expert Systems Applications, DEXA, 2013.

[12] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. ManyNets: an interface for multiple network analysis and visualization. In *International Conference on Human Factors in Computing Systems*, 2010.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11:10–18, 2009.

[14] D. Hansen, B. Shneiderman, and M. A. Smith. *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*. Morgan Kaufmann Publishers, 2011.

[15] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *International Conference on Human Factors in Computing Systems*, 2005.

[16] N. Henry, J. Fekete, and M. J. McGuffin. Nodetrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13:1302–1309, 2007.

[17] JUNG. Java universal network/graph framework. http://jung.sourceforge.net/.

[18] I. Jusufi, Y. Dingjie, and A. Kerren. The network lens: Interactive exploration of multivariate networks using visual filtering. In *Conference on Information Visualisation*, 2010.

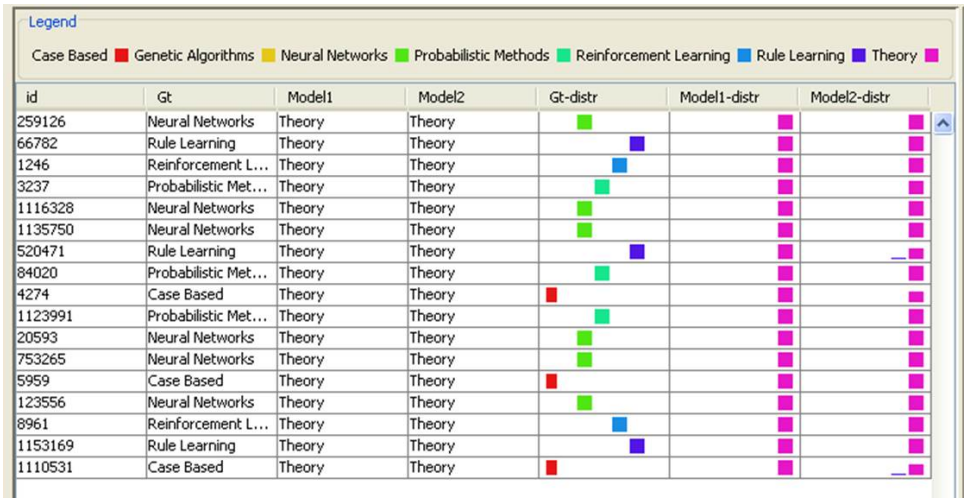
[19] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. In *IEEE Visual Analytics Science & Technology (VAST)*, 2012.

[20] H. Kang, L. Getoor, and L. Singh. C-group: A visual analytic tool for pairwise analysis of dynamic group membership. In *IEEE Symposium on Visual Analytics Science and Technology*, 2007.

[21] LINQS. Machine learning research group @ umd. Available from http://www.cs.umd.edu/ sen/lbc-proj/LBC.html.

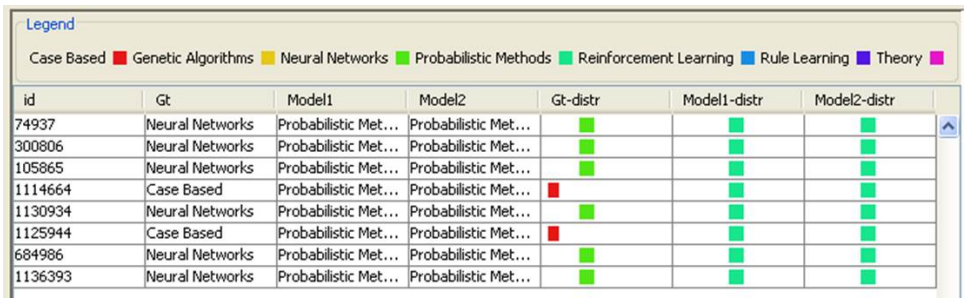
[22] Z. Liu, B. Lee, S. Kandula, and R. Mahajan. Netclinic: Interactive visualization to enhance automated fault diagnosis in enterprise networks. In *IEEE Symposium on Visual Analytics Science and Technology*, 2010.

[23] J. Mann and S. B. R. Team. Shark bay dolphin project. http://www.monkeymiadolphins.org, 2011.



id	Gt	Model1	Model2	Gt-distr	Model1-distr	Model2-distr
259126	Neural Networks	Theory	Theory			
66782	Rule Learning	Theory	Theory			
1246	Reinforcement L...	Theory	Theory			
3237	Probabilistic Met...	Theory	Theory			
1116328	Neural Networks	Theory	Theory			
1135750	Neural Networks	Theory	Theory			
520471	Rule Learning	Theory	Theory			
84020	Probabilistic Met...	Theory	Theory			
4274	Case Based	Theory	Theory			
1123991	Probabilistic Met...	Theory	Theory			
20593	Neural Networks	Theory	Theory			
753265	Neural Networks	Theory	Theory			
5959	Case Based	Theory	Theory			
123556	Neural Networks	Theory	Theory			
8961	Reinforcement L...	Theory	Theory			
1153169	Rule Learning	Theory	Theory			
1110531	Case Based	Theory	Theory			

Figure 9: Detail Table for query-filtered nodes misclassifed as Theory by both algorithms.



id	Gt	Model1	Model2	Gt-distr	Model1-distr	Model2-distr
74937	Neural Networks	Probabilistic Met...	Probabilistic Met...			
300806	Neural Networks	Probabilistic Met...	Probabilistic Met...			
105865	Neural Networks	Probabilistic Met...	Probabilistic Met...			
1114664	Case Based	Probabilistic Met...	Probabilistic Met...			
1130934	Neural Networks	Probabilistic Met...	Probabilistic Met...			
1125944	Case Based	Probabilistic Met...	Probabilistic Met...			
684986	Neural Networks	Probabilistic Met...	Probabilistic Met...			
1136393	Neural Networks	Probabilistic Met...	Probabilistic Met...			

Figure 10: Detail Table for query-filtered nodes misclassifed as Probabilistic Methods by both algorithms.

[24] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, New York, NY, USA, August 2006. ACM.

[25] G. Namata. Gaia (graph alignment, identification and analysis), a software library and tool for analyzing and running machine learning algorithms over graph data. https://github.com/linqs/GAIA.

[26] NetMiner. Netminer - social network analysis software. Available from http://www.netminer.com.

[27] J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y. Boey. Analysis and visualization of network data using jung. *Journal of Statistical Software*, 10:1–35, 2005.

[28] Prefuse. The prefuse visualization toolkit. http://prefuse.org.

[29] R. Software environment for statistical computing and graphics. http://www.r-project.org/.

[30] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

[31] E. Santos, D. Koop, H. T. Vo, E. W. Anderson, J. Freire, and C. Silva. Using workflow medleys to streamline exploratory tasks. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM, 2009.

[32] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[33] H. Sharara, A. Sopan, G. Namata, L. Getoor, and L. Singh. G-pare: A visual analytic tool for comparative analysis of uncertain graphs. In *IEEE VAST*, pages 61–70, 2011.

[34] J. Stasko, C. Gorg, and Z. Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7:118–132, 2008.

[35] Voreen. Volume rendering engine for interactive visualization of volumetric data sets. http://www.voreen.org/.

[36] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

Zips: Mining Compressing Sequential Patterns in Streams

Hoang Thanh Lam
TU Eindhoven
Eindhoven, the Netherlands
t.l.hoang@tue.nl

Toon Calders
TU Eindhoven
Eindhoven, the Netherlands
t.calders@tue.nl

Jie Yang
TU Eindhoven
Eindhoven, the Netherlands
j.yang.1@student.tue.nl

Fabian Mörchen
Amazon.com Inc
Seattle, WA, USA
moerchen@amazon.com

Dmitriy Fradkin
Siemens Corporate Research
Princeton NJ, USA
dmitriy.fradkin@siemens.com

ABSTRACT

We propose a streaming algorithm, based on the minimal description length (MDL) principle, for extracting non-redundant sequential patterns. For static databases, the MDL-based approach that selects patterns based on their capacity to compress data rather than their frequency, was shown to be remarkably effective for extracting meaningful patterns and solving the redundancy issue in frequent itemset and sequence mining. The existing MDL-based algorithms, however, either start from a seed set of frequent patterns, or require multiple passes through the data. As such, the existing approaches scale poorly and are unsuitable for large datasets. Therefore, our main contribution is the proposal of a new, streaming algorithm, called Zips, that does not require a seed set of patterns and requires only one scan over the data. For Zips, we extended the Lempel-Ziv (LZ) compression algorithm in three ways: first, whereas LZ assigns codes uniformly as it builds up its dictionary while scanning the input, Zips assigns codewords according to the usage of the dictionary words; more heaviliy used words get shorter code-lengths. Secondly, Zips exploits also non-consecutive occurences of dictionary words for compression. And, third, the well-known space-saving algorithm is used to evict unpromising words from the dictionary. Experiments on one synthetic and two real-world large-scale datasets show that our approach extracts meaningful compressing patterns with similar quality to the state-of-the-art multi-pass algorithms proposed for static databases of sequences. Moreover, our approach scales linearly with the size of data streams while all the existing algorithms do not.

1. INTRODUCTION

Mining frequent patterns is an important research topic in data mining. It has been shown that frequent pattern mining helps finding interesting association rules, or can be useful for classification and clustering tasks when the extracted patterns are used as features [1]. However, in descriptive data mining the pattern frequency is not a reliable measure. In fact, it is often the case that highly frequent patterns are just a combination of very frequent yet independent items.

There are many approaches that address the aforementioned issues in the literature. One of the most successful approaches is based on data compression which looks for the set of patterns that compresses the data most. The main idea is based on the *Minimum Description Length Principle* (MDL) [5] stating that the best model describing data is the one that together with the description of the model, it compresses the data most. The MDL principle has been successfully applied to solve the redundancy issue in pattern mining and to return meaningful patterns [4, 6].

So far most of the work focussed on mining compressing patterns from static datasets or from modestly-sized data. In practice, however databases are often very large. In some applications, data instances arrive continuously with high speed in a streaming fashion. In both cases, the algorithms must ideally scale linearly with the data size and be able to quickly handle fast data updates. In the streaming case, the main challenge is that whole data cannot be kept in memory and hence the algorithm has to be single-pass. None of the approaches described in the literature scales up to the arbitrarily large data or obey the single-pass constraint.

In this work, we study the problem of mining compressing sequential patterns in a data stream where events arrive in batches, e.g. like stream of tweets. We first introduce a novel encoding that encodes sequences with the help of patterns. Different from the encodings used in recent work [2, 3, 7], the new encoding is online which enables us to design online algorithms for efficiently mining compressing patterns from a data stream. We prove that there is a simple algorithm using the proposed online encoding scheme and achieving a near optimal compression ratio for data streams generated by an independent and identical distributed source, i.e. the same assumption that guarantees the optimality of the *Huffman* encoding in the offline case [9].

Subsequently, we formulate the problem of mining compressing patterns from a data stream. Generally, the data compression problem is NP-complete [11]. Under the streaming context with the additional single pass constraint, we propose a heuristic algorithm to solve the problem. The proposed algorithm scales linearly with the size of data. In the experiments with one synthetic and two real-life large-scale datasets, the proposed algorithm was able to extract mean-

IDEA'13, August 11th, 2013, Chicago, IL, USA.

ingful patterns from the streams while being much more scalable than the-state-of-the-art algorithms.

2. RELATED WORK

The SubDue system [6] is the first work exploiting the MDL principle for mining useful frequent subgraphs. In the field of frequent itemset mining, the well-known Krimp algorithm [4] was shown to be very good at solving the redundancy issue and at finding meaningful patterns.

The MDL principle was first applied for mining compressing patterns in sequence data in [2, 3] and in [7]. The GoKrimp algorithm in the former work solved the redundancy issue effectively. However, the first version of the GoKrimp algorithm [2] used an encoding scheme that does not punish large gaps between events in a pattern. In an extended version of the GoKrimp algorithm [3] this issue is solved by introducing gaps into the encoding scheme based on Elias codes. Besides, a dependency test technique is proposed to filter out meaningless patterns. Meanwhile, in the latter work the SQS algorithm proposed a clever encoding scheme punishing large gaps. In doing so, the SQS was able to solve the redundancy issue effectively. At the same time it was able to return meaningful patterns based solely on the MDL principle.

However, a disadvantage of the encoding defined by the SQS algorithm is that it does not allow encoding of overlapping patterns. Situations where patterns in sequences overlap are common in practice, e.g. message logs produced by different independent components of a machine, network logs through a router etc. Moreover, neither the GoKrimp algorithm nor the SQS algorithm were intended for mining compressing patterns in data streams. The encodings proposed for these algorithms are *offline encodings*. Under the streaming context, an offline encoding does not work because of the following reasons:

1. Complete usage information is not available at the moment of encoding because we don't know the incoming part of the stream

2. When the data size becomes large, the dictionary size usually grows indefinitely beyond the memory limit. Temporally, part of the dictionary must be evicted. In the latter steps, when an evicted word enters the dictionary again we loose the historical usage of the word completely.

3. Handling updates for the offline encoding is expensive. In fact, whenever the usage of the word is updated, all the words in the dictionary must be updated accordingly. On one hand, this operation is expensive, on the other hand, it is impossible to update the compression size correctly for the case that part of the dictionary has been evicted.

In contrast to these approaches, the Zips algorithm proposed in this work inherits the advantages of both state-of-the-art algorithms. It defines a new *online encoding* scheme that allows to encode overlapping patterns. More importantly, under reasonable assumptions, it provably scales linearly with the size of the stream making it the first work in this topic being able to work efficiently on very large datasets. Our work is tightly related to the *Lempel-Ziv*'s data compression algorithm [9]. However, since our goal is to mine interesting patterns instead of compression, the main differences between our algorithm and data compression algorithms are:

1. Data compression algorithms do not aim to a set of patterns because they only focus on data compression.

2. Encodings of data compression algorithms do not consider important patterns with gaps. The *Lempel-Ziv* compression algorithms only exploit repeated strings (consecutive subsequences) to compress the data while in descriptive data mining we are mostly interested in patterns interleaved with noises and other patterns.

3. DATA STREAM ENCODING

In this work, we assume that events in a data stream arrive in batches. This assumption covers broad types of data streams such as tweets, web-access sequences, search engine query logs, etc. This section discusses online encodings that compress a data stream by a set of patterns. For education reasons, we first start with the simplest case when only singletons are used to encode the data. The generalized case with non-singletons is described in the next subsection.

3.1 Online encoding using singletons:

We discuss an online encoding that uses only singletons to compress the data. Since this encoding does not exploit any pattern for compress the data, we consider the representation of the data in this encoding as an uncompressed form of that data. Let $\sum = \{a_1, a_2, \cdots, a_n\}$ be an alphabet containing a set of characters a_i, the online data encoding problem can be formulated as follows:

DEFINITION 1 (ONLINE ENCODING PROBLEM). *Let A be a sender and let B be a receiver. A and B communicate over some network, where sending information is expensive. A observes a data stream $S_t = b_1 b_2 \cdots b_t$. Upon observing a character b_t, A needs to compress the character and transmit it to the receiver B, who must be able to uncompress it. Since sending information on the network is expensive, the goal of A and B is to compress the stream as much as possible to save up the network bandwidth.*

In the offline scenario, i.e. when S_t is finite and given in advance, one possible solution is to first calculate the frequency of every item a (denoted as $f(a)$) of the alphabet in the sequence S_t. Then assign each item a a codeword with length proportional to its entropy, i.e. $-\log f(a)$. It has been shown that when the stream is independent and identically distributed (i.i.d) this encoding, known as the *Huffman* code in the literature, is optimal [9]. However, in the streaming scenario, the frequency of every item a is unknown and the codeword of a must be assigned at the time a arrives and B must know that codeword to decode the compressed item.

We propose a simple solution for Problem 1 as follows. First, in our proposed encoding we need codewords for natural numbers. This work uses the *Elias Delta* code [8] denoted $E(n)$ to encode the number n. The Elias was chosen because it was provable as near optimal when the upper bound on n is unknown in advance. The length of the codeword $E(n)$ is $\lfloor \log_2 n \rfloor + 2\lfloor \log_2 (\lfloor \log_2 n \rfloor + 1) \rfloor + 1$ bits.

A first notifies B of the size of the alphabet by sending $E(|\sum|)$ to B. Then it sends B the dictionary containing all

Figure 1: A first sends B the alphabet $abcd$ then it sends the codewords of gaps between consecutive occurrences of a character. B decodes the gaps and uses them to refer to the characters in part of the stream having been already decoded. The reference stream is $E(3)E(1)E(6)E(5)E(2)E(4)$.

characters of the alphabet $\sum$ in the lexicographical order. Every character in the dictionary is encoded by a binary string with length $\lceil \log_2 |\sum| \rceil$. Finally, when a new character in the stream arrives A sends the codeword of the gap between the current and the most recent occurrence of the character. When B receives the codeword of the gap it decodes the gap and uses that information to refer to the most recent occurrence of the character which has been already decoded in the previous step. Since the given encoding uses a reference to the most recent occurrence of a word to encode its current occurrence we call this encoding the *reference encoding* scheme. We call the sequence of encoded gaps sent by A to B the *reference stream*.

EXAMPLE 1. *Figure 1 shows an example of a reference encoding scheme. A first sends B the alphabet in lexicographical order. When each item of the stream arrives A sends B the codeword of the gap to its most recent occurrence. For instance, A sends $E(3)$ to encode the first occurrence of b and sends $E(1)$ to encode the next occurrence of b. The complete reference stream is $E(3)E(1)E(6)E(5)E(2)E(4)$.*

Let O be a *reference encoding*; denote $L^O(S_t)$ as the length of the data including the length of the alphabet. The average number of bits per character is calculated as $\frac{L^O(S_t)}{t}$. The following theorem shows that when the data stream is generated by an *i.i.d* source, i.e. the same assumption guaranteeing the optimality of the *Huffman* code, the *reference encoding* scheme approximates the optimal solution by a constant factor with probability 1.

THEOREM 1 (NEAR OPTIMALITY). *Given an i.i.d data stream S_t, let $H(P)$ denote the entropy of the distribution of the characters in the stream. If the Elias Delta code is used to encode natural numbers then:*

$$Pr\left(\lim_{t\mapsto\infty} \frac{L^O(S_t)}{t} \leq H(P) + \log_2(H(P)+1) + 1\right) = 1$$

PROOF. Due to space limit the complete proof of this theorem is available in an extension version[1]. □

It has been shown that in expectation the lower bound of the average number of bits per character of any encoding scheme is $H(P)$ [9]. Therefore, a corollary of Theorem 1 is that the *reference encoding* approximates the optimal solution by a constant factor $\alpha = 2$ plus one extra bit.

In the proof of Theorem 1 we can also notice that the gaps between two consecutive occurrences of a character represent the usage of the character in the offline encoding because in expectation the gap is proportional to the entropy of the character, i.e. $-\log p_i$. This property is very important because it provides us with a lot of conveniences in designing an effective algorithm to find compressing patterns in a data stream. In particular, since gaps can be calculated instantly without the knowledge about the whole data stream, using reference encoding we can solve all the aforementioned issues of the offline encodings.

[1] `http://www.win.tue.nl/~lamthuy/projects/zips.pdf`

3.2 Online encoding with non-singletons

The *reference encoding* can be extended to the case using singleton together with non-singleton patterns to encode a data stream. Let $\mathfrak{S} = S_1 S_2 \cdots S_t$ denote a stream of sequences where each S_i is a sequence of events. Let D be a dictionary containing all characters of the alphabet and some non-singletons. *Reference encodings* compress a data stream $\mathfrak{S}$ by replacing instances of words in the dictionary by references to the most recent occurrences of the words. If the words are non-singletons, beside the references, gaps between characters of the encoded words must be stored together with the references. Therefore, in a reference encoding, beside the reference stream we also have a *gap stream*.

Similar to the case with non-singleton, first we need to encode the dictionary D (now contains both singleton and non-singleton). We add a special symbol $\sharp$ to the alphabet. The binary representation of the dictionary starts with the codeword of the size of the dictionary. It is followed by the codewords of all the characters in the alphabet each with length $\lceil \log_2 |D| \rceil$. The representations of every non-singleton follow right after that. The binary representation of a non-singleton contains codewords of the characters of the non-singleton. Non-singletons are separated from each other by the special character $\sharp$.

EXAMPLE 2 (DICTIONARY REPRESENTATION). *The dictionary $D = \{a, b, c, \sharp, ab, abc\}$ can be represented as follows $E(6)C(a)C(b)C(c)C(\sharp)C(a)C(b)C(\sharp)C(a)C(b)C(c)$. The representation starts with $E(6)$ indicating the size of D. It follows by the codewords of all the characters and the binary representation of the non-singletons separated by $\sharp$.*

Having the binary representation of the dictionary, A first sends that representation to B. After that A send the encoding of the actual stream with the reference stream and the gap stream. The following example show how to encode a sequence with a reference encoding.

EXAMPLE 3 (REFERENCE ENCODING). *Given a dictionary $D = \{a, b, c, \sharp, ab, abc\}$, a sequence $S = abbcacbacbacbabc$. Figure 2 show an encoding of S using dictionary words(the numbers below the characters denote the positions of the character in the original sequence). The reference stream is $E(1)E(7)E(5)E(8)E(2)E(2)E(2)E(2)E(8)$, where $E(1)$ is the reference of the first abc to the position of abc in the*

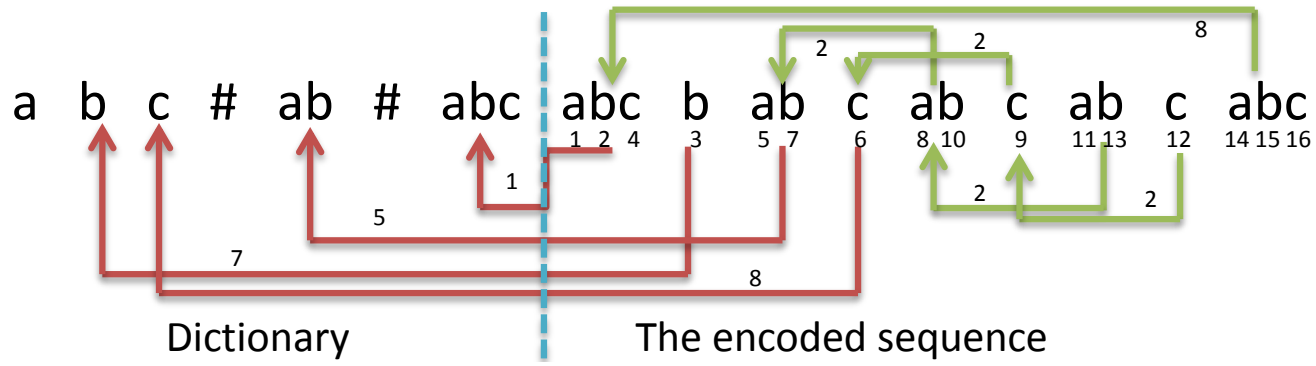


Figure 2: An example of encoding the sequence $S = abbcacbacbacbabc$ with a reference encoding. The arrows represent the references between two consecutive encoded occurrences of a word which represent as a reference stream $E(1)E(7)E(5)E(8)E(2)E(2)E(2)E(2)E(8)$. Having the reference stream we can reconstruct the stream but not in the same order of event occurrence. Therefore, we need a gap stream indicating the gaps between consecutive characters in each encoded non-singletons. For example, the gap stream is $E(1)E(2)E(2)E(2)E(2)E(1)E(1)$.

dictionary, $E(7)$ is the reference of the following b to the position of b in the dictionary and so on. The gap stream is $E(1)E(2)E(2)E(2)E(2)E(1)E(1)$, where for instance, the first codewords $E(1)E(2)$ indicate the gaps between a and b, b and c in the first occurrence of abc at position $(1, 2, 4)$ in the stream. For non-singleton, there is no gap information representing in the gap stream.

EXAMPLE 4 (DECODING). *In Figure 2, the sequence can be decoded as follows. Reading the first codeword of the reference stream, i.e. $E(1)$, the decoder refers one step back to get abc. There will be two gaps (between a, b and b, c) so the decoder reads the next two codewords from the gap stream, i.e. $E(1)$, and $E(2)$. Knowing the gaps it can infer the positions of abc. In this case, the positions are 1,2 and 4. Subsequently, the decoder reads the next codeword from the reference stream, i.e. $E(7)$, it refers seven steps back and decode the current reference as b. There is no gap because the word is a singleton, the position of b in the stream corresponds to the earliest position that has not been occupied by any decoded character, i.e. 3. The decoder continues decode the other references of the stream in the same way.*

Different from the singleton case, there might be a lot of different reference encodings for a stream given a dictionary. Each reference encoding incurs different description lengths. Finding an optimal dictionary and an optimal reference encoding is the main problem we solve in this paper.

4. PROBLEM DEFINITION

Given a data stream $\mathfrak{S}$ and a dictionary D denote $L_D^C(\mathfrak{S})$ as the description length of the data (including the cost to store the dictionary) in the encoding C. The problem of mining compressing sequential patterns in data stream can be formulated as follows:

DEFINITION 2 (COMPRESSING PATTERNS MINING). *Given a stream of sequences $\mathfrak{S}$, find a dictionary D and an encoding C such that $L_D^C(\mathfrak{S})$ is minimized.*

Generally, the problem of finding the optimal lossless compressed form of a sequence is NP-complete [11]. In this work,

Algorithm 1 Zips(S)

```
1: Input: Event stream 𝔖 = S_1 S_2 ···
2: Output: Dictionary D
3: D ⟵ ∅
4: for t = 1 to ∞ do
5:    while S_t ≠ ϵ do
6:       w = encode(S_t)
7:       w* = extend(w)
8:       update(w*)
9:    end while
10: end for
11: Return D
```

Problem 2 is similar to the data compression problem but with additional constraint on the number of passes through data. Therefore, in next section we discuss a heuristic algorithm inspired by the idea of the *Lempel-Ziv*'s data compression algorithm [9] to solve this problem.

5. ALGORITHMS

In this section, we discuss an algorithm for finding a good set of compressing patterns from a data stream. Our algorithm is single-pass, memory-efficient and scalable. We call our algorithm Zips as for *Zip a stream*. There are three important subproblems that Zips will solve. The first problem concerns how to grow a dictionary of promising candidate patterns for encoding the stream. Since the memory is limited, the second problem is how to keep a small set of important candidates and evict from the dictionary unpromising candidates. The last problem is that having a dictionary how to encode the next sequence effectively with existing words in the dictionary.

The pseudo-code depicted in Algorithm 1 shows how Zips work. It has three subroutines each of them solves one of the three subproblems:

1. Compress a sequence given a dictionary: for every new sequence S_t in the stream, Zips uses the subroutine $encode(S_t)$ to find the word w in the dictionary which gives the most compression benefit when it is used to

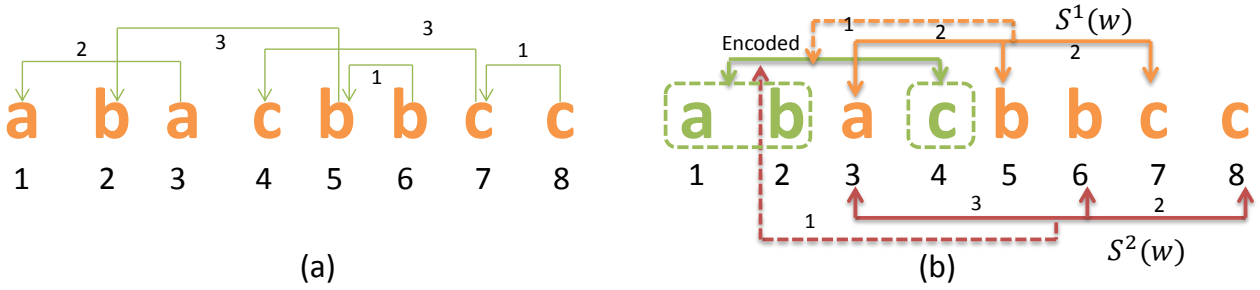


Figure 3: An illustration of how compression benefit is calculated: (a) The sequence S in the uncompressed form. (b) two instances of $w = abc$: $S^1(w)$ and $S^2(w)$ and their references to the most recent encoded instance of w highlighted by the green color.

encode the uncompressed part of the sequence. Subsequently, the instance corresponds to the best chosen word is removed from S_t. Detail about how to choose w is discussed in subsection 5.1.

2. Grow a dictionary: Zips uses $extend(w)$ to extend the word w chosen by the procedure $encode(S_t)$. Word extensions are discussed in subsection 5.2.

3. Dictionary update: the new extension is added to the dictionary. When the dictionary size exceeds the memory limit, a space-saving algorithm is used to evict unpromising words from the dictionary (subsection 5.3).

These steps are iteratively repeated as long as S_t is not encoded completely. When compression of the sequence finishes, Zips continues to compress the next in a similar way.

5.1 Compress a sequence:

Let S be a sequence, consider a dictionary word $w = a_1a_2\cdots a_k$, let $S(w)$ denote an instance of w in S. Let $g_2, g_3, \cdots, g_k$ be the gaps between the consecutive characters in $S(w)$. We denote the gap between the current occurrence and the most recent encoded occurrence of w by g. Let $\overline{g_i}$ $i = 1, \cdots, k$ be the gap between the current and the most recent occurrence of a_i. Therefore, to calculate the compression benefit we subtract the size of encoding $S(w)$ and the cost of encoding the gap to the last encoded occurrence of $S(w)$ from the size of encoding each singleton:

$$B(S(w)) = \sum_{i=1}^{k} |E(\overline{g_i})| - |E(g)| - \sum_{i=2}^{k} |E(g_i)| \qquad (1)$$

Example 5 (Compression benefit). *Figure 3.a shows a sequence S in the uncompressed form and Figure 3.b shows the current form of S. Assume that the instance of $w = abc$ at positions $1, 2, 4$ is already compressed. Consider two instances of abc in the uncompressed part of S:*

1. *$S^1(w) = (a,3)(b,5)(c,7)$: the cost to replace this instance by a pointer is equal to the sum of the cost to encode the reference to the previous encoded instance of abc $|E(1)|$ plus the cost of gaps $|E(2)|+|E(2)|$. The cost of representing this instance in an uncompressed form is $|E(2)| + |E(3)| + |E(3)|$. Therefore the compression benefit of using this instance to encode the sequence is $B(S^1(w)) = |E(2)|+|E(3)|+|E(3)|-|E(1)|-|E(2)|-|E(2)| = 3$ bits.*

2. *$S^2(w) = (a,3)(b,6)(c,8)$: the compression benefit of using $S^2(w)$ to encode the sequence is calculated in a similar way: $B(S^2(w)) = |E(2)| + |E(1)| + |E(1)| - |E(1)| - |E(3)| - |E(2)| = -3$ bits.*

In order to ensure that every symbol of the sequence is encoded, the next instance considered for encoding has to start at the first non-encoded symbol of the sequence. There maybe many instances of w in S that start with the first non-encoded symbol of S, denote $S^*(w) = argmax_{S(w)}B(S(w))$ as the one that results in the maximum compression benefit. We call $S^*(w)$ the *best match* of w in S. Given a dictionary, the encoding function depicted in Algorithm 2 first goes through the dictionary and finds the best match starting at the next uncompressed character of every dictionary word in the sequence S (line 4). Among all the best matches, it greedily chooses the one that results in the maximum compression benefit (line 6).

For any given dictionary word $w = a_1a_2\cdots a_k$, the most important subroutine of Algorithm 2 is to find the best match $S^*(w)$. This problem can be solved by creating a directed acyclic graph $G(V, E)$ as follows:

1. Initially, V contains a start node s and an end node e

2. For every occurrence of a_i at position p in S, add a vertex (a_i, p) to V

3. Connect s with the node (a_1, p) by a directed edge and add to that edge a weight value equal to $|E(\overline{g}_1)| - |E(g)|$.

4. Connect every vertex (a_k, p) with e by a directed edge with weight 0.

5. For all $q > p$ connect (a_i, p) to (a_{i+1}, q) by a directed edge with weight value $|E(\overline{g}_{i+1})| - |E(q-p)|$

Algorithm 2 encode(S)

1: **Input**: a sequence S and dictionary $D = w_1w_2\cdots w_N$
2: **Output**: the word w that starts at the first non-encoded symbol gives the most additional compression benefit
3: **for** $i = 1$ **to** N **do**
4: $S^*(w_i) = bestmatch(S, w_i)$
5: **end for**
6: $max = argmax_i B(S^*(w_i))$
7: Return w_{max}

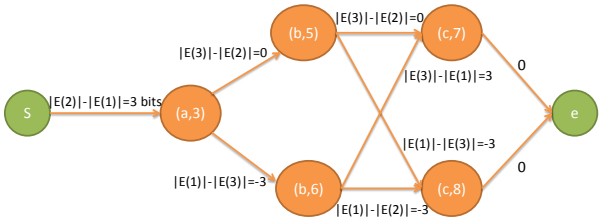


Figure 4: A directed acyclic graph created from the instances of *abc* in the uncompressed part of S shown in Figure 3.b

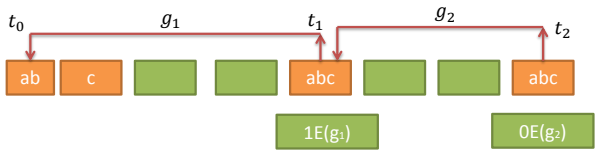


Figure 5: An example of word is extended and encoded the first time and the second time. One extra flag bit is added to every reference to differentiate two cases.

THEOREM 2. *The best match $S^*(w)$ corresponds to the directed path from s to e with the maximum sum of the weight values along the path.*

The proof of theorem 2 is trivial since any instance of w in S corresponds to a directed path in the directed acyclic graph and vice versa. The sum of the weights along a directed path is equal to the benefit of using the corresponding instance of w to encode the sequence. Finding the directed path with maximum weight sum in a directed graph is a well-known problem in graph theory. That problem can be solved by a simple dynamic programming algorithm in linear time of the size of the graph, i.e. $O(|S|^2)$[12].

EXAMPLE 6 (THE BEST MATCH IN A GRAPH). *Figure 4 shows the directed acyclic graph created from the instances of abc in the uncompressed part of S shown in Figure 3.b. The best match of abc corresponds to the path $s(a,3)(b,5)(c,7)e$ with the maximum sum of weights equal to 3 bits.*

It is important to notice that in order to evaluate the compression benefit of a dictionary word, Equation 1 only requires bookkeeping the position of the most recent encoded instance of the word. This is in contrast to the offline encodings used in recent work [2, 3, 7] in which the bookkeeping of the word usage and the gaps cost is a must. When a new instance of a word is replaced by a pointer, the relative usage and the codewords of all dictionary words also change. As a result, the compression benefit needs to be recalculated by a pass through the dictionary. This operation is an expensive task when the dictionary size is unbounded.

5.2 Dictionary extension:

Initially, the dictionary contains all singletons; it is iteratively expanded with the locally best words. In each step, when the best match of a dictionary word has been found, Zips extends the best match with one extra character and adds this extension to the dictionary. There are different options to choose the character for extension. In this work, Zips chooses the next uncompressed character right after the word. The choice is inspired by the same extension method suggested by the *Lempel-Ziv* compression algorithms.

Moreover, there is another reason behind our choice. When w is encoded for the first time, the reference to the previous encoded instance of w is undefined although the word has been already added to the dictionary. Under that circumstance, we have to differentiate between two cases: either a reference to an extension or to an encoded word. To achieve this goal one extra flag bit is added to every reference. When the flag bit is equal to 1, the reference refers to an extension of an encoded word. Otherwise, it refers to an encoded word. When the former case happens by extending the word with the character right after it, the decoder always knows where to find the last character of the extension. When a word is added to a dictionary all of its prefixes have been already added to the dictionary. This property enables us to store the dictionary by using a prefix tree.

EXAMPLE 7. *Figure 5 shows the first moment t_0 when the word $w = abc$ is added to the dictionary and two other moments t_1 and t_2 when it is used to encode the stream. At t_1, the flag bit 1 is used to notify the decoder that the gap g_1 is a reference to an extension of an encoded word, while at t_2 the decoder understands that the gap g_2 is a reference to the previous encoded instance of w.*

References to extensions may cause ambiguous references. For instance, a potential case of ambiguous reference called *reference loop* is discussed in the following example:

EXAMPLE 8 (REFERENCE LOOPS). *At time point t_0 the word $w = ab$ at positions p_1p_2 is extended by c at position p_3. Later on at another time point $t_1 > t_0$, another instance of abc at $q_1q_2q_3$ ($q_1 > p_1$) refers back to the instance abc at $p_1p_2p_3$. At time point $t_2 > t_1$, another instance of abc at $r_1r_2p_3$ ($r_1 > q_1$) refers back to the instance abc at $q_1q_2q_3$. In this case, c at p_3 and c at q_3 refers to each other forming a reference loop.*

Reference loops result in ambiguity when we decode the sequence. Therefore, when we look for the next best matches, in order to avoid reference loops, we always check if the new match incurs a loop. The match that incurs a loop is not considered for encoding the sequence. Checks for ambiguity can be done efficiently by creating paths of references. Vertices of a reference path are events in the sequence. Edge between two consecutive vertices of the path corresponds to a reference between the associated events. Since the total sizes of all the paths is at most the sequence length, its is cheap to store the paths for a bounded size sequence. Moreover, updating and checking if a path is a loop can be done in $O(1)$ if vertices of the path are stored in a hashmap.

EXAMPLE 9 (REFERENCE PATHS). *The references paths of the encoding in Example 8 are: $(a,r_1) \mapsto (a,q_1) \mapsto (a,p_1)$, $(b,r_2) \mapsto (b,q_2) \mapsto (b,p_2)$ and $(c,p_3) \mapsto (c,q_3) \mapsto (c,p_3)$. The last path is a loop.*

5.3 Dictionary update:

New extensions are added to the dictionary until the dictionary exceeds memory limit. When it happens the *space-saving* algorithm is used to evict unpromising words from the dictionary. The space-saving algorithm [10] is a well-known

Algorithm 3 update(w^*)

1: **Input**: a word w^* and dictionary $D = \{w_1, w_2, \cdots, w_N\}$
2: **Output**: the dictionary D
3: $m \leftarrow |i : w_i$ is a non-singleton$|$
4: $v = argmin_i w_i[1]$ and v is non-singleton at a leaf of the prefix-tree
5: **if** $m > M$ and $w^* \notin D$ **then**
6: $D = D \setminus \{v\}$
7: $w^*[1] = w^*[2] = v[1]$
8: $D = D \bigcup \{w^*\}$
9: **else if** $w^* \notin D$ **then**
10: $w^*[1] = w^*[2] = 0$
11: $D = D \bigcup \{w^*\}$
12: **else**
13: add additional compression benefit to $w^*[1]$
14: **end if**
15: Return D

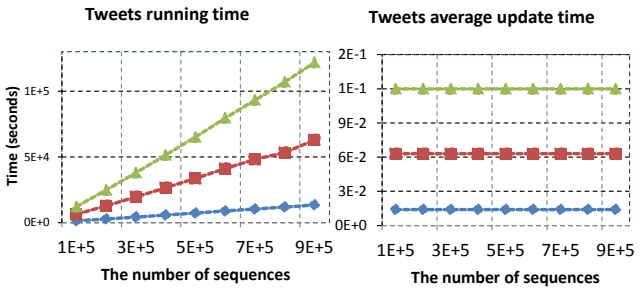


Figure 6: The running time and the average update time per sequence of the Zips algorithm in the Tweets dataset when the stream size increases. Zips scales linearly with the size of the stream.

method proposed for finding the most frequent items in a stream of items given a budget on the maximum memory. In this work, we propose a similar space-saving algorithm to keep the number of non-singleton words in the dictionary at below a predefined number M while it can be able to return the set of compressing patterns with high accuracy.

The algorithm works as follows, for every non-singleton word w it maintains a counter with two fields. The first field denoted as $w[1]$ contains an over-estimate of the compression benefit of w. The second field denoted as $w[2]$ contains the compression benefit of the word with least compression benefit in the dictionary at the moment that w is inserted into the dictionary.

Every time when a word w is chosen by Algorithm 2 to encode its best match in the sequence S_t, the compression benefit of the word is updated. The word w is then extended to w^* with an extra character by the extension subroutine. In its turn, Algorithm 3 checks if the dictionary already contains w^*. If the dictionary does not contains w^* and it is full with M non-singleton words, the least compressing word v resident at a leaf of the dictionary prefix-tree is removed from the tree. Subsequently, the word w^* is inserted into the tree and its compression benefit can be over-estimated as $w[1] = w[2] = v[1]$. The first counter of every word is always greater than the true compression benefit of the

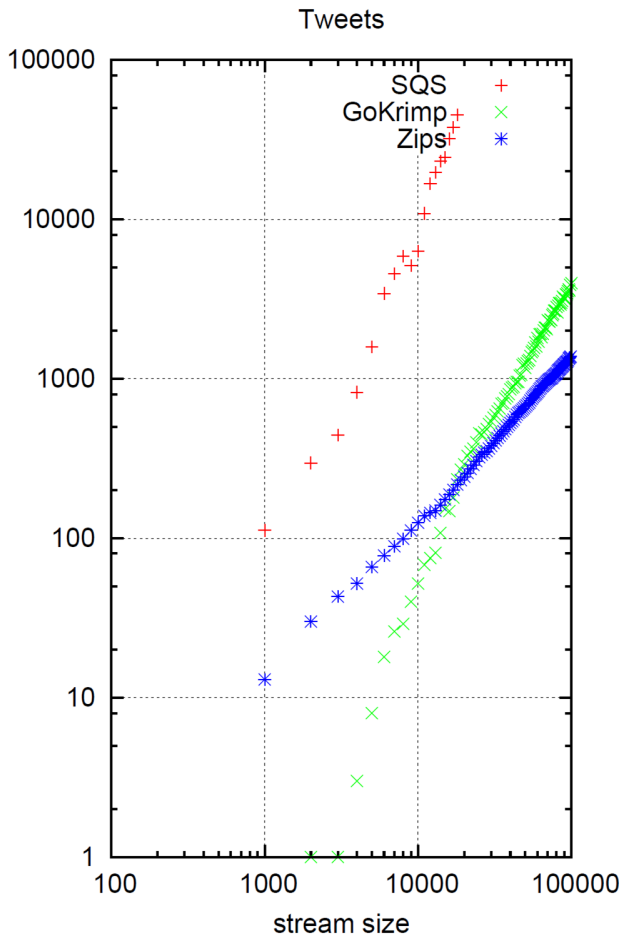


Figure 7: Running time (x-axis) against the data size (y-axis) of three algorithms in log-log scales. The Zips algorithm scales linearly with the data size while the GoKrimp and the SQS algorithm scales quadratically with the data size.

words. This property ensures that the new emerging word is not removed very quickly because its accumulated compression benefit is dominated by long lasting words in the dictionary. For any word w, the difference between $w[2]$ and $w[1]$ is the actual compression benefit of w since the moment that w is inserted into the dictionary. At anytime point when we need to find the most compressing patterns, we compare the value of $w[2] - w[1]$ and select those with highest $w[2] - w[1]$. In section 6 we show empirical results with different datasets that this algorithm is very effective in finding the most compressing patterns with high accuracy even with limited memory.

6. EXPERIMENTS

We perform experiments with one synthetic dataset with known ground truth and two real-world datasets. Our implementation of the Zips algorithm in C++ together with the datasets are available for download at our project website[2]. All the experiments were carried out on a machine

[2] `www.win.tue.nl/~lamthuy/zips.html`

with 16 processor cores, 2 Ghz, 12 GB memory, 194 GB local disk, Fedora 14 / 64-bit. As baseline algorithms, we choose the GoKrimp algorithm [2, 3] and the SQS algorithm [7] for comparison in terms of running time, scalability, and interpretability of the set of patterns.

6.1 Data

Three different datasets are:

1. **JMLR**: contains 787 abstracts of articles in the Journal of Machine Learning Research. English words are stemmed and stop words are removed. JMLR is small but it is considered as a benchmark dataset in the recent work [2, 3, 7]. The dataset is chosen also because the set of extracted patterns can be easily interpreted.

2. **Tweets**: contains over 1270000 tweets from 1250 different twitter accounts[3]. All tweets are ordered ascending by timestamp, English words are stemmed and stop words are removed. After preprocessing, the dataset contains over 900000 tweets. Similar to the JMLR dataset, this dataset is chosen because the set of extracted patterns can be easily interpreted.

3. **Plant**: is a synthetic dataset generated in the same way as the generation of the plant10 and plant50 dataset used in [7]. The plant10 and plant50 are small so we generate a larger one with ten patterns each with 5 events occurs 100 times at random positions in a sequence with length 100000 generated by 1000 independent noise event types.

6.2 Running time and Scalability

Figure 6 plots the running time and the average update time per sequence of the Zips algorithm in the Tweets dataset when the data stream size (the number of sequences) increases. Three different lines correspond to different maximum dictionary size settings $M = 1000$, $M = 5000$ and $M = 10000$ respectively. The results show that the Zips algorithm scales linearly with the size of the stream. The average update time per sequence is constant given a maximum dictionary size setting. For example, when $M = 10000$, Zips can handle one update in about 20-100 milliseconds.

Figure 7 shows the running time in y-axis of the Zips algorithm against the stream size in x-axis in the Tweets dataset when the maximum dictionary size is set to 1000. In the same figure, the running time of the baseline algorithms GoKrimp and SQS are also shown. There are some missing points in the results corresponding to the SQS algorithm because we set a deadline of ten hours to get the results corresponding to a point. The missing points corresponding to the cases when the SQS program did not finish in time.

In the log-log scale, running time lines resemble straight lines. This result shows that the running time of Zips, GoKrimp and SQS is the power of data size, i.e. $T \sim \alpha |S|^{\beta}$. Using linear fitting functions in log-log scale we found that with the Zips algorithm $\beta = 1.01$, i.e. Zips scales linearly with the data size. Meanwhile, for the SQS algorithm the exponent is $\beta = 2.2$ and for the GoKrimp algorithm the exponent is $\beta = 2.01$. Therefore, both GoKrimp and SQS do not scale linearly with the data size and hence they are not suitable for data stream applications.

[3] `http://user.informatik.uni-goettingen.de/~txu/cuckoo/dataset.html`

6.3 JMLR

In Figure 8, we show the first 20 patterns extracted by two baseline algorithms GoKrimp and SQS and the Zips algorithm from the JMLR dataset. Three lists are slightly different but the important patterns such as "support vector machine", "data set", "machine learn", "bayesian network" or "state art" were discovered by all of the three algorithms. This experiment confirms that the Zips algorithm was able to find important patterns that are consistent with the results of state-of-the-art algorithms.

6.4 Tweets

Since the tweet dataset is large, we schedule the programs so that they terminate their running after two weeks. The SQS algorithm was not able to finish its running before the deadline while GoKrimp finished running after three days and Zips finished running after 35 hours. The set of patterns extracted by the Zips algorithm and the GoKrimp algorithm are shown in Figure 9. Patterns are visualized by the wordcloud tool in R such that more important patterns are represented as larger words. In both algorithms, the sets of patterns are very similar. The result shows the daily discussions of the 1250 twitter accounts about the topics regarding "social media", "Blog post', about "youtube video", about "iphone apps", about greetings such as "happy birthday", "good morning" and "good night", about custom service complaint etc.

6.5 Plants

It has been shown [7] that SQS successfully returned all ten patterns. We obtain the same result with GoKrimp and Zips. All three algorithm ranked 10 true patterns with highest scores.

7. CONCLUSIONS AND FUTURE WORK

In this paper we studied the problem of mining compressing patterns from a data stream. A new encoding scheme for sequence is proposed. The new encoding is convenient for streaming applications because it allows encoding the data in an online manner. Because the problem of mining the best set of patterns with respect to the given encoding is shown to be unsolvable under the streaming context, we propose a heuristic solution that solves the mining compressing problem effectively. In the experiments with one synthetic dataset with known ground-truths Zips was able to extract the most compressing patterns with high accuracy. Meanwhile, in the experiments with two real-world datasets it can find patterns that are similar to the-state-of-the-art algorithms extract from these datasets. More importantly, the proposed algorithm was able to scale linearly with the size of the stream while the-state-of-the-art algorithms were not. There are several options to extend the current work. One of the most promising future work is to study the problem of mining compressing patterns for different kinds of data stream such as a stream of graphs.

8. REFERENCES

[1] Hong Cheng, Xifeng Yan, Jiawei Han, Philip S. Yu: Direct Discriminative Pattern Mining for Effective Classification. ICDE 2008: 169-178

[2] Hoang Thanh Lam, Fabian Moerchen, Dmitriy Fradkin, Toon Calders: Mining Compressing Sequential Patterns. SDM 2012: 319-330

Method	Patterns			
SQS	**support vector machin**	**larg scale**	featur select	sampl size
	machin learn	nearest neighbor	graphic model	learn algorithm
	state art	decis tree	**real world**	princip compon analysi
	data set	**neural network**	**high dimension**	logist regress
	bayesian network	cross valid	mutual inform	model select
GOKRIMP	**support vector machin**	**state art**	**neural network**	well known
	real world	**high dimension**	experiment result	special case
	machin learn	reproduc hilbert space	sampl size	solv problem
	data set	**larg scale**	supervis learn	signific improv
	bayesian network	independ compon analysi	support vector	object function
Zips	**support vector machin**	featur select	**high dimension**	cross valid
	data set	**machine learn**	paper propose	decis tree
	real world	**bayesian network**	graphic model	**neutral network**
	learn algorithm	model select	**larg scale**	well known
	state art	optim problem	result show	hilbert space

Figure 8: The first 20 patterns extracted from the JMLR dataset by two baseline algorithms GoKrimp and SQS and the Zips algorithm. Common patterns discovered by all the three algorithms are bold.

Figure 9: The top 20 most compressing patterns extracted by Zips (left) and by GoKrimp (right) from the Tweets dataset.

[3] Hoang Thanh Lam, Fabian Moerchen, Dmitriy Fradkin, Toon Calders: Mining Compressing Sequential Patterns. Accepted for publish in Statistical Analysis and Data Mining, A Journal of American Statistical Association, Wiley.

[4] Jilles Vreeken, Matthijs van Leeuwen, Arno Siebes: Krimp: mining itemsets that compress. Data Min. Knowl. Discov. 23(1): 169-214 (2011)

[5] Peter D. Grünwald The Minimum Description Length Principle MIT Press 2007

[6] L. B. Holder, D. J. Cook and S. Djoko. Substructure Discovery in the SUBDUE System. In Proceedings of the AAAI Workhop on Knowledge Discovery in Databases, pages 169-180, 1994.

[7] Nikolaj Tatti, Jilles Vreeken: The long and the short of it: summarising event sequences with serial episodes. KDD 2012: 462-470

[8] Ian H. Witten, Alistair Moffat and Timothy C. Bell Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition. The Morgan Kaufmann Series in Multimedia Information and Systems. 1999

[9] Thomas M. Cover and Joy A. Thomas. Elements of information theory. Second edition. Wiley Chapter 13.

[10] Ahmed Metwally, Divyakant Agrawal, Amr El Abbadi: Efficient Computation of Frequent and Top-k Elements in Data Streams. ICDT 2005: 398-412

[11] James A. Storer. Data compression via textual substitution Journal of the ACM (JACM) 1982

[12] Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill

Augmenting MATLAB with Semantic Objects for an Interactive Visual Environment

Changhyun Lee
Georgia Institute of Technology
Atlanta, GA 30332, USA
clee407@gatech.edu

Jaegul Choo
Georgia Institute of Technology
Atlanta, GA 30332, USA
jaegul.choo@cc.gatech.edu

Duen Horng (Polo) Chau
Georgia Institute of Technology
Atlanta, GA 30332, USA
polo@gatech.edu

Haesun Park
Georgia Institute of Technology
Atlanta, GA 30332, USA
hpark@cc.gatech.edu

ABSTRACT

Analysis tools such as Matlab, R, and SAS support a myriad of built-in computational functions and various standard visualization techniques. However, most of them provide little interaction from visualizations mainly due to the fact that the tools treat the data as just numerical vectors or matrices while ignoring any semantic meaning associated with them. To solve this limitation, we augment Matlab, one of the widely used data analysis tools, with the capability of directly handling the underlying semantic objects and their meanings. Such capabilities allow users to flexibly assign essential interaction capabilities, such as brushing-and-linking and details-on-demand interactions, to visualizations. To demonstrate the capabilities, two usage scenarios in document and graph analysis domains are presented.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: User Interfaces

General Terms

visual analytics, interactive visualization, clustering, dimension reduction

1. INTRODUCTION

The various data analysis tools and computing languages are available such as Matlab[1], R[2], and SAS[3]. These tools are often geared towards handling generic, typically numerical, data types, and thus they are easily applied in a wide variety of domains as long as the data can be represented as numerical values. These data analysis tools are generally composed of two main capabilities: *computation* and *presentation*. By the former, we mean the capabilities of applying various data processing and transformation techniques to input data while by the latter, we mean those of conveying the input data as well as computed output data to users mainly via *interactive visualization*.

From the perspective of computational capabilities, data analysis tools often support a myriad of built-in computational functions and flexible ways to create users' own ones. It comes to interactive visualization capabilities, even though various standard visualization techniques such as a bar graph, a line graph, a scatter plot, and a heatmap, are supported, little interactions with them are usually allowed. That is, even simple but essential interaction techniques such as brushing-and-linking, which highlights two or more interconnected data items between different visualizations and details-on-demand interactions, which let users explore data details, are hardly provided. Although some tools provide brushing-and-linking, the way of connection between objects is limited and not informative.

The reason for such a problem is mainly due to the fact that these data analysis tools do not inherently assume any semantic meanings associated with the data for the sake of maintaining a wide applicability to different domains. That is, even though the data comes from particular real-world applications such as image, document, gene expression, etc., data analysis tools treat the data in the same way once they are represented as numerical vectors or matrices. As a result, it becomes nontrivial for the data analysis tools to natively support the above-mentioned interactions, which would make sense only when the meanings of data are present in visualizations.

For example, in the context of document analysis, suppose scatter plot is given showing an overall relationship between data points where a data point represent a document. Data analysis tools may maintain only their horizontal and vertical coordinates to draw data points in the scatter plot, but they are not aware of the raw document data

[1] `http://www.mathworks.com/`
[2] `http://www.r-project.org/`
[3] `http://www.sas.com/`

IDEA'13, August 11th, 2013, Chicago, IL, USA.

type. Nonetheless, users would still want to explore data points in terms of which documents the data points represent. Furthermore, suppose another visualization is given, say, a bar graph showing the number of documents vs. a year when they were published. One crucial interaction that users want would be brushing-and-linking between different visualizations, say, to analyze when particular data points of interest in the scatter plot were published, etc.

As ways to handle these fundamental needs of users, this paper proposes to augment existing data analysis tools by imposing the capabilities of handling semantic objects so that essential interaction capabilities such as brushing-and-linking and details-on-demand, can be naturally supported. More specifically, from a command-line interface in the data analysis tools, we add a new set of functionalities that enable data analysis tools to maintain a new data type of semantic objects in addition to a generic numerical data type. Furthermore, based on these semantic objects that are explicitly maintained, we provide another set of functionalities that can flexibly assign the interaction capabilities to visualizations.

To realize our idea, we extend Matlab, one of the widely used analysis tools and computational languages, and develop a novel visual analytics system supporting newly proposed command sets while maintaining the full capabilities available in Matlab. The proposed set of commands and capabilities can be expanded in principle to other data analysis tools such as R. In this sense, our work has significant potential to be applied immediately in countless analysis problems and domains where these analysis tools are utilized.

To summarize, the main contribution of this paper includes

- Adding a new set of functionalities and new data type to the existing data analysis tool, Matlab.
- Imposing the capabilities of handling semantic objects with visualizations so that essential interactions can be supported.

The rest of this paper is organized as follows. Section 2 provides the overall description of the proposed system, and Section 3 presents the details of the supported commands in the system. Section 5 discusses related work. Section 4 describes the two real-world analysis examples using the proposed system. Finally, Section 6 concludes the paper and discusses about the future work.

2. DATA ANALYSIS TOOLS AUGMENTED WITH SEMANTIC OBJECTS

Starting with Matlab, the proposed visual analytics system incorporates the concept of *semantic objects*. Examples of semantic objects include any type of meaningful data entities such as documents, images, etc. Similar to the workspace in Matlab where all the variables are maintained, our system has a workspace for semantic objects where semantic objects are created, modified, and deleted, as shown in Figure 1 (1). These semantic objects contain the information about their contents in the forms of text documents or image files that can be accessed when users perform details-on-demand interactions.

Additionally, in order to flexibly associate these semantic objects with visualizations, our system also explicitly handles *visual objects* such as points, lines, axes, etc., composing a particular visualization. The visual object usually determined by numerical values, and they are represented by the graphical view. For example, a set of dots or cells are displayed on the scatter plot or the heatmap, respectively, in Figure 1.

Given such an explicit management of semantic as well as visual objects, our system provides the functionalities to create links between visual and semantic objects. In addition, the ordinary numerical matrix and/or vector objects from Matlab can also be linked with semantic objects in terms of the matrix/vector row or column indices. For example, each column of a term-by-document matrix, which is generated by a document data set, can be referred as a specific document. Therefore, it can be thought of that there are communication channels between visual object and the index of matrix, which makes it easy to apply visual analytic results to the input matrix.

Next, we will describe the details of the system in terms of the structure and functionalities.

2.1 Detailed Structure and Functionalities

Figure 1 is the main view of our system, which has three main panels labeled as '1' through '3' and an additional popup view labeled as '4.' The first component (Figure 1 (1)) shows the list of semantic objects. The various types of semantic objects, e.g., documents, images, other tabular data such as comma-separated value (CSV) files, etc., can be imported through the command-line interface. In general, the user can handle multiple lists of semantic object sets on the system.

The graphical view panel (Figure 1 (2)) plays a central role in the visualization part. We support five essential graphical views: a line graph, a scatter plot, a bar graph, a heatmap, and parallel coordinates. A dot, which is a part of a line, in the line graph or a dot in the scatter plot corresponds to a visual object. Similarly, a bar in the bar graph or a cell in the heatmap is mapped to a visual object.

Figure 1 (3) is a command-line interface for both Matlab and our own command sets. Once the user issues a particular command on the command-line interface, the interpreter module distinguishes whether the input command belongs to original Matlab command set or our own set we newly defined. To distinguish between the two sets, we attach an additional dot in front of our own command. For example, if an additional dot is put in the beggining of the command-line input, *.imagesc*, our system treats it as our own command set. Figure 2 shows a general workflow of the interpreter module. It begins when the user types some command. Then, this tool interprets whether it is a Matlab command or not. If it is Matlab's, the system opens communication channel with a Matlab processor. The Matlab executes a received command, then the result is delivered to our system. Otherwise, the system tries to match an appropriate command from our own set of commands and then executes it.

Figure 1 (4) is a semantic-object viewer as a popup window for details-on-demand interactions. For example, Figure 1 (4) shows a tabular form of semantic objects, whose original input data is from a CSV file. Depending on the semantic object type, such as documents, image, a tabular form of data, a corresponding semantic-object viewer opens up.

This tool, whose goal is mainly assisting visual analy-

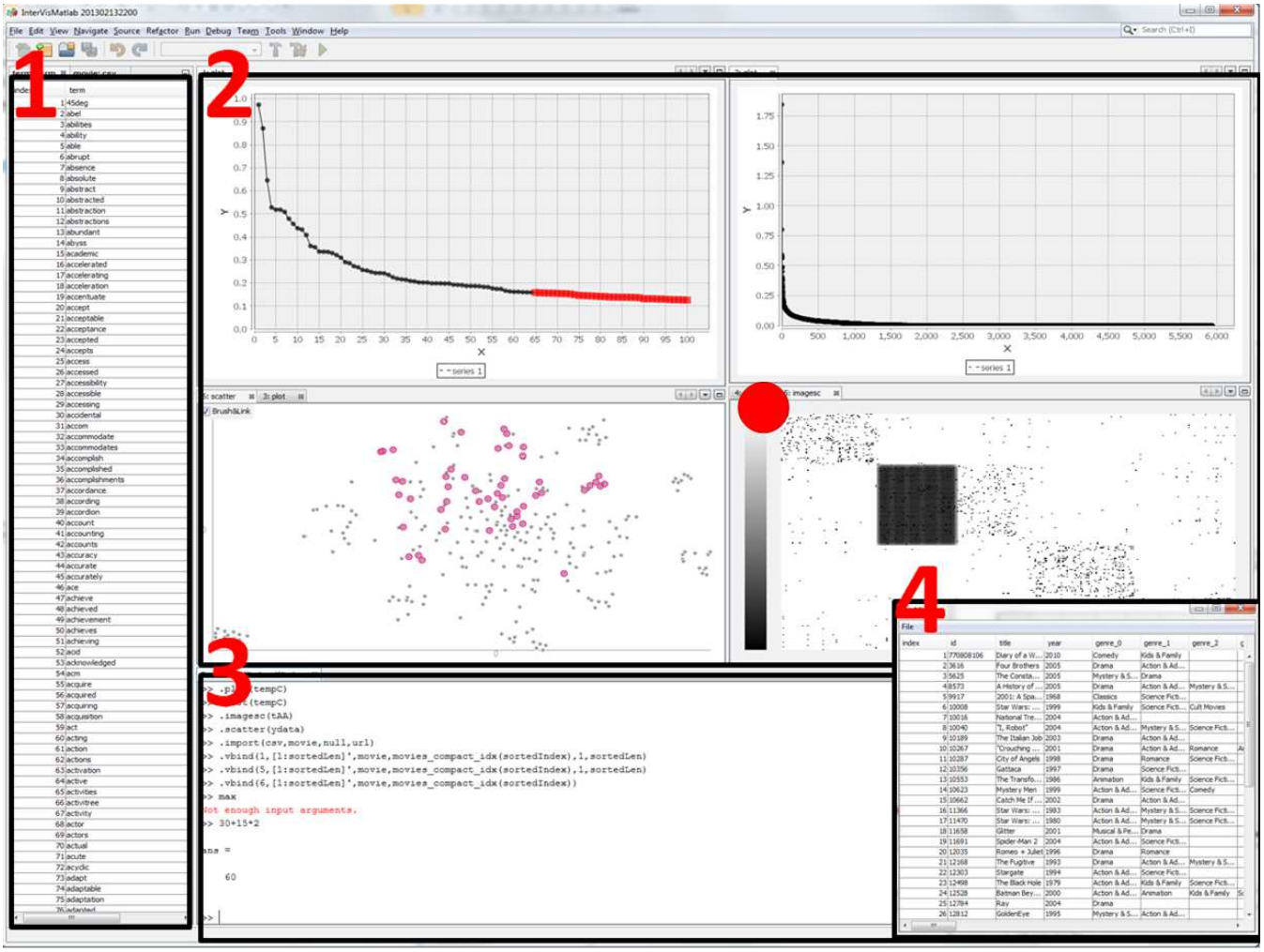


Figure 1: The left-side panel (1) shows the multiple set of semantic objects. The graphical view can be placed in the upper-right side (2). Matlab command or our own command could be executed through the panel placed in the lower side (3). The right-lower side (4) is a semantic-object viewer that shows detailed semantic information. These detailed semantic information can be accessible from visual objects that are linked with semantic objects.

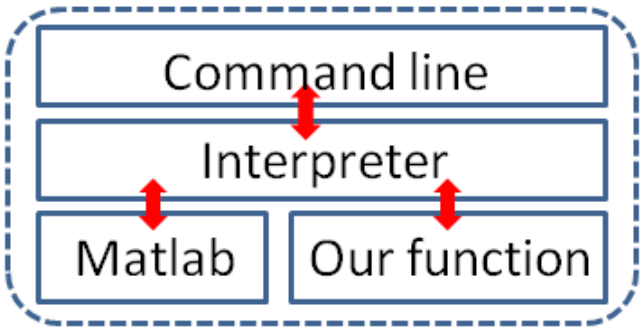


Figure 2: This is a workflow of the command. When the user sends a command, the system interprets it if it is for Matlab or not. If it is for Matlab, it sends command to the Matlab process and then receives computation results. Otherwise, it executes the appropriate function on its own.

sis, provides not only static graphical views but also several interactivities such as multiple linkage options between visual objects and semantic objects, which enable communication between them. It enables the user to look into detailed semantic information along with graphical views as well as to highlight related visual objects by using multiple brushing-and-linking operations. For instance, Figure 1, that is marked as a red circle, represents a heatmap of a term-by-document matrix, and each cell of the heatmap is connected to two semantic objects, a i-th term and a j-th document. In addition, dot representation on the scatter plot and the line graph in Figure 1 represent a dimensional reduction result of some documents and a term distribution of some group of documents, respectively. The user can freely select subset of visual objects by clicking them or dragging a rectangle around them on the plot. The detailed semantic information can be accessible from a semantic-object viewer that pops up by right-clicking the selection. The user can also gain a set of semantic object IDs from the selection. These selected semantic object IDs will be saved in the Matlab as a variable, and the user is allowed to freely use them in the command line. For example, the user can adjust the properties of the visual objects or make additional connections with different semantic objects. The other important functionality, the brushing-and-linking operation, highlight visual objects that linked with selected visual objects via semantic objects. The both detailed information and brushing-and-linking operations enable the user to compare the visual aspects of the graphical view and the meaning of them simultaneously with little additional efforts.

3. COMMANDS

To impose the general capabilities of handling semantic objects, we provide our own set of commands for, say, importing a list of semantic objects and linking between semantic objects and visual objects.

In the command-line interface provided by our system (Figure 1 (3)), the user can seamlessly perform both any Matlab commands and our own commands. The newly defined command sets allow the user to import semantic objects or to interactively manipulate the properties of the objects.

Next, we introduce our functionalities with comprehensive examples.

3.1 Graphical View

The graphical view is an essential view in visual analytic tools. We provide five well-known graphical views: a scatter plot (*.scatter*), a heatmap (*.imagesc*), a bar graph (*.bar*), a line graph (*.plot*), and parallel coordinates (*.pcs*). Matlab already has these kinds of graphical views, but in order to impose additional interactive capabilities, the proposed system generates our own graphical views independently of Matlab. Each visual object can be connected with multiple semantic objects. In addition, the user can change the properties of the visual object to focus on the subset of important visual objects or for enhanced visual understanding.

To generate views, it requires a $N \times M$ matrix as an input argument. In some cases, N represents the number of visual objects, and M means the dimension of each data item. For example, we need N by M matrix to render N data items that have M-dimensional information on the parallel coordinate plot. However, in the case of heatmap, each cell of a heatmap is represented as a visual object and consequently it draws $N \times M$ visual objects on the view. Every visual object has its own ID which is necessary to bind it with semantic objects. The ID is automatically allocated when creating a view. The heatmap view assigns the visual object ID to each cell using row-major order. Each visual object has various properties and can be adjusted by calling *'.setcolor'* or *'.setsize'* functions. The user can also specify properties before creating a graphical view by passing pairs of the Name-Value type argument in the function parameter.

3.2 Bind

We need additional commands for object binding. There are three types of objects: the visual object, the semantic object, and the matrix object. The visual object is presented in the graphical view. The semantic object can be any kind of meaningful information such as a document/image file or a just string or a CSV file. The matrix object is a general matrix/vector variable in Matlab, which represents visual objects on the graphical view.

There are two types of bind operations: *.vbind* for binding the visual and semantic objects, and *.mbind* for binding indices of a matrix object and the semantic objects. *.vbind* requires a list of the visual object IDs and a corresponding list of the semantic object IDs to make correspondences betweem them. Each visual object can be linked with multiple semantic objects. For instance, we consider that N document data items are visualized on the 2D scatter plot, and the documents are already clustered via some clustering algorithm. Then, each visual object can be referred to as a specific document or a particular cluster it belongs to. Furthermore, if we apply multiple clustering algorithms, we can compare different groups of clusters by connecting multiple cluster memberships. When the visual objects are linked with the semantic objects, other visual objects that are linked with the same semantic object are all inter-linked via same semantic objects. This linkage makes it possible to perform the brushing-and-linking functionality. Next, *.mbind* requires the list of the row/column indices of a matrix and the corresponding list of semantic object IDs. Therefore,

the indices of the matrix and the set of semantic objects are also inter-connected via same semantic objects. Consequently, the user can obtain a subset of indices of the matrix from the visual objects by using a command, *.mindex*, that requires a variable name and a set of semantic object IDs. It enables the user to conveniently modify the related matrix of visual objects.

3.3 Implementations

Our tool is currently implemented in Java (version 1.6) and Matlab 7.13 (R2011). To commmunicate with Matlab, we use Java Matlab Interface (JMI). [4] In addition, We embed Beanshell library [5] for enabling dynamic execution of the Matlab syntax. Finally, JFreeChart [6] is adopted to display the high-quality line chart.

4. ANALYSES

In this section, we show the two different case studies, for a graph data set and a document data set. To highlight our contribution, we represent how it is convenient to analyze the data if the user can easily approach to the meaning of the visual object. In addition, we present the way of data refining and the way of increasing the document clustering quality by making a list of top frequency terms in each cluster.

4.1 Data Sets

First, the graph data set is generated from the pairwise similarity scores obtained from a popular movie rating website.[7] To be specific, we crawl the information about almost 200,000 movies and make edges between movies if they have a value of similarity. As it is too large to deal with or visualize, we select a subset of 300 randomly choosen movies that have degrees between 30 and 80 so that a sub-graph is not too sparse nor dense.

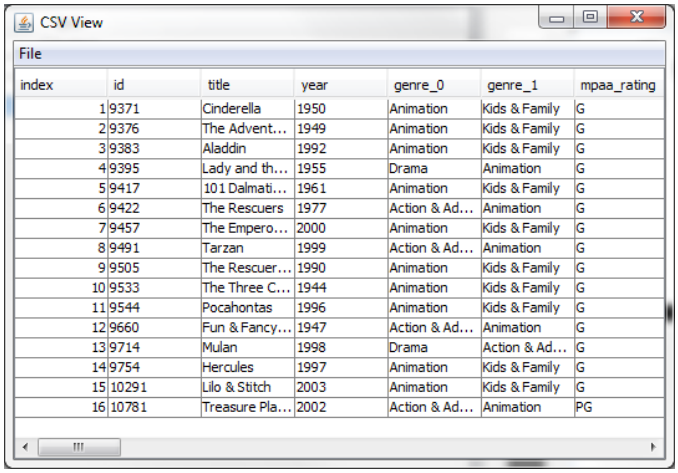
Second, we use a document data set composed of InfoVis and VAST conference papers from 2001 to 2012.[8] We build up a bag-of-words representation for each document and build a term-document matrix. The total number of document is 515, and the total number of word is 5,935 after a stopword removal step.

4.2 Usage Scenarios - Movie Data

To gain nontrivial insight about the data, we provide multiple views simultaneously as well as multiple brushing-and-linking functionalities. Figure 3 shows the clustering result of the graph as a heatmap and the dimensionality reduction result of the graph as a scatter plot. The heatmap shows the clustering results that computed by nonnegative matrix factorization (NMF) [6], and the scatter plot shows the relationship among movies that comes from the dimension reduction algorithm, t-distributed stochastic neighborhood embedding [9]. To make clear understanding of the relationship shown in the scatter plot, we apply different colors to visual objects based on the cluster membership.

The set of blue visual objects on the scatter plot is distant from others. Based on the semantic objects, the connected

[4] https://code.google.com/p/matlabcontrol/wiki/JMI
[5] http://www.beanshell.org/
[6] http://www.jfree.org/jfreechart/
[7] http://www.rottentomatoes.com/
[8] http://www.cc.gatech.edu/gvu/ii/jigsaw/datafiles.html



index	id	title	year	genre_0	genre_1	mpaa_rating
1	9371	Cinderella	1950	Animation	Kids & Family	G
2	9376	The Advent...	1949	Animation	Kids & Family	G
3	9383	Aladdin	1992	Animation	Kids & Family	G
4	9395	Lady and th...	1955	Drama	Animation	G
5	9417	101 Dalmati...	1961	Animation	Kids & Family	G
6	9422	The Rescuers	1977	Action & Ad...	Animation	G
7	9457	The Empero...	2000	Animation	Kids & Family	G
8	9491	Tarzan	1999	Action & Ad...	Animation	G
9	9505	The Rescuer...	1990	Animation	Kids & Family	G
10	9533	The Three C...	1944	Animation	Kids & Family	G
11	9544	Pocahontas	1996	Animation	Kids & Family	G
12	9660	Fun & Fancy...	1947	Action & Ad...	Animation	G
13	9714	Mulan	1998	Drama	Action & Ad...	G
14	9754	Hercules	1997	Animation	Kids & Family	G
15	10291	Lilo & Stitch	2003	Animation	Kids & Family	G
16	10781	Treasure Pla...	2002	Action & Ad...	Animation	PG

Figure 4: The detailed movie information of the isolated group in Figure 3.

columns in the heatmap are highlighted when we select the isolated blue group, as shown in the zoomed-in part in Figure 3. The heatmap shows that the selected movies are strongly connected to each other. In fact, the main genres of these movies are 'Animation' and 'Kids & Family,' which are preferred by children.

The strength of our tool is that we can use any kinds of the Matlab variables and functionalities. We can obtain the selected column indices of the heatmap, which are the same as the matrix indices. With these column indices, we generate constraints that these movies should be clustered together, which was not the case in the current clustering result. To this end, we run a semi-supervised clustering based on NMF algorithm, which is already implemented in Matlab. This algorithm causes new movies to be included in the constrained cluster if they are closely related to the movies in the constrained cluster.

After running semi-supervised NMF algorithm with the above-describe constraint and a number of clusters as five, we obtain the new clustering result as shown in Figure 5. Clusters are shown to be relatively balanced compared with the previous result. In addition, the last cluster seems to have very strong internal connections. When we open detailed movie information of this cluster, two additional movies, marked as red rectangles in Figure 5, are newly joined into this group. Interestingly, the movie marked with a red circle in Figure 5 does not belong to "Animation", but the movie is shown to be closely related to "Animation" genre because it is filmed by "Walt Disney".

4.3 Usage Scenarios - document data

In this scenario, our system shows the topics of document clusters and helps refining the topic of each group. We construct a term-by-document matrix and run a simple NMF algorithm with rank 10 to obtain topic clusters. To verify the quality of the cluster result, we may want to see the term-wise distribution of each group and check how clearly the representative terms represent in each group. The higher value the term has, the more closely the term is related with the group.

For simplicity, we only show four selected cluster results out of 10 cluster results: third, fifth, seventh, and ninth

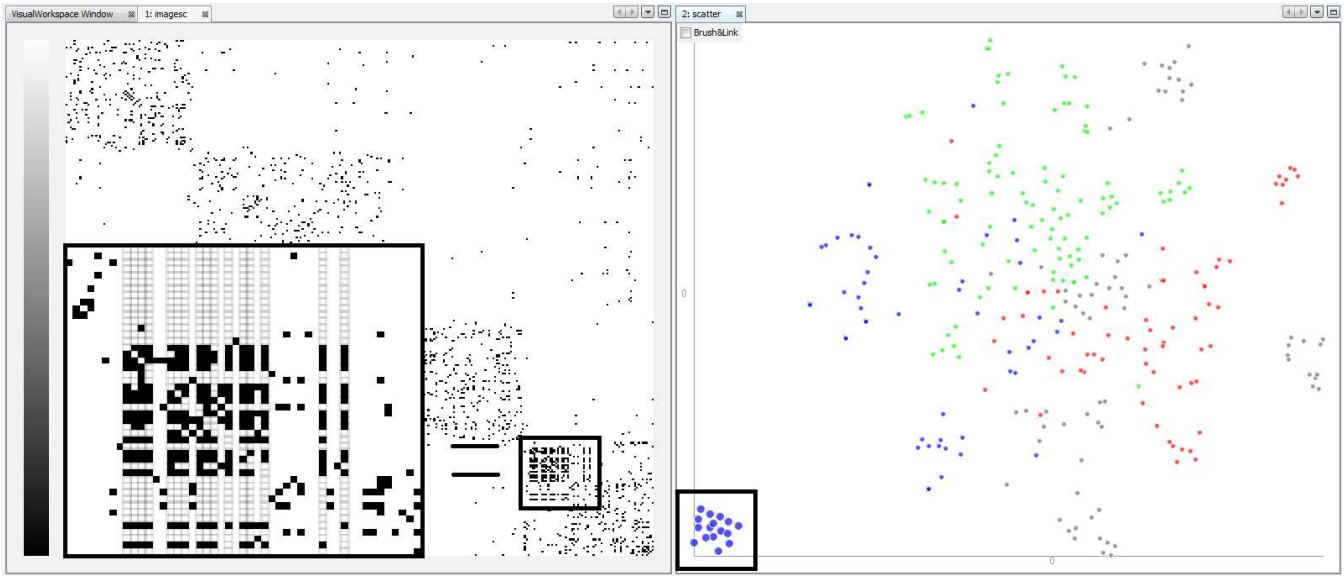


Figure 3: The heatmap represents the clustering result that computed by nonnegative matrix factorization, and the scatter plot represents the dimension reductino result computed by t-distributed stochastic neighborhood embedding.

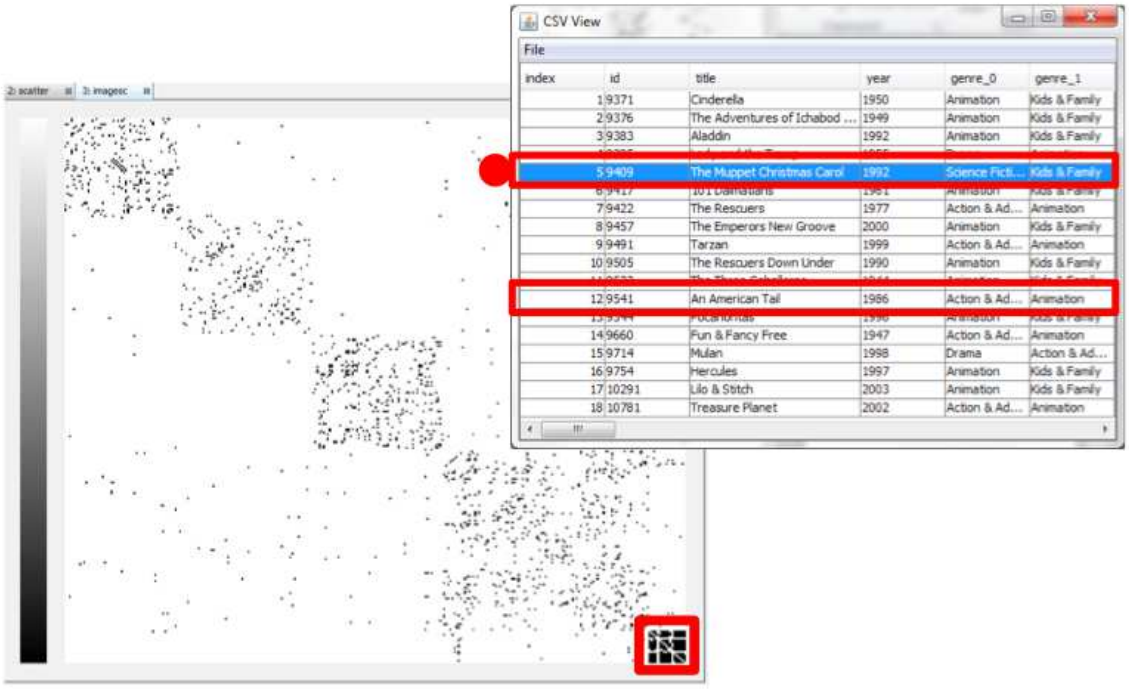


Figure 5: The heatmap represents the semi-supervised clustering result of the movie graph. The semantic-object viewer represents the detailed movie information in the cluster highlighted as a red rectangle. The two movies marked in a red color in the viewer are additionally joined movies to this cluster due to semi-supervision, and the genre of the movie with a red circle does not belong to "Animation" genre while the other movies in the cluster do.

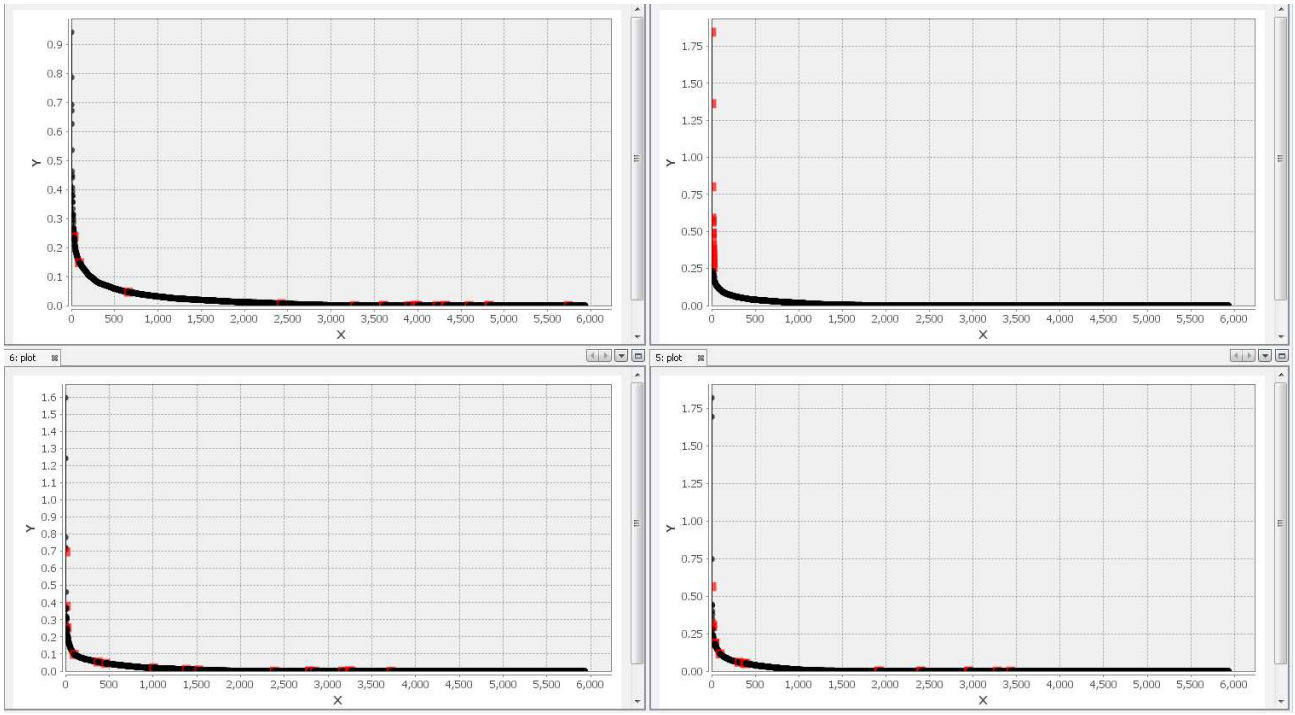


Figure 6: It represents the sorted line graph of term distribution of the group 3 (top-left), 5 (top-right), 7 (bottom-left), and 9 (bottom-right). Top 100 most frequent keywords in the cluster fifth (top-right) are selected and corresponding keywords are highlighted in the other clusters.

Table 1: The three most frequent words in each cluster.

Cluster 3	Cluster 5	Cluster 7	Cluster 9
information	multidimensional	analytics	text
visualizations	dimensions	analytic	documents
context	dimensional	analysis	document

Table 2: The three most frequent words after refining a term-by-document matrix.

Cluster 3	Cluster 5	Cluster 7	Cluster 9
visualizations	dimension	analytic	text
context	parallel	collaborative	document
design	coordinate	visual	corpora

clusters. We draw line graphs of term-wise distributions of topic clusters in an order of decreasing term weights. (Figure 6). This tool binds each value of the line graph with the corresponding term in the dictionary so that the user can easily check which words strongly belong to the specific group. We select top 100 keywords that have the highest values in each group by drawing a rectangle around the line graph. Brushing-and-linking informs the user that corresponding terms have lower coefficients in the other clusters, which explains that each cluster represents distinct topic. Figure 6 shows the one case, the selection of fifth cluster, of brushing-and-linking.

Among the top 100 keywords, we show top three keywords in the Table 1 to understand the topical meaning of each group. The top keywords in each of the seventh and ninth clusters have the same/similar meanings, and thus we merged these words into a single dimension in the term-by-document matrix. In the case of the third cluster, however, the top word "information" does not have much meaning to the user. Therefore, we decided to delete this word from the term-by-document matrix. In this way, we can refine the term-by-document matrix and make the clear topic for each group.

Table 2 represents the top three most frequent words after refining the matrix. In fifth cluster, we can detect more meaningful information such as parallel coordinates. Furthermore, we can see various meaningful words in seventh cluster compared with the previous result.

5. RELATED WORK

In this section, we briefly discuss some of the relevant literature in the context of data analysis tools emphasizing exploratory analysis aspects via interactive visualization. Traditionally, Turkey et al. [8] introduced primitive ways of effective data visualization. Starting from this, many interactive visualization tools have been developed, and some relevant work will be reviewed as follows: NodeTrix [4], Guess [1], iVisClassifier [2], and Tableau [9].

NodeTrix [4] visualizes not only overall graph relationships of the data but also detailed adjacency matrix in one view. The main contribution of NodeTrix is that it shows the overall relationships through graph representations as well as detailed relationships through adjacency matrix. It helps the users gain intuitions from both visualizations, but ex-

[9] http://www.tableausoftware.com/

tendability is not excellent due to the limited type of graphical viewer and domain. Next, Guess [1] is a graph exploration tool, which allows the user to interact with the graph through its own interpreter, Gython. Guess has a general command-line interface and it gives us some inspiration, but it is only feasible to graph representation. Testbed [3] shows multiple views concurrently to help analyzing the data set. Next, Tableau, which is a commercial successor of Polaris [7], is a database visualization system. However, all of these tools may be appropriate in their own domain, and the main obstacle for these tools is that they require additional learning efforts to use the tools.

The tool that was introduced in Kehrer et al. [5] tries to apply the general statistical tool, R, to the specific functionality, solving statistical problem. The main contribution of the tool is that it automatically computes statistical results during the data selection and dynamically updates the results for the interactive visualization. The basic idea of this paper is similar to ours in that it tries to improve the interactivity of general statistical tool, but the presented approaches are somewhat limited to the generic data types and specific functionalities, selection and updating.

In this paper, we suggest the tool that imports various data types in Matlab environment in order to seamlessly generalize and enhance the interactivity with visualizations.

6. CONCLUSION AND FUTURE WORK

Previously, it was challenging to have the concept of semantic objects originated from application domains in general-purpose data analysis tools. Some domain-specific tools may naturally provide various interaction capabilities applicable to the particular data type. However, it is not always able to be generalized to other domains, and thus the user should consider various tools depending on the domain. Therefore, starting from one of the widely-used tool, Matlab, we have proposed binding capabilities between general semantic objects and visual objects for helping visual analysis. In addition, our system does not compromise any of the Matlab functionalities, and thus the user can still use any kinds of Matlab functionalities and commands.

There are still a few issues to be improved in our system. We use command-line interface, but it usually needs some time to get used to. Thus, we plan to implement an additional window panel that visualize summary of current binding information. In addition, we plan to refine the user interfaces, especially by improving the binding operation into easier GUI interfaces.

7. ACKNOWLEDGMENTS

The work of these authors was supported in part by the National Science Foundation grant CCF-0808863. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] E. Adar. Guess: a language and interface for graph exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 791–800, New York, NY, USA, 2006. ACM.

[2] J. Choo, H. Lee, J. Kihm, and H. Park. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 27–34. IEEE, 2010.

[3] J. Choo, H. Lee, Z. Liu, J. Stasko, and H. Park. An interactive visual testbed system for dimension reduction and clustering of large-scale high-dimensional data. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*. To appear, 2013.

[4] N. Henry, J.-D. Fekete, and M. J. McGuffin. Nodetrix: a hybrid visualization of social networks. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1302–1309, 2007.

[5] J. Kehrer, R. N. Boubela, P. Filzmoser, and H. Piringer. A generic model for the integration of interactive visualization and statistical computing using R. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 233–234. IEEE, 2012.

[6] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.

[7] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):52–65, 2002.

[8] J. W. Tukey. Exploratory data analysis. *Reading, MA*, 231, 1977.

[9] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Online Spatial Data Analysis and Visualization System

Yun Lu, Mingjin Zhang, Tao Li, Yudong Guang, Naphtali Rishe
School of Computing and Information Sciences
Florida International University, Miami, Florida, 33199, USA
{yun,zhangm,taoli,yuguang,rishen}@cs.fiu.edu

ABSTRACT

With the exponential growth of the usage of web map services, the geo data analysis has become more and more popular. This paper develops an online spatial data analysis and visualization system, TerraFly GeoCloud, which facilitates end users to visualize and analyze spatial data, and to share the analysis results.

Built on the TerraFly Geo spatial database, TerraFly GeoCloud is an extra layer running upon the TerraFly map and can efficiently support many different visualization functions and spatial data analysis models. Furthermore, users can create unique URLs to visualize and share the analysis results. TerraFly GeoCloud also enables the MapQL technology to customize map visualization using SQL-like statements. The system is available at http://terrafly.fiu.edu/GeoCloud/.

1. INTRODUCTION

With the exponential growth of the World Wide Web, there are many domains, such as water management, crime mapping, disease analysis, and real estate, open to Geographic Information System (GIS) applications. The Web can provide a giant amount of information to a multitude of users, making GIS available to a wider range of public users than ever before. Web-based map services are the most important application of modern GIS systems. For example, Google Maps currently has more than 350 million users. There are also a rapidly growing number of geo-enabled applications which utilize web map services on traditional computing platforms as well as the emerging mobile devices.

However, due to the highly complex and dynamic nature of GIS systems, it is quite challenging for the end users to quickly understand and analyze the spatial data, and to efficiently share their own data and analysis results to others. First, typical geographic visualization tools are complicated and fussy with a lot of low-level details, thus they are difficult to use for spatial data analysis. Second, the analysis of large amount spatial data is very resource-consuming. Third, current spatial data visualization tools are not well integrated for map developers and it is difficult for end users to create the map applications on their own spatial datasets.

To address the above challenges, this paper presents TerraFly GeoCloud, an online spatial data analysis and visualization system, which allows end users to easily visualize and share various types of spatial data. First, TerraFly GeoCloud can accurately visualize and manipulate point and polygon spatial data with just a few clicks. Second, TerraFly GeoCloud employs an analysis engine to support the online analysis of spatial data, and the visualization of the analysis results. Many different spatial analysis functionalities are provided by the analysis engine. Third, based on the TerraFly map API, TerraFly GeoCloud offers a MapQL language with SQL-like statements to execute spatial queries, and render maps to visualize the customized query results.

Our TerraFly GeoCloud online spatial data analysis and visualization system is built upon the TerraFly system using TerraFly Maps API and JavaScript TerraFly API add-ons in a high performance cloud Environment. The function modules in the analysis engine are implemented using C and R language and python scripts. Comparing with current GIS applications, our system is more user-friendly and offers better usability in the analysis and visualization of spatial data. The system is available at http://terrafly.fiu.edu/GeoCloud/.

The rest of this paper is organized as follows: Section 2 presents the background and motivation; Sections 3 describes the architecture of TerraFly GeoCloud; Section 4 describes the visualization solutions in TerraFly GeoCloud; Section 5 presents a case study on the online spatial analysis; Section 6 discusses the related work; and finally Section 7 concludes the paper.

2. BACKGROUND

2.1 TerraFly

TerraFly is a system for querying and visualizing of geospatial data developed by High Performance Database Research Center (HPDRC) lab in Florida International University (FIU). This TerraFly system serves worldwide web map requests over 125 countries and regions, providing users with customized aerial photography, satellite imagery and various overlays, such as street names, roads, restaurants, services and demographic data [1].

TerraFly Application Programming Interface (API) allows rapid deployment of interactive Web applications and has been used to produce systems for disaster mitigation, ecology, real estate, tourism, and municipalities. TerraFly's Web-based client interface is accessible from anywhere via any standard Web browser, with no client software to install.

TerraFly allows users to virtually ‘fly’ over enormous geographic information simply via a web browser with a bunch of advanced functionalities and features such as user-friendly geospatial querying interface, map display with user-specific granularity, real-time data suppliers, demographic analysis, annotation, route dissemination via autopilots and application programming interface (API) for web sites, etc. [1][2].

TerraFly's server farm ingests geo-locates, cleanses, mosaics, and cross-references 40TB of base map data and user-specific data streams. The 40TB TerraFly data collection includes, among others, 1-meter aerial photography of almost the entire United States and 3-inch to 1-foot full-color recent imagery of major urban areas. TerraFly vector collection includes 400 million geo-located objects, 50 billion data fields, 40 million polylines, 120

IDEA'13, August 11th, 2013, Chicago, IL, USA.

million polygons, including: all US and Canada roads, the US Census demographic and socioeconomic datasets, 110 million parcels with property lines and ownership data, 15 million records of businesses with company stats and management roles and contacts, 2 million physicians with expertise detail, various public place databases (including the USGS GNIS and NGA GNS), Wikipedia, extensive global environmental data (including daily feeds from NASA and NOAA satellites and the USGS water gauges), and hundreds of other datasets [3].

2.2 Visualizing spatial data

Information visualization (or data visualization) techniques are able to present the data and patterns in a visual form that is intuitive and easily comprehendible, allow users to derive insights from the data, and support user interactions [4].

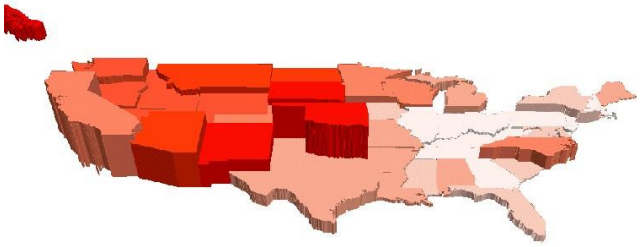


Figure 1: Population Total (height) vs. Density (color) of US

For example, Figure 1 shows the map of Native American population statistics which has the geographic spatial dimensions and several data dimensions. The figure displays both the total population and the population density on a map, and users can easily gain some insights on the data by a glance [5]. In addition, visualizing spatial data can also help end users interpret and understand spatial data mining results. They can get a better understanding on the discovered patterns.

Visualizing the objects in geo-spatial data is as important as the data itself. The visualization task becomes more challenging as both the data dimensionality and richness in the object representation increase. In TerraFly GeoCloud, we have devoted lots of effort to address the visualization challenge including the visualization of multi-dimensional data and the flexible user interaction.

TerraFly GeoCloud integrates spatial data mining and data visualization. The integration of spatial data mining and information visualization has been widely to discover hidden patterns. For spatial data mining to be effective, it is important to include the visualization techniques in the mining process and to generate the discovered patterns for a more comprehensive visual view [6].

2.3 Map Rendering

The process of rendering a map generally means taking raw geospatial data and making a visual map from it. Often it applies more specifically to the production of a raster image, or a set of raster tiles, but it can refer to the production of map outputs in vector-based formats. "3D rendering" is also possible when taking the map data as an input. The ability of rendering maps in new and interesting styles, or highlighting features of special interest, is one of the most exciting aspects in spatial data analysis and visualization.

TerraFly map render engine is a toolkit for rendering maps and is used to render the main map layers. It supports a variety of geospatial data formats and provides flexible styling options for designing many different kinds of maps, and the render speed is fast [7][8].

TerraFly map render engine is written in C++ and can be used as a web service. It uses the AGG library and offers anti-aliasing rendering with pixel accuracy. It can read different kind of file like PostGIS, TIFF rasters, .osm files, and other shape files. Packages are available for both Window and Linux [8].

3. TerraFly GeoCloud

Figure 2 shows the system architecture of TerraFly GeoCloud. Based on the current TerraFly system including the Map API and all sorts of TerraFly data, we developed the TerraFly GeoCloud system to perform online spatial data analysis and visualization. In TerraFly GeoCloud, users can import and visualize various types of spatial data (data with geo-location information) on the TerraFly map, edit the data, perform spatial data analysis, and visualize and share the analysis results to others. Available spatial data sources in TerraFly GeoCloud include but not limited to demographic census, real estate, disaster, hydrology, retail, crime, and disease. In addition, the system supports MapQL, which is a technology to customize map visualization using SQL-like statements.

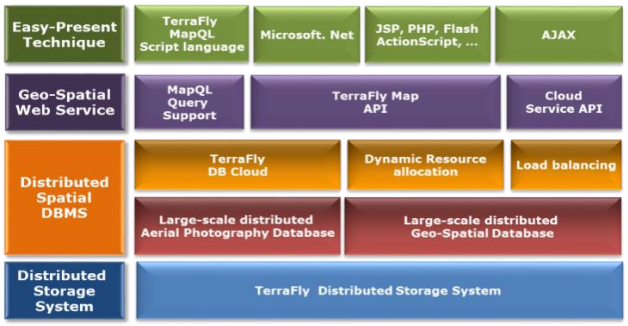


Figure 2: The Architecture of TerraFly GeoCloud

The spatial data analysis functions provided by TerraFly GeoCloud include spatial data visualization (visualizing the spatial data), spatial dependency and autocorrelation (checking for spatial dependencies), spatial clustering (grouping similar spatial objects), and Kriging (geo-statistical estimator for unobserved locations).

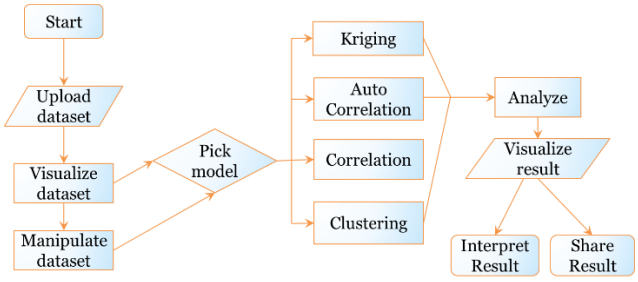


Figure 3: The Workflow of TerraFly GeoCloud Analysis

Figure 3 shows the data analysis workflow of the TerraFly GeoCloud system. Users first *upload datasets* to the system, or view the available datasets in the system. They can then *visualize the data sets* with customized appearances. By *Manipulate dataset*, users can edit the dataset and perform pre-processing (e.g., adding more columns). Followed by pre-processing, users can choose proper spatial analysis functions and perform the analysis. After the analysis, they can visualize the results and are also able to share them with others.

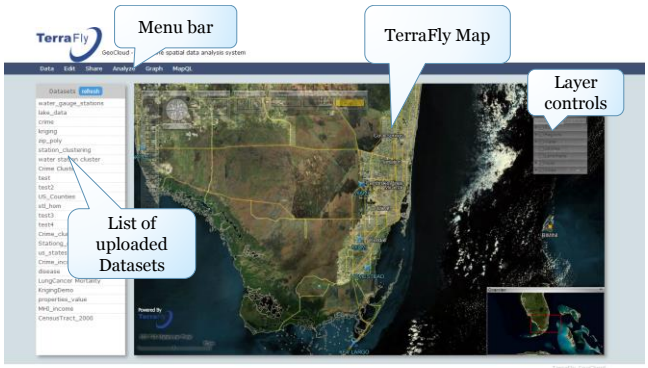


Figure 4: Interface of TerraFly GeoCloud

Figure 4 showed the interface of the TerraFly GeoCloud system. The top bar is the menu of all functions, including *Data*, *analysis*, *Graph*, *Share*, and *MapQL*. The left side shows the available datasets, including both the uploaded datasets from the user and the existing datasets in the system. The right map is the main map from TerraFly. This map is composed by TerraFly API, and it includes a detailed base map and diverse overlays which can present different kinds of geographical data.

TerraFly GeoCloud also provides MapQL spatial query and render tools. MapQL supports SQL-like statements to realize the spatial query, and after that, render the map according to users' inputs. MapQL tools can help users visualize their own data using a simple statement. This provides users with a better mechanism to easily visualize geographical data and analysis results.

4. Visualization in TerraFly GeoCloud

4.1 Spatial Data Visualization

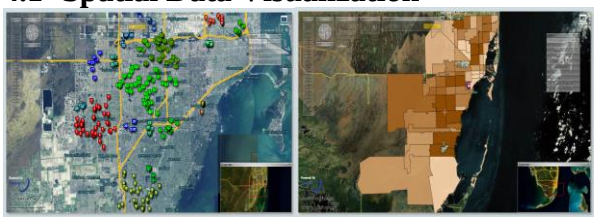


Figure 5: Spatial Data Visualization: Left subfigure: Point Data; Right subfigure: Polygon Data

For spatial data visualization, the system supports both point data and polygon data and users can choose color or color range of data for displaying. As shown in Figure 5, the point data is displayed on left, and the polygen data is displayed on the right. The data labels are shown on the base map as extra layers for point data, and the data polygons are shown on the base map for polygon data. Many different visualization choices are supported for both point data and polygon data. For point data, user can customize the icon style, icon color or color range, label value and so on. For polygon data, user can customize the fill color or color range, fill alpha, line color, line width, line alpha, label value and so on.

4.2 Spatial Data Mining Results Visualization

TerraFly GeoCloud integrates spatial data mining and data visualization. The spatial data mining results can be easily visualized. In addition, visualization can often be incorporated into the spatial mining process.

4.2.1 Spatial dependency and Auto-Correlation

Spatial dependency is the co-variation of properties within geographic space: characteristics at proximal locations that appear to be correlated, either positively or negatively. Spatial dependency leads to the spatial autocorrelation problem in statistics [9].

Spatial autocorrelation is more complex than one-dimensional autocorrelation because spatial correlation is multi-dimensional (i.e. 2 or 3 dimensions of space) and multi-directional. The TerraFly GeoCloud system provides auto-correlation analysis tools to discover spatial dependencies in a geographic space, including global and local clusters analysis where Moran's I measure is used [10].

Formally, Moran's I, the slope of the line, estimates the overall global degree of spatial autocorrelation as follows:

$$I = \frac{n}{\sum_i^n \sum_j^n w_{ij}} \times \frac{\sum_i^n \sum_j^n w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_i^n (y_j - \bar{y})^2}$$

where w_{ij} is the weight, $w_{ij}=1$ if locations i and j are adjacent and zero otherwise $w_{ii}=0$ (a region is not adjacent to itself).y_i and $\bar{y}$ are the variable in the ith location and the mean of the variable, respectively. n is the total number of observations. Moran's I is used to test hypotheses concerning the correlation, ranging between –1.0 and +1.0.

Moran's I measures can be displayed as a checkerboard where a positive Moran's I measure indicates the clustering of similar values and a negative Moran's I measure indicate dissimilar values. TerraFly GeoCloud system provides auto-correlation analysis tools to check for spatial dependencies in a geographic space, including global and local clusters analysis

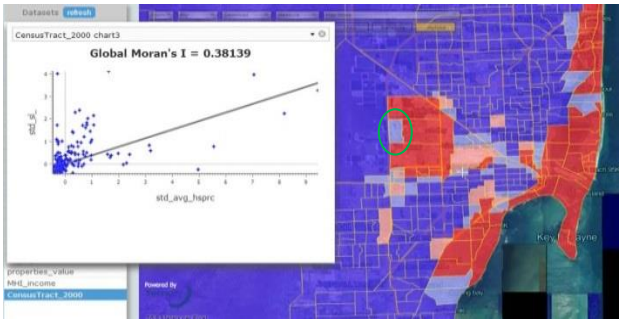


Figure 6: Average properties price by zip code in Miami

Local Moran's I is a local spatial autocorrelation statistic based on the Moran's I statistic. It was developed by Anselin as a local indicator of spatial association or LISA statistic [11]. The fact that Moran's I is a summation of individual cross products is exploited by the "Local indicators of spatial association" (LISA) to evaluate the clustering in those individual units by calculating Local Moran's I for each spatial unit and evaluating the statistical significance for each Ii. From the previous equation we then obtain:

$$I_i = z_i \sum_j^n w_{ij} z_j$$

where z_i are the deviations from the mean of y_i, and the weights are row standardized.

Figure 6 shows an example of spatial auto-correlation analysis on the average properties price by zip code data in Miami (polygon

data). Each dot here in the scatterplot corresponds to one zip code. The first and third quadrants of the plot represent positive associations (high-high and low-low), while the second and fourth quadrants represent associations (low-high, high-low). For example, the *green circle area* is in the low-high quadrants. The density of the quadrants represents the dominating local spatial process. The properties in Miami Beach are more expensive, and are in the high-high area.

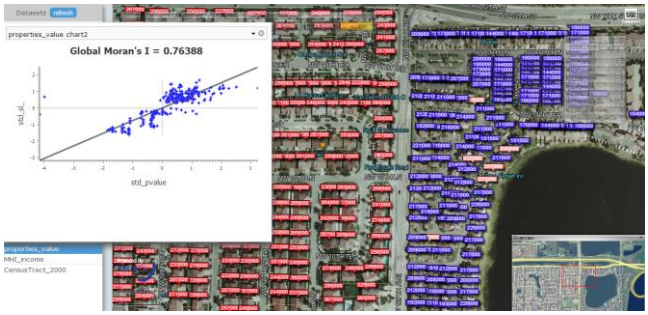


Figure 7: Properties value in Miami

Figure 7 presents the auto-correlation analysis results on the individual properties price in Miami (point data). Each dot here in the scatterplot corresponds to one property. As the figure shows, the properties near the big lake are cheaper, while the properties along the west are more expensive.

4.2.2 Spatial Data Clustering

The TerraFly GeoCloud system supports the DBSCAN (for density-based spatial clustering of applications with noise) data clustering algorithm [12]. It is a density-based clustering algorithm because it finds a number of clusters starting from the estimated density distribution of corresponding nodes.

DBSCAN requires two parameters as the input: *eps* and the minimum number of points required to form a cluster *minPts*. It starts with an arbitrary starting point that has not been visited so far. This point's neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as a noise point [12]. If a point is found to be a dense part of a cluster, its neighborhood is also part of that cluster. Hence, all points that are found within the neighborhood are added. This process continues until the density-connected cluster is completely identified. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise points [13].

Figure 8 shows an example of DBSCAN clustering on the crime data in Miami. As shown in Figure 6, each point is an individual crime record marked on the place where the crime happened, and the number displayed in the label is the crime ID. By using the clustering algorithm, the crime records are grouped, and different clusters are represented by different colors on the map.

Figure 8: DBSCAN clustering on the crime data in Miami

4.2.3 Kriging

Kriging is a geo-statistical estimator that infers the value of a random field at an unobserved location (e.g. elevation as a function of geographic coordinates) from samples (see spatial analysis) [14].

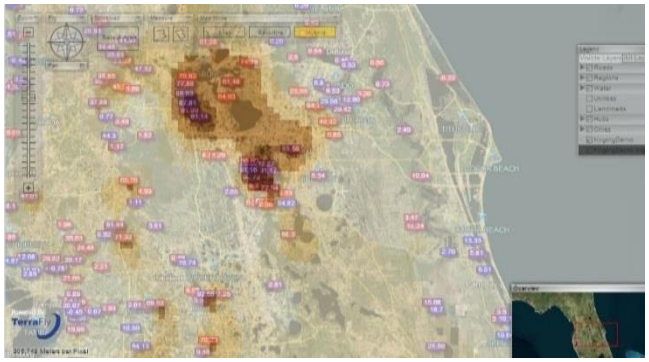


Figure 9: Kriging data of the water level in Florida

Figure 9 shows an example of Kriging. The data set is the water level from water stations in central Florida. Note that not all the water surfaces are measured by water stations. The Kriging results are estimates of the water levels and are shown by the yellow layer.

4.3 Customized Map Visualization (Supported by MapQL)

TerraFly GeoCloud also provides MapQL spatial query and render tools, which supports SQL-like statements to facilitate the spatial query and more importantly, render the map according users' requests. This is a better interface than API to facilitate developer and end user to use the TerraFly map as their wish. By using MapQL tools, users can easily create their own maps.

4.3.1 Implementation

The implementation of MapQL is shown in Figure 10. The input of the whole procedure is MapQL statements, and the output is map visualization rendered by the MapQL engine.

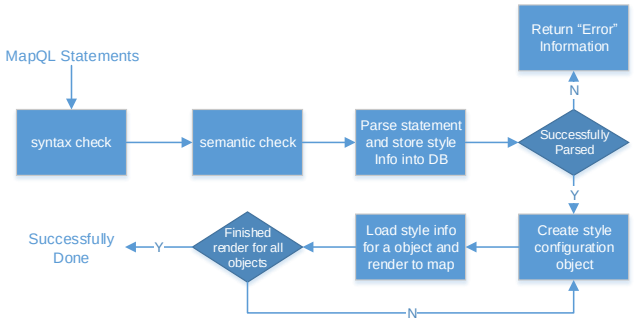


Figure 10: MapQL implementation

Shown in Figure 10, the first step is syntax check of the statements. Syntax check guarantees that the syntax conforms to the standard, such as the spelling-check of the reserved words. Semantic check ensures that the data source name and metadata which MapQL statements want to visit are correct. After the above two checks, system will parse the statements and store the parse results including the style information into a spatial database. The style information includes where to render and what to render. After all the style information is stored, system will create style configuration objects for render. The last step is for each object, load the style information form spatial database and render to the map according to the style information.

We implemented the MapQL tools using C++. For the last step which is rendering the objects to the map visualization, we employed the TerraFly map render engine [8].

For example, if we want to query the house prices near Florida International University, we use MapQL statements like this:

```
SELECT
    '/var/www/cgi-bin/house.png' AS T_ICON_PATH,
r.price AS T_LABEL,
    '15' AS T_LABEL_SIZE,
r.geo AS GEO
FROM
realtor_20121116 r
WHERE
ST_Distance(r.geo,  GeomFromText('POINT(-80.376283 25.757228)')) <
0.03;
```

There are four reseverd words in the statements, *T_ICON_PATH* , *T_LABEL*, *T_LABEL_SIZE* , and *GEO*. We use *T_ICON_PATH* to store the customized icon. Here we choose a local png file as icon. *T_LABEL* denotes that icon label that will be shown on the map, . *T_LABEL_SIZE* is the pixel size of the label; and *GEO* is the spatial search geometry.

The statement goes through the syntax check first. If there is incorrect usage of reserved words or wrong spelling of the syntax, it will be corrected or Error information will be sent to users. For example, if the spelling of "*select*" is not correct, Error information will be sent to user. Semantic check makes sure that the data source name *realtor_20121116 and* metadata *r. price* and *r.geo* are exist and available.

After the checks, the system parsed the statements. The SQL part will return corresponding results including the locations and names of nearby objects, the MapQL part will collect the style information like icon path and icon label style. Both of them are stored into a spatial database. The system then created style configuration objects for query results. The last step is rendering all the objects on the map visualizations. The style information needed includes icon picture and label size, and the data information includes label value and location (Lat, Long).

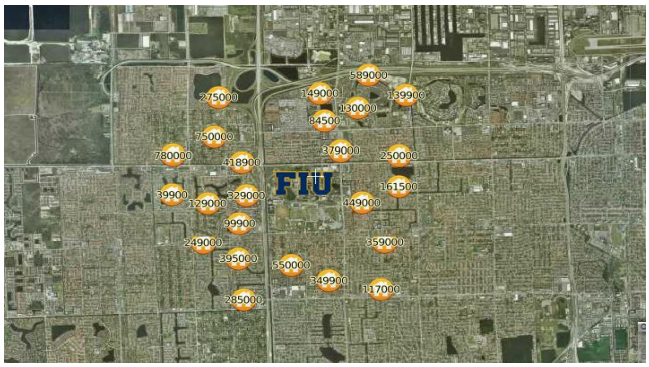


Figure 11: Query data near the point

Figure 11 shows the result of this query. Please be noticed that the unit of the distance function in all the demos is Lat-Long.

4.3.2 Other Samples

Figure 12 shows all the hotels along a certain street within a certain distance and also displays the different stars of the hotels. The MapQL statement for this query is listed below:

```
SELECT
    CASE
        WHEN star >= 1 and star < 2 THEN '/var/www/cgi-bin/hotel_1star.png'
        WHEN star >= 2 and star < 3 THEN '/var/www/cgi-bin/hotel_2stars.png'
        WHEN star >= 3 and star < 4 THEN '/var/www/cgi-bin/hotel 3stars.png'
        WHEN star >= 4 and star < 5 THEN '/var/www/cgi-bin/hotel 2stars.png'
        WHEN star >= 5 THEN '/var/www/cgi-bin/hotel 2stars.png'
        ELSE '/var/www/cgi-bin/hotel 0star.png'
    END AS T ICON PATH,
h.geo AS GEO
FROM
osm_fl o
LEFT JOIN
hotel_all h
ON
ST Distance(o.geo, h.geo) < 0.05
WHERE
    o.name = 'Florida Turnpike';
```

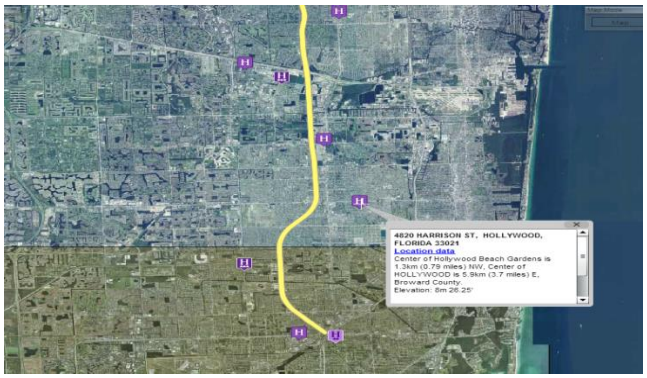


Figure 12: Query data along the line

Figure 13 shows the traffic of Santiago where the colder the color is, the faster the traffic is, the warmer the color is, and the worse the traffic is. The MapQL statement is listed below:

```
SELECT
    CASE
        WHEN speed >= 50 THEN 'color(155, 188, 255)'
        WHEN speed >= 40 and speed < 50 THEN 'color(233, 236, 255)'
        WHEN speed >= 30 and speed < 40 THEN 'color(255, 225, 198)'
        WHEN speed >= 20 and speed < 30 THEN 'color(255, 189, 111)'
        WHEN speed >= 10 and speed < 20 THEN 'color(255, 146, 29)'
        WHEN speed >= 5 and speed < 10 THEN 'color(255, 69, 0)'
        WHEN speed >= 0 and speed < 5 THEN 'color("red")'
else 'color("grey")'
    END AS T FILLED COLOR,
    '3' AS T THICKNESS,
GEO
FROM santiago traffic;
```

Figure 13: Traffic of Santiago

Figure 14 shows the different average incomes with in different zip codes. In this demo, users can customize the color and style of the map layers, different color stand for different average incomes. And the MapQL statement is listed below:

```
SELECT
u.geo AS GEO,
u.zip AS T LABEL,
    '0.7' AS T OPACITY,
    '15' AS T LABEL SIZE,
'color("blue")' AS T BORDER COLOR,
    CASE
        WHEN avg(i.income) < 30000 THEN 'color(155, 188, 255)'
        WHEN avg(i.income) >= 30000 and avg(i.income) < 50000 THEN 'color(233,
236, 255)'
        WHEN avg(i.income) >= 50000 and avg(i.income) < 70000 THEN 'color(255,
225, 198)'
        WHEN avg(i.income) >= 70000 and avg(i.income) < 90000 THEN 'color(255,
189, 111)'
        WHEN avg(i.income) >= 90000 and avg(i.income) < 110000 THEN 'color(255,
146, 29)'
        WHEN avg(i.income) >= 110000 and avg(i.income) < 130000 THEN 'color(255,
69, 0)'
        WHEN avg(i.income) >= 130000 THEN 'color("red")'
else 'color("grey")'
    END AS T FILLED COLOR
FROM
us_zip u left join income i
ON
ST_Within(i.geo, u.geo)='t'
GROUP BY
u.geo, u.zip;
```

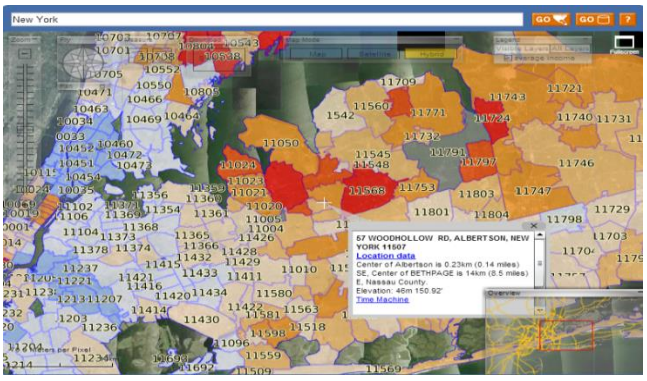


Figure 14: Income at New York

All these examples demonstrate that in TerraFly GeoCloud, users can easily create different map applications using simple SQL-like statements.

5. A Case Study

In this section, we present a case study on using TerraFly GeoCloud for spatial data analysis and visualization. As discussed in 4.2.1, we know the results of auto correlation can be shown in a scatter diagram, where the first and third quadrants of the plot represent positive associations, while the second and fourth quadrants represent negative associations. The second quadrant stands for low-high which means the value of the object is low and the values of surrounding objects are high.

A lay user whose name is Erik who has some knowledge about the database and data analysis wanted to invest a house property in Miami with a good appreciation potential. By using TerraFly GeoCloud, he may obtain some ideas about where to buy. He believes that if a property itself has low price and the surrounding properties have higher values, then the property may have good appreciation potential, and is a good choice for investment. He wants to first identify such properties and then do a field trip with his friends and the realtor agent.

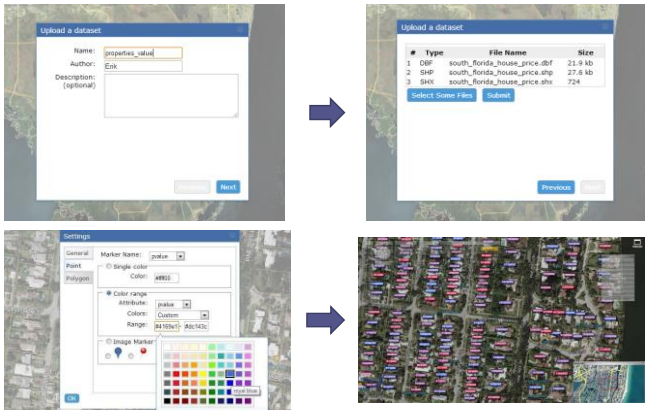


Figure 15: Data Set Upload and Visualization

To perform the task, first, Erik checked the average property prices by zip code in Miami which is shown in Figure 6. He found the *green circled area* in the low-high quadrants, which means that the average price of properties of this area is lower than the surrounding areas. Then, Erik wanted to obtain more insights on the property price in this area. He uploaded a detailed spatial data set named as *south_florida_house_price* into the TerraFly GeoCloud system as shown in Figure 15. He customized the label color range as the properties price changes. And then, he chose different areas in the *green circled area* in Figure 6 to perform the auto-correlation analysis.

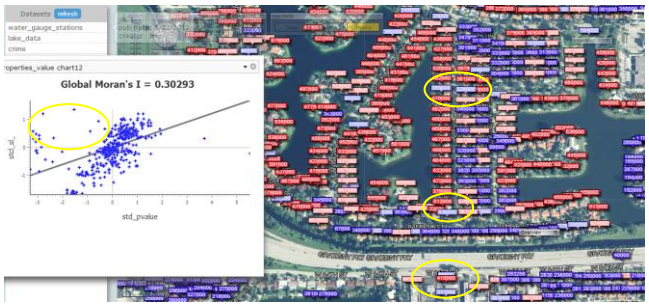


Figure 16: Properties in Miami

Finally, he found an area shown in Figure 16, where there are some good properties in the low-high quadrants (in yellow circles) with good locations. And one interesting observation is, lots of properties along the road *Gratigny Pkwy* has lower prices. He was then very excited and wanted to do a query to find all the cheap properties with good appreciation potential along the *Gratigny Pkwy*. Erik composed the MapQL statements like:

```
SELECT
    CASE
        WHEN h.pvalue >= 400000 THEN '/var/www/cgi-bin/redhouse.png'
        WHEN h.pvalue >= 200000 and h.pvalue < 400000 THEN '/var/www/cgi-
bin/bluehouse.png'
        WHEN h.pvalue >= 100000 and h.pvalue < 200000 THEN '/var/www/cgi-
bin/greenhouse.png'
```

```
        ELSE '/var/www/cgi-bin/darkhouse.png'
    END AS T ICON PATH,
h.geo AS GEO
FROM
osm fl o
LEFT JOIN
south_florida_house_price h
ON
ST Distance(o.geo, h.geo) < 0.05
WHERE
    o.name = 'Gratigny Pkwy' AND
    h.std pvalue<0 AND
    h.std sl pvalue>0;
```

Figure 17: MapQL results

The Figure 17 presents the final results of the MapQL statements. Finally, Erik sent the URL of the map visualization out by email, waiting for the response of his friends and the realtor agent.

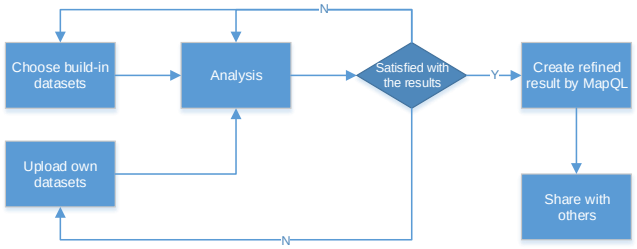


Figure 18: The flow path of Erik case

Figure 18 illustrates the whole workflow of the case study. In summary, Erik first viewed the system build-in datasets, conducted the data analysis, and then he identified properties of interest. He then composed MapQL statements to create his own map visualization to share with his friends. The case study demonstrates that TerraFly GeoCloud supports the integration of spatial data analysis and visualization and also offers user-friendly mechanisms for customized map visualization.

6. Related work and products

In the geospatial discipline, web-based GIS services can significantly reduce the data volume and required computing resources at the end-user side [16][17]. To the best of our knowledge, TerraFly GeoCloud is one of the first systems to study the integration of online visualization of spatial data, data analysis modules and visualization customization language.

Various GIS analysis tools are developed and visualization customization languages have been studied in the literature. ArcGIS is a complete, cloud-based, collaborative content management system for working with geographic information. But systems like ArcGIS and Geoda focus on the content management and share, not online analysis [18][19]. Azavea has many functions such as optimal Location find, Crime analysis, data aggregation and visualization. It is good at visualization, but has very limited analysis functions [20].

Various types of solutions have been studied in the literature to address the problem of visualization of spatial analysis [19]. However, on one hand, good analysis visualization tools like Geoda and ArcGIS do not have online functions. To use them, users have to download and install the software tools, and download the datasets. On the other hand, good online GIS systems like Azavea, SKE, and GISCloud have limited analysis functions. Furthermore, none of above products provides a simple and convenient way like MapQL to let user create their own map visualization [21][22].

The related products are summarized in Table 1. Our work is complementary to the existing works and our system also integrates the data mining and visualization.

Table 1: GIS Visualization Products

Name	Website	Product features description	Comments
ArcGIS Online	http://www.arcgis.com	http://www.arcgis.com ArcGIS Online is a complete, cloud-based, collaborative content management system for working with geographic information.	No online Analysis, focus on the content management and share.
Azavea	http://www.azavea.com/products/	optimal Location find, Crime analsis, data aggregated and visualized	Good visualization. Very limited Analysis functions
SKE	http://www.skeinc.com/GeoPortal.html	Spatial data Viewer	Focus on the spatial data viewer.
GISCloud	http://www.giscloud.com	with few analysis (Buffer , Range , Area , Comparison , Hotspot , Coverage , Spatial Selection)	Very limited simple analysis.
GeoIQ	http://www.geoiq.com/ http://geocommons.com/	filtering, buffers, spatial aggregation and predictive	Focus on GIS, very good Visualization and interactive operation. Very limited and simple analysis: currently provide predictive(Pearsons Correlation).

7. CONCLUSIONS AND FUTURE WORK

Web map services become increasingly widely used for various commercial and personal purposes. GIS application needs to be able to easily analyze and visualize spatial data and satisfy the increasing demand of information sharing. This paper presents a solution, TerraFly GeoCloud, an online spatial data analysis and visualization system, to address the challenges. TerraFly GeoCloud is built upon the TerraFly Geo spatial database, to offer a convenient way to analyze geo spatial data, visualize the results, and share the result by a unique URL. Our system also allows users to customize their own spatial data visualization using a SQL-like MapQL language rather than writing codes with Map API.

In our future work, we will research and develop an extra layer between end users who have limit knowledge in writing SQL statements and the MapQL, a query composing interfaces for the MapQL statements, to facilitate lay users to create their own map visualizations. Also, we will improve the scale of TerraFly GeoCloud, conduct large-scale experiments and employ

distributed computing as additional mechanisms for optimizing the system. In addition, we will explore how to apply the principle of MapQL to other applications that share similar characteristics with web GIS services.

8. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Grant Nos. CNS-0821345, CNS-1126619, HRD-0833093, IIP-0829576, CNS-1057661, IIS-1052625, CNS-0959985, OISE-1157372, IIP-1237818, IIP-1330943, IIP-1230661, IIP-1026265, IIP-1058606, IIS-1213026, OISE-0730065, CCF-0938045, CNS-0747038, CNS-1018262, CCF-0937964. Includes material licensed by TerraFly (http://teraffly.com) and the NSF CAKE Center (http://cake.fiu.edu).

9. REFERENCES

[1] Rishe, N., Chen, S. C., Prabakar, N., Weiss, M. A., Sun, W., Selivonenko, A., & Davis-Chu, D. (2001, April). TerraFly: A high-performance web-based digital library system for spatial data access. In The 17th IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany (pp. 17-19).

[2] Rishe, N., Sun, Y., Chekmasov, M., Selivonenko, A., & Graham, S. (2004, December). System architecture for 3D terrafly online GIS. In Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on (pp. 273-276). IEEE.

[3] Rishe, N., Gutierrez, M., Selivonenko, A., & Graham, S. (2005). TerraFly: A tool for visualizing and dispensing geospatial data. Imaging Notes, 20(2), 22-23.

[4] Spence, R., & Press, A. (2000). Information visualization.

[5] Old, L. J. (2002, July). Information Cartography: Using GIS for visualizing non-spatial data. In Proceedings, ESRI International Users' Conference, San Diego, CA.

[6] Yi Zhang and Tao Li. DClusterE: A Framework for Evaluating and Understanding Document Clustering Using Visualization. ACM Transactions on Intelligent Systems and Technology, 3(2):24, 2012.

[7] Teng, W., Rishe, N., & Rui, H. (2006, May). Enhancing access and use of NASA satellite data via TerraFly. In Proceedings of the ASPRS 2006 Annual Conference.

[8] Wang, H. (2011). A Large-scale Dynamic Vector and Raster Data Visualization Geographic Information System Based on Parallel Map Tiling.

[9] De Knegt, H. J., Van Langevelde, F., Coughenour, M. B., Skidmore, A. K., De Boer, W. F., Heitkönig, I. M. A., ... &Prins, H. H. T. (2010). Spatial autocorrelation and the scaling of species-environment relationships. Ecology, 91(8), 2455-2465.

[10] Li,Hongfei; Calder, Catherine A, "Beyond Moran's I: Testing for Spatial Dependence Based on the Spatial Autoregressive Model". Geographical AnalysisCressie, Noel (2007).

[11] Anselin, L. (1995). Local indicators of spatial association—LISA. Geographical analysis, 27(2), 93-115.

[12] Ester, M., Kriegel, H. P., Sander, J., &Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. ACM SIGKDD.

[13] Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery, 2(2), 169-194.

[14] Stein, M. L. (1999). Interpolation of spatial data: some theory for kriging. Springer Verlag.

[15] Bilodeau, M. L., Meyer, F., & Schmitt, M. (Eds.). (2005). Space: Contributions in Honor of Georges Matheron in the Fields of Geostatistics, Random Sets, and Mathematical Morphology (Vol. 183). Springer Science+ Business Media.

[16] Xiaoyan Li, Sharing geoscience algorithms in a Web service-oriented environment, Computers & Geosciences Volume 36, Issue 8, August 2010

[17] Fotheringham, S., & Rogerson, P. (Eds.). (2004). Spatial analysis and GIS. CRC Press.

[18] Johnston, K., Ver Hoef, J. M., Krivoruchko, K., & Lucas, N. (2001). Using ArcGIS geostatistical analyst (Vol. 380). Redlands: Esri.

[19] Anselin, L., Syabri, I., & Kho, Y. (2006). GeoDa: An introduction to spatial data analysis. Geographical analysis, 38(1), 5-22.

[20] Boyer, D., Cheetham, R., & Johnson, M. L. (2011). Using GIS to Manage Philadelphia's Archival Photographs. American Archivist, 74(2), 652-663.

[21] Hearnshaw, H. M., & Unwin, D. J. (1994). Visualization in geographical information systems. John Wiley & Sons Ltd.

[22] Boyer, D. (2010). From internet to iPhone: providing mobile geographic access to Philadelphia's historic photographs and other special collections. The Reference Librarian, 52(1-2), 47-56.

Randomly Sampling Maximal Itemsets

Sandy Moens
University of Antwerp, Belgium
sandy.moens@ua.ac.be

Bart Goethals
University of Antwerp, Belgium
bart.goethals@ua.ac.be

ABSTRACT

Pattern mining techniques generally enumerate lots of uninteresting and redundant patterns. To obtain less redundant collections, techniques exist that give condensed representations of these collections. However, the proposed techniques often rely on complete enumeration of the pattern space, which can be prohibitive in terms of time and memory. Sampling can be used to filter the output space of patterns *without* explicit enumeration. We propose a framework for random sampling of maximal itemsets from transactional databases. The presented framework can use any monotonically decreasing measure as interestingness criteria for this purpose. Moreover, we use an approximation measure to guide the search for maximal sets to different parts of the output space. We show in our experiments that the method can rapidly generate small collections of patterns with good quality. The sampling framework has been implemented in the interactive visual data mining tool called MIME[1], as such enabling users to quickly sample a collection of patterns and analyze the results.

Keywords

Maximal Itemsets, Random Walks, Output Space Sampling

1. INTRODUCTION

During the last decade an increase in data being generated and stored has taken place. An important reason for this trend is the relatively low cost of storing large amounts of data. The collected data contains a lot of useful information, which is evident because it is drawn from the real world. Therefore, extracting knowledge from the stored data has shown to be very important and interesting. The major problem, however, is that the really interesting and useful information is typically hidden in the vast volume of data.

Pattern mining techniques are able to extract *potentially* interesting knowledge from data by considering frequently occurring events as interesting. Many techniques are invented based on this idea in combination with the principle of monotonicity [2, 21, 14]. The core functionality of these methods is to give a complete set of patterns that fulfill a given quality constraint. However, the problems with these approaches are that a lot of redundancy is found, and, more importantly, many of the mined patterns are actually not interesting. In fact, the enduser inherently owns the subjective criteria to distinguish between truly interesting information, indeed, every user has it's own background knowledge of the data at hand.

Methods to reduce the amount of patterns and redundancy resulted in smaller collections of patterns, e.g. closed sets [19], maximal sets [5], non-derivable itemsets [8], etc. Unfortunately they could only partially alleviate the problem of pattern explosion and redundancy. A second downside of these approaches is that they generally rely on the enumeration of large parts of the pattern space. Only in later stages the less interesting patterns are pruned from the result. One has to realize, however, that enumerating the pattern space can actually in itself already be infeasible. To decrease redundancy in pattern collections even more, pattern set mining algorithms became increasingly important [1, 20, 17]. The goal of **pattern set mining** techniques is to find a small collection of patterns that are interesting *together*, rather than on their own. For instance, together they describe 90% of the original dataset.

On the other hand the iterative and interactive nature of data mining was already known from the start. One reason, thereto, is that typically pattern mining techniques deal with user-defined thresholds that are hard to set, making it into an iterative process to find a suitable parameter setting. Another reason is that interestingness is in essence subjective to a specific user. What is interesting for one person is by definition not always interesting for another person. Therefore, trying to define objective measures that capture the users' interest is perhaps impossible. In fact, in an ideal setting a user is interactively taking part in the **pattern exploration** process, combining her proper knowledge with objective functions that can be computed in an instant.

We differentiate the following building block for exploratory data analysis: interactivity, instant results and adaptability. First of all, exploration can only be done using a good computer-human interaction in a graphical interface. Secondly, instant results are necessary to allow a user to get immediate feedback on each request. This way she can continue the exploration task, without losing her focus. At last, we believe that results should be easily interpretable

[1]MIME is available at http://adrem.ua.ac.be/mime

IDEA'13, August 11th, 2013, Chicago, IL, USA.

and adaptable, giving the user control over the final pattern collection. Using good visualisation techniques a user should be able to understand the results better, faster and more easily [15].

The focus of our work is obtaining small collections of **maximal patterns** with low frequency that describe the data without much overlap and without explicit enumeration of the pattern space. Maximal patterns can for example be used in recommendation systems, where long patterns describe trends. For instance, a list of movies that many people like, can be used as recommendations for other users that share a large overlap of movies with the former list. In this paper we develop a sampling method that samples maximal itemsets based on monotone measures and reports near instant results. Moreover, we implemented our methods in MIME, a visual framework for data analysis developed by Goethals et al. [13]. The integration of the sampling method in a graphical framework enables users to easily explore, analyze and adapt the patterns that are reported by our sampling method. Therefore a user can quickly try out different settings and explore the data in search of her pattern collection of interest.

The following section gives an overview of the related work. In section 3 our sampling method is described and section 4 describes the framework in combination with MIME. Section 5 reports the experimental results of our method on various datasets. The last section concludes this work and discusses future work.

2. RELATED WORK

Numerous methods have been proposed to mine interesting patterns from transactional databases [1, 2, 5, 8, 21]. Some methods try to enumerate all itemsets satisfying a frequency threshold, while others try to obtain a more condensed representation, e.g. maximal itemsets. One problem with these methods is generally the overwhelming number of patterns obtained, making analysis in fact harder. Another problem is that often these techniques have to enumerate all subsets of the output patterns, making them prohibitive in practice. Not many techniques exist to sample the output space of frequent patterns *without* the enumeration step.

In work by Zhu et al. [22], the authors introduce the concept of Pattern Fusion to obtain a small collection of maximal frequent itemsets. This new mining strategy finds maximal itemsets by agglomerating small core patterns into large patterns, as such taking larger jumps in the lattice tree compared to typical sequential algorithms, e.g. Apriori [2] and Eclat [21]. The main idea behind this approach is that large maximal itemsets contain a lot of core patterns with similar characteristics. As a consequence randomly sampling from a pool of core patterns then merging them, results into large maximal itemsets.

Also for itemsets, Boley et al. [6] developed a general sampling framework for generating small collections of patterns from a user-defined distribution. In this work a user constructs a distribution over the complete pattern space by defining singleton preference weights and a multiplicative or additive evaluation scheme. Weights are then used in a Metropolis-Hastings sampling framework with linear time and space requirements. Their method does not materialize the complete pattern space.

The rest of this overview on sampling patterns deals with graph databases. Two methods for randomly sampling maximal subgraphs where proposed by Chaoji et al. [9] and Hasan and Zaki [3].

Chaoji et al. [9] use the same technique as our approach to sample maximal subgraphs. They utilize a two-step approach for finding orthogonal maximal frequent subgraphs. In a first step, a random walk is used to randomly sample maximal graphs. The second step then prunes the output to a smaller set of orthogonal patterns. In their work the random walk only traverses the lattice tree downwards. I.e. for a given graph, a random node can only be extended and nodes are never deleted from the current graph. The proposed method does not guarantee uniformity of the output collection of patterns in step 1, rather a biased traversal is used to simulate uniformity. In contrast, our work presents a framework for sampling maximal patterns using an evaluation and an approximation measure. The latter is used to guide the search to different parts of the lattice.

Hasan and Zaki [3] restrict themselves to uniformly sampling k-maximal subgraphs for a given threshold. They use Metropolis-Hastings and construct a probability matrix that enforces uniformity. Therefore, in each step all neighboring nodes (i.e. subgraphs) have to be checked for maximality. If the neighboring node is maximal, the probability of going to that state is proportional to the inverse of its degree.

Hasan and Zaki [4] proposed a method for sampling frequent subgraphs. A random walk over the partially ordered graph of frequent patterns is used to sample according to a prior interestingness distribution. Their work is similar to ours, in the sense that they also use a branch-and-bound technique to walk over the patterns. The differences are twofold: the output space is different and the underlying random walk is also different. Hasan and Zaki use Metropolis-Hastings as chain simulation method to sample according to the desired distribution. In our method a simple random walk over the remaining subsets is used.

The graphical pattern mining framework MIME [13] has been used as a base for implementing and testing our sampling technique. MIME is equipped with an intuitive user-interface and many interestingness measures to analyze the mined patterns. Moreover, it lets users quickly run various data mining algorithms using different parameter settings.

3. RANDOM MAXIMAL ITEMSETS

3.1 Preliminaries

We denote by $\mathcal{D}$ a dataset containing transactions t_j. Each transaction t_j is a subset over a finite set of items $\mathcal{I} = \{i_1, \ldots, i_n\}$. Individual items are also called singletons and will be used interchangeably in the rest of the paper. An itemset X is also a subset over the set of items $\mathcal{I}$. The cover of an itemset is the set of transactions that contain the itemset, i.e. $cov(X) = \{t_j | X \subseteq t_j\}$. The support of an itemset is defined as the size of its cover, i.e. the number of transactions that contain the itemset. A quality function q is an monotonically decreasing function that gives the interestingness of an itemset in the corresponding dataset $q : X \to \mathbb{R}$ (e.g. support). p is a monotonic function that is used as approximation for the quality function.

σ is a minimum quality threshold such that iff $q(X) >= \sigma$ then X is interesting. The negative border are the itemsets that are not interesting wrt σ and q, but whose subsets are all interesting. The positive border, is the set of itemsets

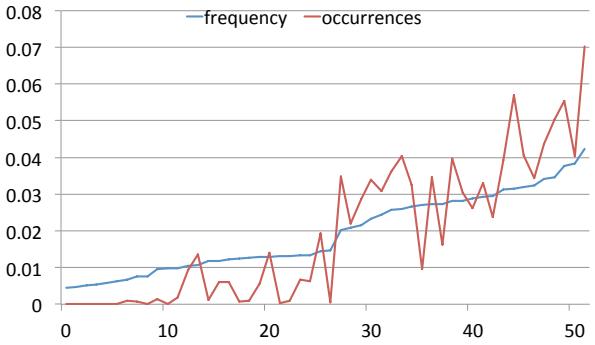


Figure 1: Singleton frequency vs singleton occurrences in maximal sets for mammals datasets with minimum support of 10%

that are interesting, but whose supersets are not. The positive border is also called the set of maximal itemsets.

3.2 Uniform Sampling

In this section we give an overview of two algorithms for uniform sampling of maximal sets.

A first, and naïve, way is to first generate the complete set of maximal sets, then performing sampling on the result. This way each maximal set has an equal, uniform probability of being drawn. The downside of this method, however, is that generating all maximal itemsets is both time and memory consuming. I.e. we have to enumerate almost the complete set of frequent patterns to obtain the collection of maximal frequent patterns. Furthermore, we have to rely on subset checking or other techniques [7] to quickly check if a potential maximal set has a superset in the result already.

A second approach is proposed by Al Hasan and Zaki on graphs [3] and can be adopted to itemsets. For this method we use Metropolis-Hastings to simulate a random walk over the complete pattern space. In each step, the current node represents the current itemset. The connected nodes (i.e. possible transition states) are the subsets with length that is one smaller and the supersets with length that is one larger than the current set. To obtain uniform samples we have to create a doubly stochastic probability matrix P which is used to propose new transitions. An easy way to do so is using the following probabilities [3]:

$$P(u,v) = \begin{cases} \frac{1}{Z} & \text{if } u, v \text{ non max} \\ \frac{1/d_v}{Z} & \text{if } u \text{ non-max, } v \text{ max} \\ \frac{1/d_u}{Z} & \text{if } u \text{ max } v \text{ non-max} \\ 0 & \textit{otherwise}, \end{cases} \quad (1)$$

with d_x the degree of x and, Z a normalization factor over all neighboring nodes. Note that using these weights, we always have to check each neighboring superset for maximality.

Sampling according to the described MH-chain results in a uniform sample of maximal patterns. The chain, however, has to be restarted from a random state for each sample and has to be run for a large number of iterations until a sample is returned. Generally, MCMC methods take a long number of iterations before converging to the stationary distribution, this is called the mixing time. When a chain converges fast to the desired distribution, the chain is called rapidly mixing, otherwise it is slowly mixing.

Algorithm 1 Random Sampling

Require: set of singletons $\mathcal{I}$, approximation function p
quality function q, quality threshold σ
Returns: random maximal itemset R

1: $R \leftarrow \{\}$
2: $C \leftarrow \{c_j \in I : q(c_j) \geq \sigma\}$
3: **while** $|C| > 0$ **do**
4: $M \leftarrow (c_i, p(R \cup c_i))$ for all $c_i \in C$
5: $x \leftarrow$ sample from M
6: $R \leftarrow \{R \cup x\}$
7: $C \leftarrow \{c_j \in C : q(R \cup c_j) \geq \sigma\}$
8: **return** R

3.3 Random Sampling

Randomly generating maximal sets can be done in a faster and easier way [9]. The only downside is that uniformity of the method is not guaranteed.

We randomly generate maximal itemsets by performing a simple random walk over all singleton items $\{i_1, \ldots, i_n\}$. We walk over the space of frequent patterns by starting at the empty set and gruadually appending new singletons. Initially each item is given a probability proportional to it's approximation score $p(i_j)$, favoring highly ranked items in the result. Then, for each iteration of the walk we prune the items that result in non-interesting itemsets according to q. The remaining items (also called extensions) are evaluated with respect to the current set and the approximation measure p, i.e. we evaluate $p(X \cup i_k)$ for all remaining i_k. This process continues until no more items can be added such that the set remains interesting. If no new items can be added a maximal itemset has been found.

The basic intuition behind this sampling approach is that higher valued items occur more frequently in the set of interesting maximal sets. This is under the assumption that the approximation function simulates the original quality function. Figure 1 shows an example for the mammals dataset when using frequency as quality and approximation measure. The minimum support threshold used is 10%. The blue line shows the relative support of singletons wrt the cumulative support of all singletons. The red line gives the number of occurrences of an item in the set of maximal frequent itemsets. The relation between both curves is not exact, but we clearly see that high support items occur more often in the set of maximal patterns than low support items.

The sampling procedure is outlined in Algorithm 1. Lines 1 and 2 are the initialization part where singletons that do not meet the quality threshold are discarded. In line 4 and 5 we perform biased sampling using p as bias for all remaining extensions. Lines 6 and 7 make sure that all non interesting supersets are discarded from the set of new samples. This simple random walk over the pattern space can be repeated any number of times.

3.4 Downgrading

Our method tries to sample small collections of maximal patterns that describe different parts of the data. As such, we want to find patterns that have a small overlap in the data described by them. An easy way to do so is using a weighting (or discounting) scheme in the sampling procedure: by giving less weight to patterns that are already covered by previously found patterns, we force the method

to come up with different results [16]. Essentially, we construct a distribution over a set of events that dynamically changes over time.

In our method we build a probability map based on possible extensions for the current set. To guide the pattern search process, we can downgrade the probabilities of the extensions (i.e. singletons) that have already been reported in previous sets, using weighting schemes. One downgrading scheme is proposed together with two others that have been adopted from Lavrač et al. [16]:

- **multiplicative weights** [16] decrease exponentially by the number of times a singleton has already been covered. Given a parameter $0 \leq \gamma \leq 1$, we bias singleton weights using the formula $w(i_j, k) = \gamma^k$, where k is the number of times a singleton i_j occurs already in the result.

- **additive weights** [16] can be implemented using the weight assignment $w(i_j, k) = \frac{1}{k+1}$. Using the additive weighting scheme the decrease is less drastical, but also less effective in settings where we want to sample only a small collection of patterns.

- **adaptive weights** is an adaptive scheme favoring singletons that have not been sampled for a long time. The formula is as follows: $w(i_j, k) = \left(1 - \frac{k}{tot}\right)$, where tot is the total number of patterns discounted so far.

Downgrading should be implemented with a precaution however. We still have to guarantee maximality of the patterns being found. As an example suppose multiplicative weights are used with $\gamma = 0$. In this case any item that has been found in just one sample will be given a weight of 0. Using the constructed probability map the 0-weighted items would never be sampled. After a while, only 0-weight elements will be left in the set of possible extensions. An effective way to deal with this problem is to start sampling uniformly in cases where only 0-weighted extensions are left. Indeed, now there is no real favoring of one the items that is left. One could come up with several other methods to deal with this problem aswell.

3.5 Completeness

Our sampling method can use any function as quality and approximation function, with the only restriction that the function has to be monotonic. This monotonic property allows to prune non interesting itemsets during each iteration [2], without losing guarantee that the random walk generates a maximal pattern. In fact, our method can mine *any* of the existing maximal itemset for the given quality threshold. This is easy to see, in the end a maximal itemset is build up by a number of subsets. During the random walk a chain of subsets is sampled, each with probability $P(X'_i)$, where X'_i represents a subset of size i. Now if a maximal set can be generated by a set of distinct random walks R_{max}, then the probability of sampling a maximal itemset of length n is given by:

$$\begin{aligned} P(X_{max}) &= \frac{\sum_{r \in R_{max}} P(r)}{Z} \\ &= \frac{\sum_{r \in R_{max}} P(X'_1)P(X'_2)...P(X'_n)}{Z} \\ &= \frac{\sum_{r \in R_{max}} P(e_1)P(e_2|e_1)...P(e_n|e_1...e_{n-1})}{Z}, \end{aligned}$$

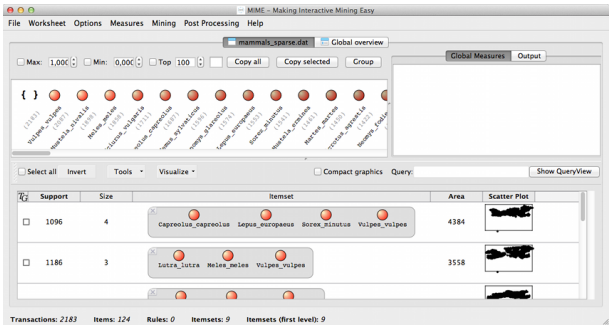


Figure 2: Interface of MIME

with Z a normalization factor over all maximal sets S_{max}. The first equality gives the probability of sampling a maximal set as the sum of probabilities of random walks that lead to this set. The second equality elaborates on individual chains and gives it's probability as the product of the intermediate steps. The last equation links probabilities of intermediate sets to the previously sampled subsets.

Since we know the probability of sampling a maximal itemset, we can give an estimate for the number of samples necessary to sample each distinct maximal itemsets using the generalized version of the Coupon Collector's problem [10]:

$$E(C_m) = \sum_{q=0}^{m-1} (-1)^{(m-1-q)} \sum_{|J|=q} \frac{1}{1 - P_J} \quad \text{with } m = |S_{max}|$$

$$P_J = \sum_{X \in S_{max}} P(X).$$

We would like to conclude this section with a small remark. To be completely correct, the approximation function does not have to be monotone. Rather it is possible to use non-monotonic approximation measures, however it is then impossible to see the influence in the final sample.

4. PATTERN EXPLORATION

The goal of our sampling method is to quickly generate small collections of maximal patterns that can easily be explored. Exploration is typically done in a visual framework with easy user interaction. Therefore, we implemented our framework into MIME, a user friendly pattern mining and visualization tool developed by Goethals et al. [13]. The main philosophy of MIME is to give users pattern exploration capabilities by providing a snappy interface with quick response times. Thereto, different caching and threading techniques are used.

In MIME a user is presented with the database in vertical/item format as shown in Figure 2. This is indeed the most intuitive way of looking at the database, since generally the user knows what an item represent. Furthermore, a user is able create patterns by clicking and using drag-and-drop operations. Alternatively she can use a lot of internal pattern mining algorithms to create an initial collection of patterns. Then she can evaluate patterns and/or pattern collections using a variety of interestingness measures [11]. Plugging our sampling method into MIME results in synergic behavior on both sides: (1) since the tool demands quick response times, the addition of this sampling framework provides good means to quickly obtain a small collec-

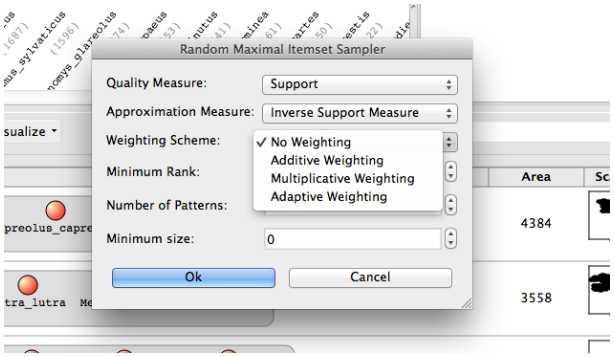


Figure 3: Sampling interface in MIME

tions of maximal patterns. (2) using the interactive interface of MIME we can easily try out different settings for the random maximal sampler and, more importantly, evaluate them immediately.

Figure 3 shows the graphical interface for sampling random maximal itemsets in MIME. Here a user is able to set all necessary information for sampling and also a minimum support size of the patterns. This is usefull when searching for long maximal itemsets with a size at least k. During the sampling process itemsets that are found pop-up one by one and the process can be stopped at any given time.

5. EXPERIMENTAL EVALUATION

5.1 Datasets

For our experiments, we used three datasets of different sizes. The main characteristics of the datasets are shown in Table 1.

The first dataset is *mammals* and is provided by T. Mitchell Jones[2] [18]. This dataset contains information on the habitat of animals in Europe. Each transaction corresponds to a 50 by 50 kilometers region and each item represents a mammal that lives somewhere in Europe. An item that is contained in a transaction corresponds to a mammal living in the specified region.

The other two datasets are publicly available from the FIMI repository[3]. The *pumsb* dataset provides census information about population and housing. The *accidents* dataset [12] contains anonymized information of traffic accidents on the Belgian roads.

$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	σ (%)	Max Sets
mammals	2.183	121	10	198.231
pumsb	49.046	2.113	55	92.221
accidents	340.183	468	5	551.073

Table 1: Characteristics of datasets

5.2 Evaluation Metrics

We evaluate our sampled pattern collections using the following metrics:

- **Size**: the size is the number of singletons contained by a pattern. It is important for our sampling method, because we mainly want to obtain long patterns. Shorter patterns are more general, but also provide less interesting information when viewed from a correlation perspective.

- **Jaccard**: the Jaccard index is a similarity measure defined as the size of the intersection of two sets relative to the size of the union: i.e. $Jaccard(X,Y) = \frac{X \cap Y}{X \cup Y}$. For our evaluation, we use the average pairwise Jaccard dissimilarity $(1 - Jaccard)$, between all pairs of patterns in the sampled collection. This gives an intuition on the overlap between two patterns in the collection of samples.

- **Duplicates**: the number of duplicates is interesting in a randomized sampling method, since obviously this is something we want to minimize.

- **Approximation Error**: this set evaluation measure was proposed by Zhu et al. [22] and evaluates a pattern collection as a whole. It measures the average edit distance of a set of patterns to a set of cluster centers. In our setting, the set of clusters is the sampled collection of patterns and the set of patterns is the complete set of maximal itemsets for the same support threshold. Then, for each maximal set we assign it to the closest center (i.e. an itemset from the sample set) using the edit distance $Edit(X,Y) = |X \cup Y| - |X \cap Y|$. This results in a clustering of the maximal sets with respect to our sampled collection. Given a sample collection P of size m, the approximation error is now defined as $\triangle(C_P^{S_{max}}) = \Sigma_{i=1}^{m} r_i/m$, with r_i the maximal edit distance between a member of the cluster and it's center. Lower approximation errors result in descriptive capabilities of the sampled collection wrt the total set of maximal itemsets.

- **Total Area**: another set evaluation measure is the total area covered by the collection. This entails the real coverage by the patterns in the data and not the cumulated sum of the areas of individual areas. Higher total area results in higher descriptive capabilities of the pattern collection wrt the original data.

5.3 Experimental Results

We test our approach to a regular top-k mining algorithm for finding maximal itemsets. We know that sequential algorithms walk over the pattern space such that similar itemsets are found close to each other. Using our sampling method, we find maximal itemsets that are spread over the complete set of maximal sets. All our methods have been implemented in Java and are available as standalone or in MIME [4].

5.3.1 Pattern Set Quality

We empirically tested sampling by instantiating our framework with different approximation measures and weighting schemes. As quality measure we used the support measure. For approximation measures we used a mixture of the frequency measure, the inverse frequency measure and an equality measure (i.e. all items are given equal weights).

[2]The mammals dataset can be obtained at http://www.european-mammals.org/

[3]The FIMI repository can be found at http://fimi.ua.ac.be/data/

[4]Implementations are available at: http://adrem.ua.ac.be/implementations

The weighting schemes used throughout the experiments are described in Section 3.4. Uniform sampling is simulated using the naïve approach as described in Section 3.2. At last, we also used consequtive chunks of maximal sets found by the Eclat algorithm with maximal itemsets pruning. As such, we try to simulate running Eclat with very short time budgets for finding maximal patterns.

For each of the methods we did 10-fold cross-validation, where collections of 10 and 20 itemsets are sampled from the pumsb dataset with a support threshold of 50%. To objectively measure the quality of the collections found, we compute the metrics defined in 5.2. Results of the experiments are shown in Table 2.

From Table 2 we can immediately see that chunking is not suitable for obtaining small sample collections to describe the data, i.e. the total area is very low compared to all other methods. Also, the Jaccard distance is very high, meaning that a lot patterns share a large overlap in the number of items. Uniform sampling is already better in terms of Jaccard distance and the approximation error, but lacks the descriptiveness of the original data. For all of the randomized methods the average Jaccard distance between any pair of samples is much lower than the chunking method and the total areas are generally much better than chunking and uniform sampling.

Interestingly, sampling with inverse frequency results in high approximation scores and, therefore, bad cluster centers. However, if we also look at the average size of the patterns found, we see they are the lowest for inverse frequency. In other words, the reason why the samples are bad cluster centers, is because they are too small compared to the average size of all maximal itemset. As such, the edit distance from any maximal set to its' cluster center will always be larger. The difference in size between frequency and inverse frequency can actually be explained intuitively. A set with high support has more extensions with high support. In turn these high support extensions have a higher probability of being sampled, resulting again in high support intermediate sets. This is in contrast to when the set already has a low support, then the extensions are also low on support, and, moreover, the lowest support extensions have highest probability of being sampled.

We analyze the weighting schemes again using Table 2, and we compare to the none weighted case for the same approximation measure. Generally speaking, the weighting schemes mildly influence the quality of the patterns for such small collections. Although no clear winner can be appointed, the additive scheme performs slightly better than the rest: the total area is always the highest and Jaccard the lowest for all measures.

In another experiment concerning the length of maximal itemsets, we sampled 100 times the number of maximal itemsets found in the mammals dataset when setting support to 437 (20%). The distribution of the maximal sets found wrt their length is given in Figure 4. The purple line gives the length distribution of the actual set of maximal patterns. This plot shows similar results for the sizes compared to Table 2. Inverse frequency favors sampling of short patterns, while frequency favors sampling of long patterns. The equal measure is somewhere in the middle, which makes sense because in each iteration high and low support items share the same sampling probability.

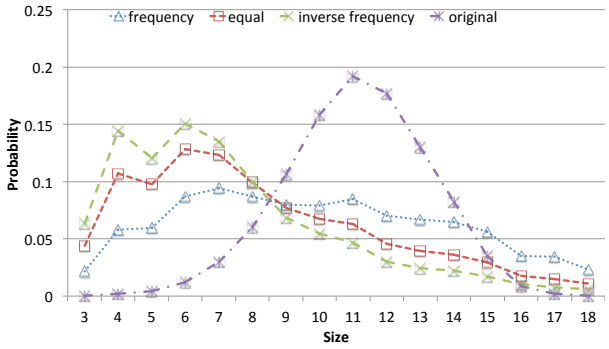


Figure 4: Distribution of length of maximal patterns

5.3.2 *Timing Results*

In the first timing experiment we sample small collections of patterns with low thresholds for the different datasets. We use frequency as quality/approximation measure during this experiment and set no weighting scheme, because weighting schemes do almost not influence the computation times. For the different tests we report the time needed to sample the collection of desired size as well as the number of non duplicate sets. The results are shown in Table 3.

Obviously, the time needed to sample X sets increases linearly by the number of sets to sample. Furthermore, we see that most of the experiments run in less than a few seconds, unless the threshold is set rather low (e.g. 1% for pumsb and accidents). On the other hand, from the experiments we see that sampling just 10 sets, takes less than a second. In an interactive setting this is very reasonable, since in fact we do not want to mine hundreds or thousands of patterns, rather we like to obtain small collections that we can analyze easily and fast. E.g. in MIME, if a user is not satisfied with the collection she can gradually ask for more patterns.

$\mathcal{D}$	σ(%)	Samples	Non-Dup	Time (s)
mammals	10 (0.5%)	10	10,0	0,007
		100	93,6	0,073
		1000	776,1	0,721
	21 (1%)	10	9,9	0,006
		100	96,6	0,055
		1000	864,5	0,551
pumsb	490 (1%)	10	10,0	0,46
		100	99,1	3,933
		1000	946,8	40,207
	26.975 (55%)	10	10,0	0,067
		100	95,6	0,265
		1000	816,1	2,545
accidents	3.401 (1%)	10	10,0	0,624
		100	100,0	4,905
		1000	999,5	48,133
	68.036 (20%)	10	10,0	0,226
		100	96,1	1,187
		1000	818,3	12,235

Table 3: Time results for sampling method using frequency as quality and approximation measure, and without weighting

In a second experiment we measure the time needed for our sampling method to obtain the complete set of maximal patterns. We would like to remind the reader that this is in fact not the main use of our sampling approach.

$\mathcal{D}$	σ (%)	Samples	Method	Weighting Scheme	Size	Jaccard	Ap. Err.	Duplicates	Total Area
pumsb	50	10	chunking	-	14,67	0,73	1,54	0,0	494.323
			uniform	-	12,83	0,28	1,38	0,0	1.164.491
			sampling_{freq}	-	13,77	0,26	1,39	0,0	1.424.332
				additive	13,00	0,20	1,50	0,0	1.507.440
				multiplicative	14,77	0,29	1,27	0,0	1.386.859
				adaptive	13,54	0,22	1,47	0,1	1.470.019
			$\text{sampling}_{invfreq}$	-	10,55	0,19	1,86	0,2	1.278.312
				additive	10,40	0,14	2,06	0,1	1.375.827
				multiplicative	11,40	0,21	1,86	0,0	1.269.745
				adaptive	10,92	0,18	1,87	0,2	1.314.444
			sampling_{equal}	-	12,91	0,22	1,53	0,0	1.410.602
				additive	11,02	0,16	1,87	0,1	1.469.940
				multiplicative	13,28	0,24	1,55	0,0	1.426.970
				adaptive	11,66	0,18	1,70	00	1.405.476
		20	chunking	-	12,45	0,65	1,63	0,0	495.525
			uniform	-	12,94	0,28	1,21	0,0	1.335.700
			sampling_{freq}	-	13,41	0,25	1,32	0,1	1.712.057
				additive	11,68	0,18	1,52	0,0	1.776.483
				multiplicative	13,22	0,25	1,30	0,1	1.683.907
				adaptive	12,85	0,21	1,30	0,1	1.728.926
			$\text{sampling}_{invfreq}$	-	11,11	0,20	1,65	0,2	1.595.981
				additive	10,02	0,15	1,79	0,1	1.766.286
				multiplicative	10,47	0,19	1,81	0,3	1.597.760
				adaptive	10,68	0,18	1,63	0,4	1.654.833
			sampling_{equal}	-	13,22	0,25	1,34	0,2	1.670.994
				additive	11,22	0,17	1,55	0,1	1.790.034
				multiplicative	13,33	0,24	1,31	0,2	1.703.623
				adaptive	12,09	0,19	1,42	0,1	1.727.109

Table 2: Overview of metrics computed on sampled collections from the pumsb dataset

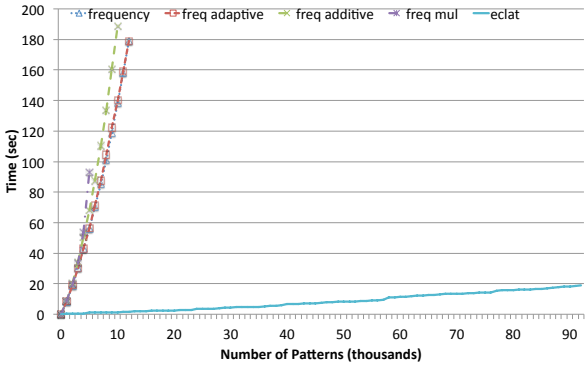


Figure 5: Detailed view of time results for finding distinct maximal sets using sampling

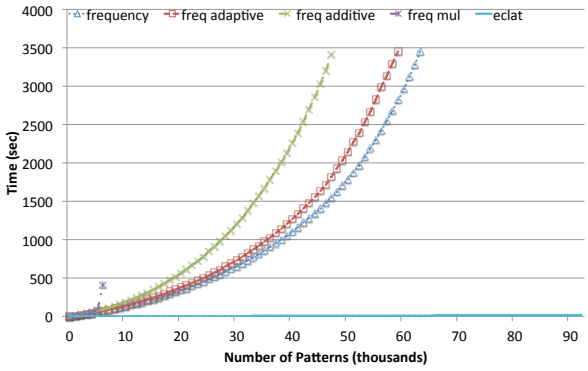


Figure 6: Time results for finding distinct maximal sets using sampling with a time budget of 1 hour

Figure 5 and 6 present time results for finding distinct patterns using different approximation measures. The dataset used for this experiment is the pumsb dataset with a support threshold of 55%. For Figure 5, we first mined the dataset for maximal itemsets using our proper Eclat implementation using bitsets. We used the optimizations described by Borgelt [7] to prune the output set for maximal itemsets. Then we sampled itemsets for 10 times the amount of time Eclat needs to mine all maximal sets. The results show that Eclat is much faster in this experiment, which is not suprising because the sampling methods have to create multiple distribution maps for each sample. In contrast, Eclat only has to make intersections of lists of transaction identifiers. Moreover, we only report the non-duplicate patterns in this timing experiment. We can also see that for the first 5K samples the sampling methods take approximately the same time. Later in the sampling process the discounting measures actually experience more difficulty in finding non-duplicate sets. The bigger picture is shown in Figure 6. For this chart we let the different methods run for 1 hour before stopping execution. Going to higher numbers of distinct patterns we observe an exponential increase in the running time. Indeed, more duplicates are found and our sampling technique has difficulties walking in non-traversed regions of the pattern space.

6. CONCLUSION

We studied the problem of randomly sampling maximal itemsets without explicit enumeration of the complete pattern space. For this purpose we employed a simple random walk that only allows additions of singletons to the current set untill a maximal set is found. The proposed frame-

work uses two types of monotone interestingness functions: a quality function that prunes the search space for maximal itemsets and an approximation measure that guides the search to patterns with different characteristics. Empirical studies have shown that our method can generate a small collection of maximal patterns fast, while preserving good quality. This method has been integrated into MIME, a framework for visual pattern mining and data exploration.

A more thorough study on the effect of approximation measures is still to be done. At last we still have to explore other quality and approximation measures that can be used in our sampling framework.

Acknowlegdements

We would like to thank T. Mitchell-Jones for providing us a copy of the mammals dataset.

7. REFERENCES

[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *Proc. ACM SIGKDD*, pages 12–19. ACM, 2004.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.

[3] M. Al Hasan and M. J. Zaki. Musk: Uniform sampling of k maximal patterns. In *SDM*, pages 650–661, 2009.

[4] M. Al Hasan and M. J. Zaki. Output space sampling for graph patterns. *Proc. VLDB Endow.*, pages 730–741, 2009.

[5] R. J. Bayardo, Jr. Efficiently mining long patterns from databases. *SIGMOD Rec.*, pages 85–93, 1998.

[6] M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *Proc. ACM SIGKDD*, pages 69–77. ACM, 2012.

[7] C. Borgelt. Efficient implementations of apriori and eclat. In *Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90*, 2003.

[8] T. Calders and B. Goethals. Non-derivable itemset mining. *Data Min. Knowl. Discov.*, pages 171–206, 2007.

[9] V. Chaoji, M. Al Hasan, S. Salem, J. Besson, and M. J. Zaki. Origami: A novel and effective approach for mining representative orthogonal graph patterns. *Stat. Anal. Data Min.*, pages 67–84, 2008.

[10] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Appl. Math.*, pages 207–229, 1992.

[11] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 2006.

[12] K. Geurts, G. Wets, T. Brijs, and K. Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board*, 2003.

[13] B. Goethals, S. Moens, and J. Vreeken. Mime: a framework for interactive visual pattern mining. In *Proc. ACM SIGKDD*, pages 757–760. ACM, 2011.

[14] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, pages 1–12, 2000.

[15] D. A. Keim. Information visualization and visual data mining. *IEEE Trans. on Vis. and Comp. Graph,*, pages 1–8, 2002.

[16] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *J. Mach. Learn. Res.*, pages 153–188, 2004.

[17] M. Mampaey, J. Vreeken, and N. Tatti. Summarizing data succinctly with the most informative itemsets. *ACM Trans. Knowl. Discov. Data*, pages 16:1–16:42, 2012.

[18] A. Mitchell-Jones. *The Atlas of European Mammals*. Poyser Natural History. T & AD Poyser, 1999.

[19] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. Int. Conf. on Data. Th.*, pages 398–416. Springer-Verlag, 1999.

[20] J. Vreeken, M. Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Min. Knowl. Discov.*, pages 169–214, 2011.

[21] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithms for discovery of association rules. *Data Min. Knowl. Discov.*, pages 343–373, 1997.

[22] F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng. Mining colossal frequent patterns by core pattern fusion. In *IEEE Int. Conf. on Data Eng.*, pages 706–715. IEEE, 2007.

Towards Anytime Active Learning: Interrupting Experts to Reduce Annotation Costs

Maria E. Ramirez-Loaiza
mramire8@hawk.iit.edu

Aron Culotta
aculotta@iit.edu

Mustafa Bilgic
mbilgic@iit.edu

Illinois Institute of Technology
10 W 31st
Chicago, IL 60616 USA

ABSTRACT

Many active learning methods use annotation cost or expert quality as part of their framework to select the best data for annotation. While these methods model expert quality, availability, or expertise, they have no direct influence on any of these elements. We present a novel framework built upon decision-theoretic active learning that allows the learner to directly control label quality by allocating a time budget to each annotation. We show that our method is able to improve performance efficiency of the active learner through an interruption mechanism trading off the induced error with the cost of annotation. Our simulation experiments on three document classification tasks show that some interruption is almost always better than none, but that the optimal interruption time varies by dataset.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Experimentation, Human Factors, Measurement, Performance

Keywords

Active learning, anytime algorithms, value of information, empirical evaluation

1. INTRODUCTION

Active learning [2] seeks to reduce the human effort required to train a classifier. This is typically done by optimizing which instances are annotated in order to maximize accuracy while minimizing the total cost of annotations. In this paper, we begin with the simple observation that in many domains, the expert/user forms an opinion about the class of an instance incrementally by continuously analyzing the instance. For example, in document classification, the expert forms an opinion about the topic of the document incrementally while reading the document. In tumor detection by CT-scan, a radiologist forms an opinion as s/he spends more and more time on the images. In intrusion detection, a security analyst must inspect various aspects of network activity to determine whether an attack has occurred.

We introduce a novel framework in which the active learner has the ability to interrupt the expert and ask his/her best guess so far. We refer to such a framework as *anytime active learning*, since the expert may be expected to return an annotation for an instance at any time during their inspection. For example, in document classification, we may show the expert only the first k words of a document and ask for the best guess at its label. We refer to a portion of an instance as a *subinstance*. Of course, the downside of this approach is that it can introduce annotation error — reading only the first k words may cause the annotator to select an incorrect label for the document. Assuming that both the cost to annotate an instance and the likelihood of receiving a correct label increase with the time the expert spends on an instance, the active learner has a choice on how to spend its budget: to collect either many but low-quality or few but high-quality annotations.

Our active learning framework thus models the tradeoff between the cost of annotating a subinstance (a function of its size) and the value of the (possibly incorrectly labeled) instance. At each iteration, the algorithm searches over subinstances to optimize this tradeoff — for example, to decide between asking the human expert to spend more time on the current document or move on to another document. We build upon the value of information theory [6], where the value of a subinstance is the expected reduction in the generalization error after the instance is added to the training set. The subinstance with the highest value cost difference is shown to the expert for annotation.

While previous work has considered the cost-benefit tradeoff of each instance [7] as well as annotation error [3], to our knowledge this is the first approach that allows the learning algorithm to directly control the annotation cost and quality of an instance by either interrupting the expert or revealing only a portion of an instance. Though closely-related, our framework differs from the missing feature-value acquisition problem [1, 10]; in our framework the feature values are not missing but the expert is interrupted.

We perform experiments on three document classification tasks to investigate the effectiveness of this approach. In particular, we provide answers to the following research questions:

RQ1. **Annotation Error:** Given that greater interruption can lead to greater annotation error, how do active learning algorithms perform in the presence of increasing amount of noise? We find that naïve Bayes consistently outperforms logistic regression and support vector machines as the amount of label noise increases, both in overall accuracy and in learning rate.

IDEA'13, August 11th, 2013, Chicago, IL, USA.

RQ2. **Cost-Quality Tradeoff:** Under what conditions is the cost saved by using subinstances worth the error introduced? How does this vary across datasets? We find that some interruption is almost always better than none, resulting in much faster learning rates as measured by the number of words an expert must read. For example, in one experiment, annotating based on only the first 10 words of a document achieves a classification accuracy after 5,000 words that is comparable to a traditional approach requiring 25,000 words. The precise value of this tradeoff is unsurprisingly data dependent.

RQ3. **Adaptive Subinstance Selection**: Does allowing the learning algorithm to select the subinstance size dynamically improve learning efficiency? We find that selecting the subinstance size dynamically is comparable to using a fixed size. One advantage of the dynamic approach is that formulating the size preference in terms of the cost of improving the model may provide a more intuitive way of setting model parameters.

The rest of the paper is organized as follows: Section 2 presents our active learning framework, including implementation details and experimental methodology; Section 3 presents our results and provides more detailed answers to the three research questions above; Section 4 briefly summarizes related work; and Sections 5 and 6 conclude and discuss future directions.

2. METHODOLOGY

In this section, we describe our methodology for the active learner that has the choice to interrupt the expert at any time. Let $\mathcal{L} = \{(x_1, y_1) \dots (x_n, y_n)\}$ be a set of tuples containing an instance x_i and its associated class label $y_i \in \{y^0, y^1\}$. (For ease of presentation, we assume binary classification.) Let $P_\mathcal{L}(y|x)$ be a classifier trained on $\mathcal{L}$. Let $\mathcal{U} = \{x_{n+1} \dots x_m\}$ be a set of unlabeled instances. Let $x^k \subseteq x$ be a subinstance representing the interruption of the expert at time k; or analogously the document containing the first k words in document x. For ease of discussion, we will use the document example in the remainder of the paper.

Let $Err(P_\mathcal{L})$ be defined as the expected loss of the classifier trained on $\mathcal{L}$. The value of information for x_i^k is defined as the reduction in the expected loss:

$$VOI(x_i^k) = Err(P_\mathcal{L}) - Err(P_{\mathcal{L} \cup (x_i, y_i)})$$

where $\mathcal{L} \cup (x_i, y_i)$ is the training set expanded with the label of x_i, which is provided by the expert through inspecting x_i^k. Because we do not know what label the expert will provide, we take an expectation over possible labelings of x_i^k:

$$VOI(x_i^k) = Err(P_\mathcal{L}) - \sum_{y_j} P_\mathcal{L}(y_j|x_i^k) Err(P_{\mathcal{L} \cup (x_i, y_j)}) \quad (1)$$

Note that even though the expert labels only subinstance x_i^k, we include the entire document x_i in our expanded set $\mathcal{L} \cup (x_i, y_j)$.

The decision-theoretic active learning strategy picks the subinstance that has the highest value cost difference:

$$\arg\max_{x_i^k \subseteq x_i \in \mathcal{U}} VOI(x_i^k) - \lambda C(x_i^k) \quad (2)$$

where $C(x_i^k)$ is the cost of labeling x_i^k and λ is a user-defined parameter that translates between generalization error and annotation cost. We explain the role of this parameter in detail in Section 2.3.

We next provide the details on how we define the error function $Err(.)$, the cost function $C(.)$, and the intuition for the parameter λ.

2.1 The Error Function $Err(.)$

We define the generalization error $Err(P_\mathcal{L})$ through a *loss function* $L(P_\mathcal{L}(y|x))$ defined on an instance:

$$\begin{aligned} Err(P_\mathcal{L}) &= \mathbb{E}\left[L(P_\mathcal{L}(y|x)\right] \\ &= \int_x L(P_\mathcal{L}(y|x)) P(x) \\ &\approx \frac{1}{|\mathcal{U}|} \sum_{x \in \mathcal{U}} L(P_\mathcal{L}(y|x)) \end{aligned}$$

A common loss function is 0/1 loss. However, because we do not know the true label of the instances in the unlabeled set $\mathcal{U}$, we need to use proxies for the loss function L. For example, a proxy for 0/1 loss is:

$$L(P_\mathcal{L}(y|x)) = 1 - \max_{y^j} P_\mathcal{L}(y^j|x) \quad (3)$$

One problem with this proxy in practice is that it trivially achieves 0 loss when all the instances are classified into one class with probability 1. We instead formulate another proxy, which we call *ranked-loss*. The intuition for ranked-loss is that, in practice a percentage of instances are expected to belong to class y^0 whereas the rest are classified as class y^1. Let $p \in [0, 1]$ be the proportion of instances with label y^0. Then, in $\mathcal{U}$, we expect $p \times |\mathcal{U}|$ of instances to have label y^0 and the remaining instances to have label y^1.

When we are computing the loss, we first rank the instances in $\mathcal{U}$ in descending order of $P_\mathcal{L}(y^0|x_i)$. Let this ranking be x_{r_1}, x_{r_2}, …, $x_{r_{|\mathcal{U}|}}$. Then, ranked-loss is defined as:

$$L_{RL}(P_\mathcal{L}(y|x_{r_i})) = \begin{cases} 1 - P_\mathcal{L}(y^0|x_{r_i}) & \text{if } i < |\mathcal{U}| \times p \\ 1 - P_\mathcal{L}(y^1|x_{r_i}) & \text{otherwise} \end{cases} \quad (4)$$

where p is the proportion of instances that are expected to be classified as class y^0. This formulation requires us to know p, which is known approximately in most domains. In this paper, we use $p = 0.5$ by default, though this could be estimated directly from the training data.

Note that when we use 0/1 loss proxy (Equation 3), the trivial solution of classifying all instances as y^0 with probability 1 achieves 0 error, whereas the ranked-loss for this trivial solution leads to an error of $1 - p$. We leave for future work an empirical comparison of alternative loss functions.

2.2 The Cost Function $C(.)$

The cost function $C(x_i^k)$ denotes how much the expert charges for annotating the subinstance x_i^k. In practice, this cost depends on a number of factors including intrinsic properties of x_i, the value of k, and what the expert has just annotated (the context-switch cost). To determine the true cost, user studies need to be conducted. In this paper, we make a simplifying assumption and assume that the cost depends simply on k. For documents, we can assume:

$$C(x_i^k) = k$$

or

$$C(x_i^k) = \log(k)$$

In this paper, we follow [7] and assume the annotation cost is a linear function of the instance length: $C(x_i^k) = k$.

2.3 The Parameter λ

The value of information for x_i^k is in terms of reduction in the generalization error whereas the annotation cost $C(x_i^k)$ is in terms

of time, money, etc. The parameter λ reflects how much/little the active learner is willing to pay per reduction in error.

Note that both 0/1 loss (Equation 3) and ranked-loss (Equation 4) range between 0 and 0.5, whereas the linear annotation cost is k. A λ value of 0.0001 means that for an x_i^{100} to be considered for annotation, $VOI(x_i^{100})$ has to be at least 0.01; that is, x_i^{100} has to reduce the error by an absolute amount of 0.01.

Typically, in the early iterations of active learning, improving the classifier is easier, and hence the range of VOI is larger compared to the later iterations of active learning. Therefore, the active learner is willing to pay less initially (because improvements are easy) but should be willing to pay more in later iterations of learning. Hence, a larger λ is preferred at the earlier iterations of active learning. Following this intuition, we define an adaptive λ that is a function of the current error of the classifier, $Err(P_{\mathcal{L}})$:

$$\lambda = Err(P_{\mathcal{L}}) \times \gamma$$

where γ is a fixed parameter denoting the desired percentage improvement on the current error of the model. For a fixed γ, λ is bigger initially because $Err(P_{\mathcal{L}})$ is larger initially, and as the model improves, $Err(P_{\mathcal{L}})$ goes down and so does λ.

2.4 The Effect of Annotation Error

As discussed above, selecting small subinstances (equivalently, interrupting the expert after only a short time) can introduce annotation error. While previous work has proposed ways to model annotation error based on the difficulty of x or the expertise of the teacher [14], here the error is introduced by the active learning strategy itself.

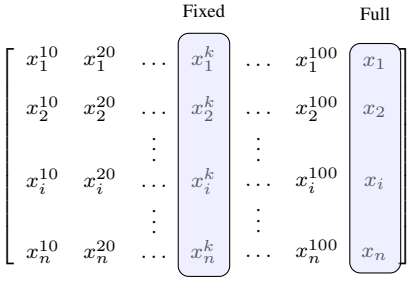
Rather than attempt to model this error directly, we observe that the objective function in Equation 2 already accounts for this error somewhat through the loss function. That is, if y^* is the true label of x_i, then we expect $Err(P_{\mathcal{L}\cup(x_i,y^*)}) < Err(P_{\mathcal{L}\cup(x_i,\neg y^*)})$. Note that the expanded training set includes the full instance, not simply the subinstance. This fact in part offsets inaccuracies in the term $P_{\mathcal{L}}(y_j|x_i^k)$ introduced by using subinstances. We study the empirical effect of this error in Section 3.

2.5 Implementation Details and Baselines

In this section, we provide details on how we implemented our objective function Equation 2 and define a few baseline approaches.

Let $\mathcal{C} = \{x_1 \dots x_s\}$ be a set of candidate instances for annotation, where $\mathcal{C} \subseteq \mathcal{U}$. We create a new set $\mathcal{C}'$ of subinstances x_i^k from $\mathcal{C}$. That is, our new search space includes x_i and all subinstances derived from it. We use this new space to apply our objective Equation 2.

We can illustrate the space defined by $\mathcal{C}'$ as a matrix where each row i allocates document x_i and all its derived subinstances x^k in ascending order of size. For simplicity, assume that k has increments of 10 words. The diagram illustrates the idea:



We use this matrix to illustrate how our algorithm works as well as several baselines. At each iteration, our algorithm searches this entire matrix and picks the best candidate according to the objective function Equation 2. We refer to this method as **QLT-γ** (where γ is the desired percentage of improvement).

Note that computing VOI for each candidate is computationally very expensive. We need to retrain our classifier for all possible labelings of each candidate. However, since a subinstance is only used for obtaining a label and we add x_i (full instance) to $\mathcal{L}$, therefore we only retrain our classifier once for each possible label of x_i. That is, our algorithm has the same computational complexity as any VOI-based strategy.

We define two baselines that ignore the cost of annotation. These approaches explore the candidates in $\mathcal{C}'$ by only searching at the fixed size column x_i^k and selecting the best from that column. These approaches are:

- **RND-FIX-k**, a baseline, uses random sampling over candidates of size k. That is, random sampling elements of column k.
- **EER-FIX-k**, a baseline that uses VOI over column k. Note that this is equivalent to expected error reduction (EER) approach (as described in [12]) on column k.
- **RND-FULL** and **EER-FULL**, baselines that use random sampling and expected error reduction (EER) respectively with full size instances. These approaches search on the last column of the matrix.

Notice that once a fragment x_i^k is selected as the best option for x_i all other fragments of that document (i.e. row i) will be ignored.

2.6 Experimental Evaluation

In this section, we first describe the datasets and then describe how we evaluated the aforementioned methods.

2.6.1 Datasets

We experimented with three real-world datasets for text classification with train and test partitions; details of these dataset are available in Table 1. Reuters known as Reuters-21578 collection [8] is a collection of documents from Reuters newswire in 1987. We created a binary partition using the two largest classes in the dataset, *earn* and *acq*. SRAA is a collection of Usenet articles from aviation and auto discussion groups from [11]. For SRAA-Aviation dataset, we created a binary dataset using the aviation-auto partition. IMDB (movie) dataset is a collection of positive and negative movie reviews used in [9].

We preprocessed the dataset so that it is appropriate for a multinomial naïve Bayes. That is, we replaced all the numbers with a special token, stemmed all the words and eliminated terms that appear fewer than five times in the whole corpus. Stop words are not removed from the data to make sure the number of words the expert sees is similar to the number of words the model sees. We stress, however, that we kept the sequence of words in the documents, and that with the preprocessing the number of words did not change significantly. For example, the average document length before preprocessing in dataset Movie was 239 words and after preprocessing the average document length is 235 words.

2.6.2 Evaluation Methodology

For the active learning results, we simulated interaction with real users by using a student classifier and an expert classifier — the expert classifier has the same form as the student, but is trained on more examples (and thus has higher accuracy). We experimented

Table 1: Description of the real-world datasets: the domain, the number of instances, and label distribution.

Name	Feat.	Train	Test	Total Inst.	Label dist.
IMDB	26,784	25,000	25,000	50,000	50%
Reuters	4,542	4,436	1,779	6,215	36%
SRAA - Aviation	32,763	54,913	18,305	73,218	37%

with the multinomial naïve Bayes (MNB) classifier implemented in Weka [5]. Moreover, we prefer MNB to simulate an expert since it intuitively simulates how a human expert works. For instance, a human annotator builds his/her belief about the label of a document as s/he reads it. That is, each read word builds evidence upon the previous ones read. Similarly, MNB builds evidence based on the terms that appear in a document to determine its classification.

The default implementation of MNB uses Laplace smoothing. This implementation does not perform well for random sampling as well as active learning strategies in our datasets based on some preliminary experiments. We instead used informed priors where the prior for each word is proportional to its frequency in the corpus. Laplace smoothing is equivalent to create two fake documents, where each document contains every word in the dictionary exactly once. Informed priors is equivalent to creating two fake documents where every word appears proportional to its frequency in the corpus. This way, the prior will smooth more highly common terms to avoid accidental correlations that exist due to limited training data. For the purposes of this document when we refer to the classifier we mean this implementation unless otherwise stated.

We used the original train-test partitions of the dataset. The test partition is used for only testing purposes. We further divided the original train split into two: half is used for training the expert model while the remaining half is used as the unlabeled set $\mathcal{U}$. We performed two-fold validation in this fashion.

Simulating Noisy Experts: Given the thousands of annotations required for our experiments, we propose a method to simulate the noise introduced by labeling document fragments. (We leave user studies to future work.) In each experiment, we reserve a portion of the data to train a classifier that will simulate the human expert. When a new annotation is required by the learning algorithm, the prediction of this classifier is used. To simulate annotation of a document fragment, we simply classify the document consisting of the first k words.

We bootstrap all methods with two randomly chosen instances (one for each class). Then at each active learning step, we select randomly 250 unlabeled instances from $\mathcal{U}$ as candidates for labeling. The current loss and other computations needed by the methods are computed on a subset of 1000 unlabeled instances from the remainder of $\mathcal{U}$ (i.e. $\mathcal{U} \setminus Candidates$).

We evaluate performance on the test split of the dataset, report averages over the two folds and five trials per fold, and measure accuracy over a budget of number of words.

3. RESULTS AND DISCUSSION

In this section, we analyze our empirical results to address the three research questions from Section 1. Since our interest is to find the best way to spend a budget and we focus on cost-sensitive methods, we analyze the performance of each method given a budget of the number of words an expert can read. We approach our discussion considering both performance measure and spending of the budget.

3.1 RQ1. Annotation Error

Given that greater interruption can lead to greater annotation error, how do active learning algorithms perform in the presence of increasing amount of noise?

To answer this question we designed an active learning experiment using different levels of label noise introduced in the training data. We tested Multinomial naïve Bayes with informed priors, L_2 regularized logistic regression (LR) and support vector machines (SVM) implementation by LibLinear [4].

For the Label Noise Effect experiments, we create noisy data from the original training sets. We flipped the labels of 10% to 50% of the instances randomly. We evaluated using a train-test split repeating the experiment 10 times. Each experiment starts selecting two random instances (one for each class) and continues sampling randomly 10 instances at a time. We report the average over 10 trials.

In our results we observed that for all classifiers the performance unsurprisingly decreases at greater levels of noise. Figure 1 shows the accuracy performance of the three classifiers on data with 10% and 20% label noise. All three classifiers performed better on datasets with 10% noise (Figure 1(a), Figure 1(b) and Figure 1(c)) compared to 20% noise (Figure 1(d), Figure 1(e) and Figure 1(f)). Similar results were found with greater levels of noise however we use only two examples for illustration.

Interestingly, we observed that LR and SVM are more affected by the noise than MNB in particular at larger budgets. Moreover, MNB outperforms both LR and SVM at later iteration in the learning curve. This becomes more evident with greater percentages of noise. For instance, in Figure 1(b) MNB has a slower learning curve compared to LR and SVM whereas in Figure 1(e) MNB outperforms both classifiers half way through the learning curve.

In general, with greater levels of label noise the performance of the tested classifiers gradually decreased. However, we find that MNB consistently outperforms LR and SVM as the amount of label error increases, both in overall accuracy and in learning rate, and so we use MNB in all subsequent experiments.

3.2 RQ2 - Cost-Quality Tradeoff

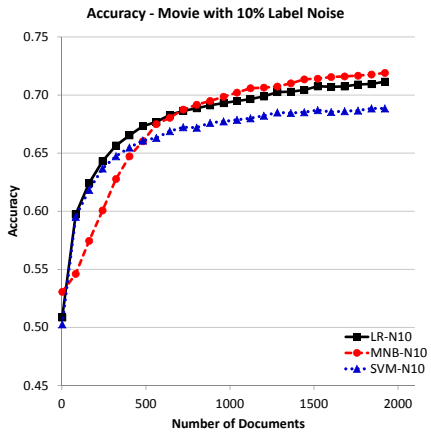
Under what conditions is the cost saved by using subinstances worth the error introduced? How does this vary across datasets?

In Section 2, we proposed that expanding the search space to include subinstances could reduce annotation cost. Furthermore, we argue that it is possible to trade off the cost and the value of the information obtained from an instance. However, we still have to establish how the use of subinstances affects the learning performance of the model. In this section, we discuss our findings on the use of labeling subinstances instead of full instances.

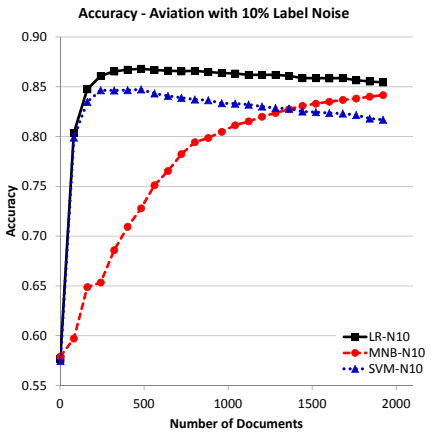
For this purpose, we tested the quality of the expert model training in one half of the data and tested showing the expert only the first k words of the documents. Also, we performed active learning experiments using random sampling and Expected Error Reduction (EER) as baselines testing subinstances of sizes 10 - 30 and 50. We followed the experimental methodology described in Section 2.6.

Our results show that using full size instances was never a good strategy, performing similar to the random sampling baseline. Figure 2 shows that for all datasets `EER-FULL` and `RND-FULL` were about the same.

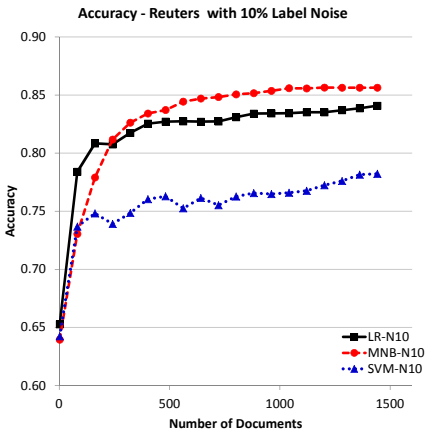
However, in general, using subinstances improves the performance on the active learner model. We conclude that the tested model prefers to obtain numerous noisy labels rather than fewer high quality ones. For instance, Figure 2(a) shows `RND-FIX--10` and `RND-FIX-30` perform better than `RND-FULL`and `EER--`



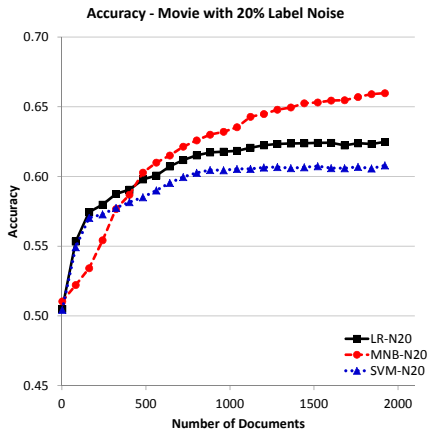
(a) Movie - Effect of 10% Noise



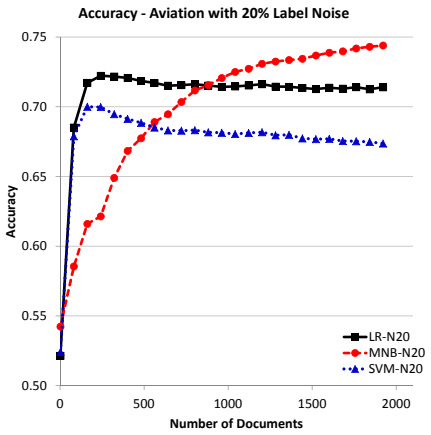
(b) Aviation - Effect of 10% Noise



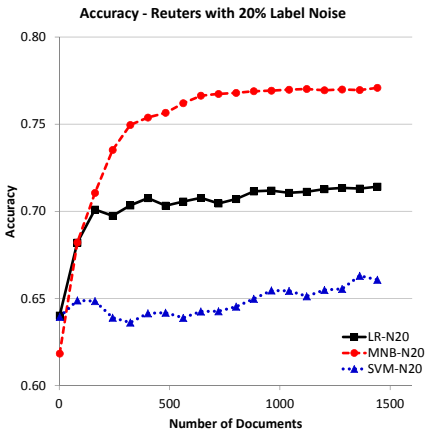
(c) Reuters - Effect of 10% Noise



(d) Movie - Effect of 20% Noise



(e) Aviation - Effect of 20% Noise



(f) Reuters - Effect of 20% Noise

Figure 1: Effect of Noise on the learning rate of several classifiers. MNB is less affected by label noise than LR and SVM.

FULL on Movie dataset. Similarly, EER-FIX-10 and EER-FIX-30 outperform both random sampling counterparts. Furthermore, Figure 2(b) shows that with subinstances of size 10 the expert quality is only about 70%, more than 15 points bellow expert quality using full instances. Similar results were observed on Reuters dataset (see Figure 2(e) and Figure 2(f)). This set of results confirms those reported in the Section 3.1.

In contrast, on Aviation dataset we observed that not all sizes of subinstances performed better than methods with full instances. In this case, EER-FIX-10 was worse than RND-FULL and EER-FULL whereas EER-FIX-50 outperformed all other methods.

In general, we find that some interruption is almost always better than none, resulting in much faster learning rates as measured by the number of words an expert must read. For example, in Figure 3(a), annotating based on only the first 10 words of a document achieves a 65% accuracy after 5,000 words that is comparable to a traditional approach requiring 25,000 words. The precise nature of this tradeoff appears to vary by dataset.

3.3 RQ3 - Adaptive Subinstance Selection

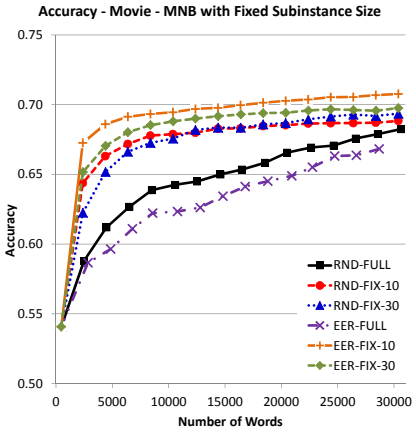
Does allowing the learning algorithm to select the subinstance size dynamically improve learning efficiency?

So far, we established a reference for the EER-FIX-k methods in terms of accuracy and how it translates to the budget efficiency. We have found that we can improve the budget efficiency by providing subinstances to the expert for labeling instead of full instances.

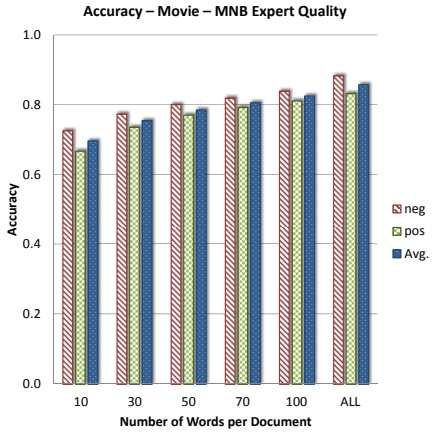
However, the best subinstance size changes across dataset. For further exploration of this idea, we implemented the proposed Equation 2 as QLT-γ methods (see Section 2 for details).

Our results suggest that the best value of γ depends on the budget. When the active learner's budget is severely limited, a smaller γ, which is equivalent to many-but-noisy labels, is preferred, whereas when the budget is large, higher quality labels can be afforded. For instance, in Figure 3(a) QLT-γ = 0.01 performs better initially than QLT-γ = 0.0001 however the latter improves with larger budgets and has the same performance as QLT-γ = 0.01 in the end. Similar results are observed in Figure 3(g) for the same methods on Reuters.

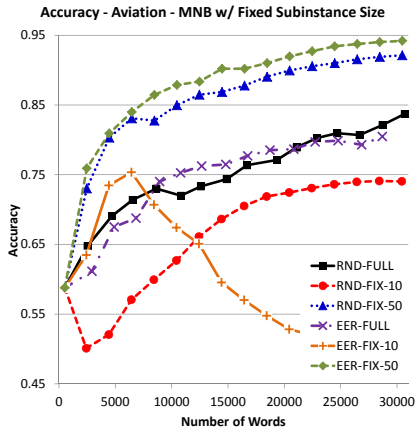
Furthermore, QLT-γ selects the size of the subinstance dynamically trading off cost and improvement. Figure 3(h) and Figure 3(b) show the average words per document for each method illustrating the dynamic approach of QLT-γ methods. Moreover, we observed different performances of the QLT-γ methods at different points of the budget spending. That is, expecting big improvements for larger budgets may not work best. For instance, Figure 3(i) shows that that QLT-γ = 0.01 works better at early iterations whereas QLT-γ = 0.0001 works better for later iterations. These differences are statistically significant as shown in Figure 3(i) where values below the 0.05 mark are statistically significant wins and above 0.95 are statistically significant losses.



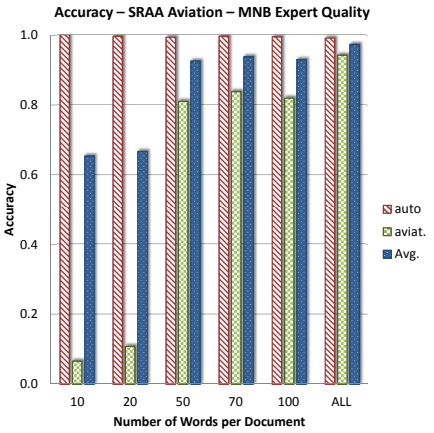
(a) Movie - Fixed Thresholds



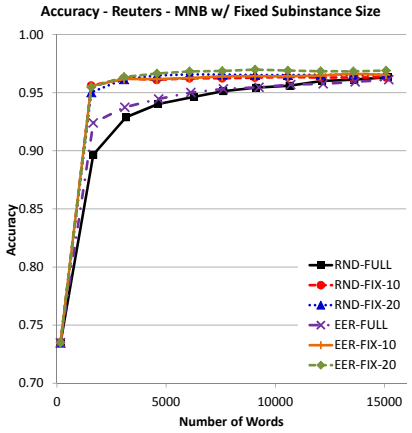
(b) Movie - Expert Quality



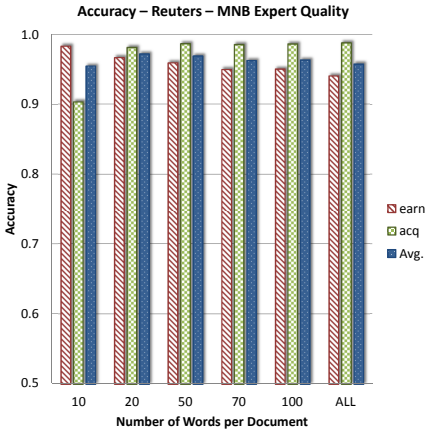
(c) Aviation - Fixed Thresholds



(d) Aviation - Expert Quality



(e) Reuters - Fixed Thresholds

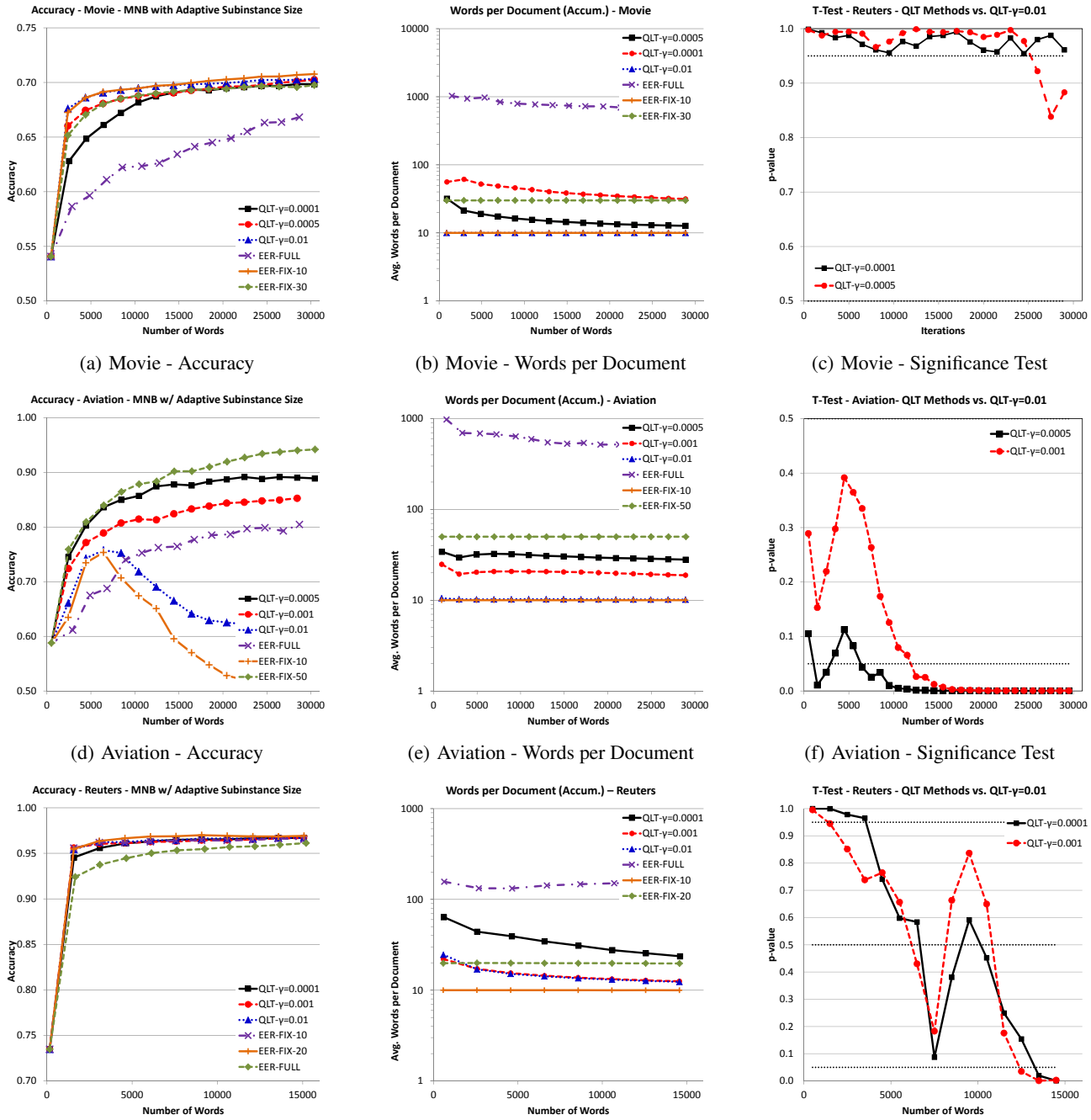


(f) Reuters - Expert Quality

Figure 2: Accuracy of fixed size subinstance baseline methods with MNB. On the right, expert quality of a MNB expert model tested on k words per document

The t-test results show that the tradeoff made by a big γ values with small budget compared to small γ for big budgets are significant. Based on this results, we conclude big γ selects mainly shorter subinstances which is beneficial for small budgets. On the other hand, small γ considers longer subinstances for labeling showing that for large budgets this works well.

Moreover, datasets that are more difficult to predict require a lower expected percentage of improvement for each word. That is, these datasets work better with smaller γ values where longer subinstances are considered also. That is the case of the Aviation dataset, considered a difficult dataset, where $\texttt{QLT-}\gamma = \texttt{0.0005}$ performs better than $\texttt{QLT-}\gamma = \texttt{0.001}$ and $\texttt{QLT-}\gamma = \texttt{0.01}$. In



(a) Movie - Accuracy (b) Movie - Words per Document (c) Movie - Significance Test

(d) Aviation - Accuracy (e) Aviation - Words per Document (f) Aviation - Significance Test

(g) Reuters - Accuracy (h) Reuters - Words per Document (i) Reuters - Significance Test

Figure 3: Comparison of fixed size subinstance methods and quality-cost adaptive subinstance size methods on a MNB classifier. On the center, the average words per document show the quality-cost methods dynamic selection of subinstance size compared to the fixed method. On the right, statistically significant p-value comparing QLT−γ methods

contrast, for an easy dataset such Reuters QLT−$\gamma = 0.01$ works best.

As we have shown, selecting the subinstance size dynamically is comparable to using a fixed size. However, finding the best fixed k value is difficult and depends on the dataset, whereas $\gamma = 0.0005$ works reasonably well across datasets. Moreover, one advantage of the dynamic approach is that formulating the size preference in terms of the cost of improving the model may provide a more intuitive way of setting model parameters.

4. RELATED WORK

Although there are commonalities among other cost sensitive approaches and our proposed method, to our knowledge this is the first approach that allows the learning algorithm to directly influ-

ence the annotation cost of an instance by revealing only a portion of it.

Feature value acquisition accounts for the value of asking questions to an expert and the expected improvement based on the answers such as the case of [1] and [10]. However, these formulations differ from ours since the feature values are known in our setting, but the expert is interrupted before the exert has the chance to fully inspect the instance.

In scenarios where multiple experts provide noisy labels, the active learner has to decide how many and which experts to query. Zheng et al. [15] discuss the use of multiple experts with different cost and accuracy. Their approach concentrates on ranking and selecting a useful set of experts, and adjusts the cost of the expert based on the corresponding accuracy while the instances are sampled by uncertainty. Similarly, Wallace et al. [14] allocate instances to experts with different costs and expertise. Nonetheless, the main difference with the current scenario is that each instance is assumed to have the same cost given the expert. Furthermore, the active learner does not have control of either the cost or the quality of the labels and rather depends on the cost of the expert.

Some decision-theoretic approaches incorporate cost into the active learning process by using linear cost function [7], or learning a cost model [3]. While these frameworks work well for their particular task, other studies report mixed results [13]. Instead, we propose an anytime active learning framework where the active learner balances cost and quality of labels by interrupting the expert labeling task.

5. FUTURE WORK

We have provided initial empirical evidence that expert interruption can lead to more efficient learning. All in all, our current set of results are important insight for anytime active learning algorithms that allow the learner to control and incorporate cost awareness. However, showing the first k words is only one way to interrupt an expert future directions include showing an automated summary of a document, showing selected sections of a document such as abstract and conclusion, etc. A future user study will provide additional insight. Another potential future direction include generalizing the anytime active learning framework to other domains, such as vision, besides text. Developing a general purpose active learning framework with anytime expert interruption is a promising new research direction.

6. CONCLUSION

Our main goal has been to design a framework that controls and accounts for cost and quality of training instances by means of interrupting an expert at any time during annotation. This interruption mechanism allows us to control the budget spending while improving the learning efficiency. We believe that this work can be extended to eliminate some of the assumptions and provide a better generalization to a broader range of domains.

References

[1] M. Bilgic and L. Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research (JAIR)*, 41:69–95, 2011.

[2] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[3] P. Donmez and J. G. Carbonell. Proactive learning: : Cost-sensitive active learning with multiple imperfect oracles. In *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, page 619, oct 2008.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[6] R. A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[7] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[8] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, Dec. 2004.

[9] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[10] P. Melville, F. Provost, M. Saar-Tsechansky, and R. Mooney. Economical active feature-value acquisition through expected utility estimation. In *Proc. of the KDD Workshop on Utility-based Data Mining*, 2005.

[11] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 792–799, 1998.

[12] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

[13] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Neural Information Processing Systems*, pages 1289–1296, 2008.

[14] B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos. Who should label what? instance allocation in multiple expert active learning. *In Proc. of the SIAM International Conference on Data Mining (SDM)*, 2011.

[15] Y. Zheng, S. Scott, and K. Deng. Active Learning from Multiple Noisy Labelers with Varied Costs. In *IEEE 10th International Conference on Data Mining (ICDM)*, pages 639–648, 2010.

Storygraph: Extracting patterns from spatio-temporal data

Ayush Shrestha
Department of Computer Science
Georgia State University
Atlanta, Georgia
ashrestha2@cs.gsu.edu

Ying Zhu
Department of Computer Science
Georgia State University
Atlanta, Georgia
yzhu@cs.gsu.edu

Ben Miller
Department of Communication
Georgia State University
Atlanta, Georgia
miller@gsu.edu

Yi Zhao
Department of Mathematics
Georgia State University
Atlanta, Georgia
yzhao6@gsu.edu

ABSTRACT

Analysis of spatio-temporal data often involves correlating different events in time and location to uncover relationships between them. It is also desirable to identify different patterns in the data. Visualizing time and space in the same chart is not trivial. Common methods includes plotting the latitude, longitude and time as three dimensions of a 3D chart. Drawbacks of these 3D charts include not being able to scale well due to cluttering, occlusion and difficulty to track time in case of clustered events. In this paper we present a novel 2D visualization technique called *Storygraph* which provides an integrated view of time and location to address these issues. We also present storylines based on Storygraph which show movement of the actors over time. Lastly, we present case studies to show the applications of Storygraph.

Categories and Subject Descriptors

I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Line and curve generation*

Keywords

Spatio-temporal visualization, Information visualization

1. INTRODUCTION

With the advent of newer and cheaper location tracking devices and services, huge amount of spatio-temporal data is generated everyday. To visually analyze these kinds of data, presenting the time information and the location information separately is quite trivial. However, presenting spatio-temporal events together in an integrated view is not. For instance, it is not easy to present locations in a time series chart, and it is not easy to present temporal information on a map. To address this issue, we present an interactive 2D visualization technique called *Storygraph* in this paper.

Storygraph consists of two parallel vertical axes similar to the vertical axes in the parallel coordinates, along with a horizontal axis. The vertical axes represent the location pair e.g. latitude and longitude while horizontal axis represent time. Each location (x, y) or $(latitude, longitude)$ is represented as a line segment referred to as *location line* in rest of the paper. Location lines are created by joining the corresponding values on the axes. Events occurring at that location at different times are plotted as markers along the location line. We assume that all events contain location (latitude and longitude) and time information in the data set.

In Storygraph, the location lines of two locations close to each other geographically are also close in the Storygraph. Thus it help users understand events in their spatial, temporal and in spatio-temporal context. This results in a unique and powerful ability of Storygraph to display recurring events or periodicity in data.

In many cases, data sets often contain attributes other than time and location. For such data sets, a common analysis technique is to highlight certain dimensions. For example, in a collection of human rights documents, besides the time and location of the violation, the name of perpetrators, victims, type, description is often present. In such data sets users might want to highlight a certain type of violence to better analyze it. To accomplish this, the Storygraph allows users to change the color, size and shape of the markers to highlight these dimensions.

Collection of events generally contain actors. Actors in this paper refer to groups of people, individuals or organizations present in the event. In a human rights violation event of "kidnapping of an individual by a guerrilla army and Red Cross helping the victims", Red Cross, guerrilla army and the individual could be three actors. Another common analysis is tracking the movement of characters. Using the same example as above, suppose the user wants to track the movement of the guerrilla army, and the army spends longer time in one place and lesser in others. Using traditional cartographic maps, users lose the sense of time. On the other hand, it is difficult to show location using timelines. To ad-

IDEA'13, August 11th, 2013, Chicago, IL, USA.

dress this, we also present *storylines* based on Storygraph which shows the movement of actors in space and time.

We demonstrate the effectiveness of Storygraph using two case studies based on Afghanistan War Diary data and Vietnam War-era Laos bombing data. Storygraph reveals interesting patterns including gaps and trends in the events, thus providing insights into the data set.

The rest of the paper is organized as follows. In Section 2 we describe the related visualization techniques and recent work on the subject. In Section 3 we describe the nature of data that we used. Section 4 describes our main visualization technique: Storygraph. Following that we describe the implementation details in Section 5. Lastly we present the case studies in Section 6.

2. RELATED WORK

We are interested in visualizing spatio-temporal data sets and storylines based on it. Hence our analysis of related work focuses on the previous visualization methods for temporal and spatial data.

Much work has been done on time series data visualization some of which try to visualize storylines [2] [3] [10] [37] [20] [14] [13] [39] [23]. None, however, integrate time, location and actors. For example, visualization methods proposed in [10] [13] [23] can visualize time, actor, and sometimes context, but they do not include location.

Many methods have been proposed for spatio-temporal data visualization and analysis. Here we classify these methods based on whether it's 2D or 3D and whether it has integrated or separate spatial and temporal views. Thus we review the visualization methods in four categories: 2D separate views, 2D integrated view, 3D integrated view, and 3D separated views. Following that we review the existing work on storylines.

2.1 2D separated views

Many visualization methods present spatial and temporal data in different views. One key question is how to synchronize the presentation of temporal and spatial data. A number of methods have been proposed. For example, [8] [7] [31] used small multiple views to link spatial data with temporal data. Each small map view shows a subset of the temporal data. Plug et al. [31] use one map per weekday. Jern et al.[21] use color coding to link temporal data with spatial data. Fredrikson et al. [11] use a Gantt chart to link temporal data with spatial data. In this case, each small map view corresponds to a bar in the Gantt chart. Authors in [11] [28] use interactive synchronized views. When the user clicks on a data point in the temporal view, the corresponding point in the spatial view is automatically highlighted, and vice versa.

These methods have their limitations. The small multiple map views used by [8] [7] [31] cannot accommodate small temporal data scale. To avoid generating too many map views, temporal data is cut into large chunks. Color coding to link temporal data with spatial data is not intuitive, as do not have a standardized correlation, while time line is generally sequential. Interactive, synchronized spatio-temporal views are also limited, as they fail to present the continuous correlation between spatial and temporal data.

2.2 2D integrated views

In this category, the spatial and temporal information are visualized in one view. For example, Andrienko et al. [5] propose a method that superimposes time graph on multiple maps. However the time graph may occlude the data points on the map view. In addition, events that happened at the same location but at different times may occlude each other. If many incidents happen on one location over a long period of time, many data points will be piled up on one spot, making it difficult to read and analyze.

2.3 3D integrated views

To address the problems of 2D integrated views, some researchers proposed 3D integrated visualization of spatio-temporal data. For example, in the space-time cube method [12], a synchronized 3D time line is plotted above a 2D map. Tominski et al. [35] place 3D icons on a 2D map to visualize events that happen on the same location over a period of time. In addition, time path is plotted above the 2D map. The benefits of 3D integrated view is that the time graph do not occlude the 2D map. If there are many data points at one location, there is a third dimension to accommodate the icons. However, the 3D integrated views also have its drawbacks. First, it's difficult to align time data with location in 3D. Techniques used by Kapler et al. [22] can be used to a certain extent to align the time data but as the number of events grows, the scalability reduces. These are the inherent problems of 3D data visualization.

2.4 3D separated views

Adrienko, et al. [1] use three separate but linked views: a 3D space-time cube, a 2D map view, and a time line view. A similar approach is used in Landesberger, et al. [24].

Compared with previous methods, our proposed method has the following benefits:

- It provides a big picture of the entire set of events in time as well as location with details displayed on demand.
- Spatial and temporal data are fully integrated in one view, and temporal and spatial data do not occlude each other.
- It is a 2D integrated visualization, thus avoiding the problems of 3D visualization.

2.5 Storylines

Starting with Minard's map of Napoleon's march to Moscow [36], there have been many efforts to create storylines [12] [4] [33] [9] [34]. Authors in [12] [4] [9] use 3D maps and plot the movement over it. Space-time paths are used to show the time spent by each of these characters in the location. To avoid over plotting when the number of actors increase, clustering is often used by the authors. Rinzivillo et al. [33] use 2D map and plot the trajectories of actors along streets. While it is ideal for actors moving with uniform velocity and provides highly accurate geographic information on the movement, it fails to show how much time did the actor spent at one point in case of non-uniform velocity of actors. Tanahashi et al. [34] create storylines of different characters in movies to show how they interact at what point in time. The resulting 2D diagram gives a good sense of time but only provide approximate location information based on the actors. Implementing storylines with Storygraph results in a visualization with high precision in both location and time making it more suited towards visualizing log data.

3. DATA CLASSIFICATION

Different types of visualizations work best on different types of data. In this paper we classify the data into three types: *Structured*, *Semi-structured* and *Unstructured*. Structured data sets contain precise geolocation and a time stamp for each event within the set. These data are uniform i.e, all the time stamps have equal precision (unlike some time stamps containing only year and another containing year, month and day) resulting in minimal or no ambiguity e.g. geo-sensor data, military operational logs etc. Most visualizations above use data from this category. Few common limitations include poor scalability though many authors have proposed workarounds by clustering. For example, Stardinates[25] works most appropriately on structured data[26] but fails to handle cases of high frequency data, Gravi++[18] works best for time dependent data series but has limitations in the frequency of the data as in case of Stardinates, method presented by Geng et al. [13] works well for large high-dimensional data but doesn't incorporate the temporal aspect well. Semi-structured data set include description of events with at least one of spatial or temporal component being in precise form. e.g. News report, Captain's Logs etc. In these data sets, the uniformity of the precision might not be guaranteed across the set, thus resulting in uncertainty. An example of this would be, data set containing three events with locations "Atlanta", "Atlanta, Georgia" and "10th Street, Atlanta Georgia". Similarly examples of non-uniformity in time include historic archives where some document have precise dates while others have only year, month-year present. Unstructured data include text reports where the extracted data may not have precise location and time, thus containing high amount of uncertainty. Examples include interview with the Fire fighters who were present in the scene during 9/11. In this paper, we only deal with structured data and leave the semi-structured and unstructured for future work.

4. METHOD

The Storygraph, is a 2D diagram consisting of two parallel vertical axes $V_\alpha \subset \Re$ and $V_\beta \subset \Re$ and an orthogonal horizontal axis $H \subset \Re$. Theoretically, all three of the axes, as in Cartesian graphs, are unbounded at both ends. In practice we take the maximum and minimum values in the data set for all corresponding axes and draw the graph accordingly. The values in the axes are ordered in ascending order: from left to right in horizontal axis and bottom to top in vertical axes. In this paper, vertical axes V_α and V_β represent the x and y coordinates of a point on a plane such as latitude and longitude. The horizontal axis, H, represents time. Thus a point plotted on Storygraph, which shall be referred to as *event* in the rest of the paper will have at least three dimensions: Parallel coordinates and a timestamp.

For any event occurring at (α, β) in time t and $t+1$ as shown in Figure 1, our algorithm first draws a *location line* by connecting the points on the two axes, $\alpha \in V_\alpha$ and $\beta \in V_\beta$. The algorithm then returns the points on this line at time t and $t+1$ respectively.

The function $f(\alpha, \beta, t) \rightarrow (x, y)$ which maps an event to the 2D Storygraph plane can be formally written as follows:

$$y = \frac{(\beta - \alpha)(x - T_{min})}{T_{max} - T_{min}} + \alpha \tag{1}$$

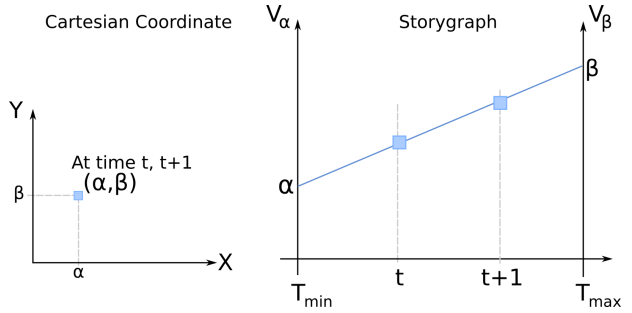


Figure 1: Left: Two events taking place at the location (α, β) at time t and $t+1$ in the map, Right: Same two events represented in a Storygraph with parallel coordinates and timestamp.

$$x = t \tag{2}$$

where T_{min} and T_{max} are the maximum and minimum timestamps within the data set.

Figure 1 illustrates how a location on a regular 2D map, coded with a Cartesian coordinate, is presented in the Storygraph. Equation 1 and 2 is used to convert a location on a regular map to a Storygraph plane, and vice versa. As seen from the equations, such conversion is very efficient and can be done in real time allowing users more interactivity.

4.1 Cluttering and Ambiguity

While plotting a large number of events at different locations, often the location lines result in over plotting and cluttering. To address this issue, we set the alpha value of the location line as a function of some attribute selected by the user similar to the method described by Atero et al. [6].

Given some attribute a with maximum and minimum values a_{max} and a_{min} and a tuning parameter p, the color c of the location line is given by

$$c = (\frac{a - a_{min}}{a_{max} - a_{min}} * 255^{(1/p)})^p \tag{3}$$

The tuning parameter p is used to control the difference between the higher values and the lower values. Figure 2 shows the change in the intensity of the location lines varying p. Higher the value of p, more evident is the difference between maximum and the minimum values of the attribute. Equation 3 is also used to color markers in the figure setting $p = 0.18$ in all four cases.

Our implementation of Storygraph also has a button to show/hide location lines. Hiding location lines is particularly useful when the data set is dense since the events align with each other over the hidden location lines giving a sense of the location. Hiding the location lines on the sparse data set may cause ambiguity since a point on the Storygraph may belong to multiple location lines. Another form of ambiguity occurs when two events at different locations get mapped to a single point on the Storygraph so that even though there are two location lines, its hard to distinguish which event belongs to which location. Our software alleviates this problem by providing zooming and filtering functions. Users can zoom into the Storygraph by clicking and dragging the mouse cursor over the Storygraph. The graph is redrawn expanding the selected area. Similarly, users can filter the range of latitude, longitude and time by specifying

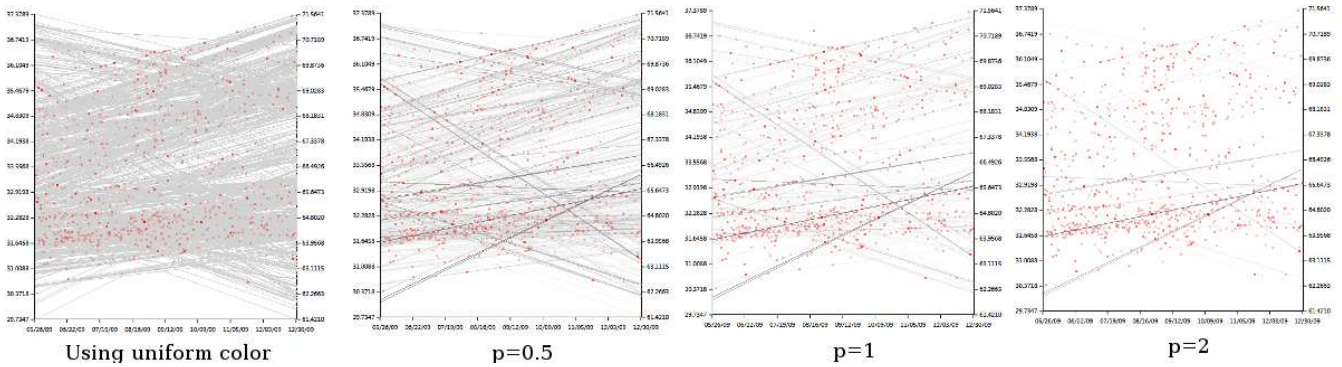


Figure 2: The location lines drawn on the same set of storylines with different tuning parameters. From left to right, the first image has location lines painted with the same color. For the second, third and the fourth image, $p = 0.5$, $p = 1$ and $p = 2$ are used respectively. Higher value of p is useful for highlighting the attributes close to maximum. The same technique is used to paint the event markers as well in all the images with $p = 0.18$.

the numbers manually from a dialog box. Zooming and filtering minimizes the likelihood that two point overlap even after the Storygraph is redrawn.

4.2 Storyline generation

In this paper, we describe storyline as a series of events associated with an actor. Hence, in order to draw storylines, the data also needs to have actor information as one of its attributes. Storyline is constructed by connecting all the events sequentially in which the actor was involved resulting in a polyline. Multiple storylines can be drawn one for each actor thus allowing users to visualize the interactions between them. Storylines are especially helpful in visualizing movement data as in [9], [29], [30] because users can clearly see how different characters converge at certain events and then move on to different directions.

Since the event data is discrete, the movement of the actor between two time points cannot be determined. This uncertainty is presented in the Storylines by using furlough lines between two points as opposed to drawing solid lines.

Each storyline has a color value associated with it to distinguish one unit from the another. The color scheme in the implementation was adapted from Color Brewer [17]. This led to a problem when the number of actors increased beyond twelve - the upper cap of the brewer. To overcome this limitation, we hashed the unit names and converted those hashes into colors assuring each distinct unit to have a distinct color. In this process some colors differed from each other negligibly making two storylines almost indistinguishable by human eye. In this paper we only focus on storylines of less than twelve units and leave the color generation for higher numbers as future work.

Storylines based on Storygraph can be compared with the well established space-time path concept in Time-geography [16] because they share similar characteristics. Both techniques attempt to present temporal and spatial data in an integrated way. However, because Time-geography is a 3D visualization, it suffers from occlusion, cluttering, and difficult depth perception as more data points are plotted. On the other hand, Storygraph is a 2D visualization and therefore does not have any occlusion or depth problem. As can be seen from Figure 3, Storygraph can present co-location in time or space and co-existence with lesser ambiguity than Time-geography.

4.3 Surroundings

When analyzing an event or a storyline, it is important to examine each event in its context or *surroundings*. In this paper, the surroundings refer to the events that have taken place in the temporal and spatial vicinity of the event. In Storygraph, surroundings can be defined in terms of space, time, actors or any of their combinations. For example, the surrounding of an event may be events that occurred in different neighboring locations. It could also be other events that an actor experienced in the past before that event. Analysis of surroundings in Storygraph can be compared to animated maps or videos of maps showing changes. An advantage of the Storygraph over the animated maps is the users do not need to track what happened where mentally frame by frame. Our implementation allows users to filter the view to get surroundings of a particular event. Users can adjust the coverage of the surroundings in terms of space, time as well as actors.

5. IMPLEMENTATION

We implemented the Storygraph using Microsoft WPF and C#. We used MySQL database in the back end to store the data set. For visualizing the data we used a layer based approach as case of most GIS applications - the events, location lines and storylines plotted in different layers. Thus the users can get additional information about the data set from different sources and overlay it on the existing Storygraph to further facilitate the analysis of the events. We implemented this feature using the Wikileaks Afghanistan war logs [15] and Cablegate data [38] as shown in Figure 5. The Afghanistan war logs served as the first layer on top of which the Cablegate material was overlayed. Since the Cablegate material only had the embassy information for all the cables, we plotted them as lines instead of points on the Storygraph. Each vertical line in the Cablegate overlay corresponds to a cable sent on that day.

6. CASE STUDIES

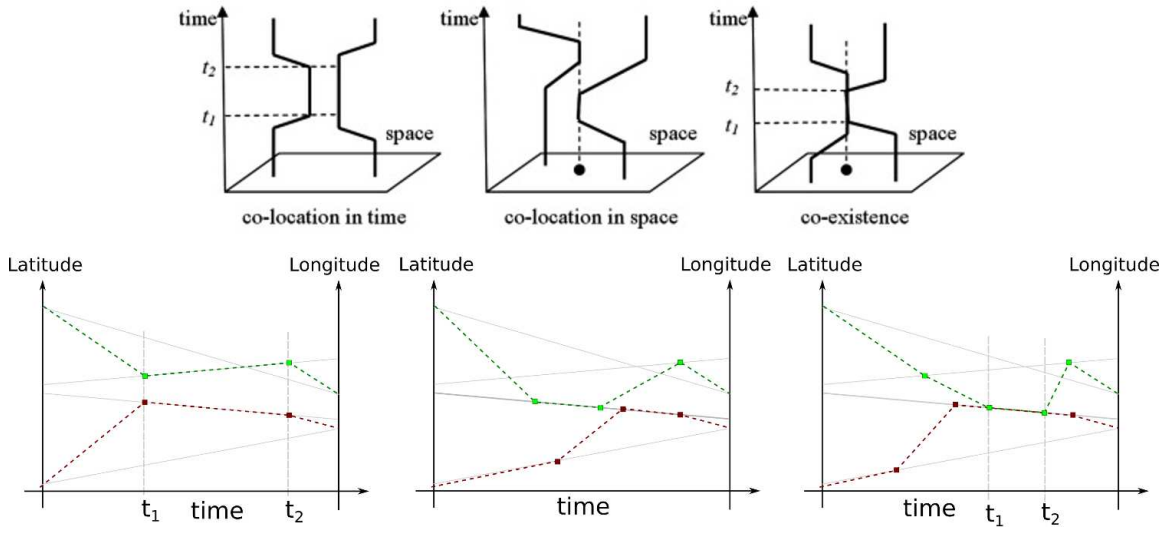


Figure 3: Comparison of the time-space paths using Time-Geography, adapted from [19] with storylines drawn using Storygraph. Starting from the left, the top figures directly compares to the bottom figure - first showing co-location in time, second showing co-location in space and third showing co-existence.

We applied our techniques on two databases: Wikileaks Afghanistan War Diary (War Diary) and data related to Laos from NARA's Southeast Asia Data Base of Records About Air Sorties Flown in Southeast Asia (Laos UXO). Our visualizations, in some cases, revealed meaningful patterns in the data for which we were able to generate hypotheses.

6.1 Afghanistan War Log (2004-2010)

The War Diary comprises U.S. military significant activity reports from Afghanistan during the period 2004-2010 [15]. Consisting of approximately 60K highly structured records, the data provides a rigorously categorized and unprecedented look at the daily conduct of war. Figure 4 shows the Storygraph generated from all the events categorized as "Enemy Action" or "Explosive Hazard" during the war. We observed few interesting patterns marked by $A, 1, 2$ and 3.

The vertical bands marked by A in Figure 4 and B in Figure 5 are formed due to the events clustering between Jul-Oct 2005 and near Sept 2009. They indicate widespread and coordinated "Enemy Action" and "Explosive hazards" events at a very short period of time. By correlating these dates to broader events in Afghanistan, we discovered that these clusters were proximal to elections; our hypothesis is that the incidence of violence, and therefore the number of activity reports, increases during elections [38]. Although this is not new information, it demonstrates the ability of Storygraph to help identify significant event patterns.

Numbers $1-3$ in the Figure 4 shows a periodicity of voids, or a lack of reports in those areas. These "holes" seem to appear around the end of the year. From location lines, we found that the geographic location by these holes correspond to a section of the Kabul-Kandahar highway, a key portion of national road system of Afghanistan and a main target of attacks. The Storygraph visualization shows that there have been regular quiet periods for that section of the highway around the end of 2005, 2006, 2007, and 2008.Since the data set does not extend beyond December 2009, we are unable to confirm if the pattern repeated in 2009. To our knowledge, this pattern has not been identified or discussed in any published report. It is beyond the scope of this paper to evaluate the significance of this pattern and explain the cause of it. However, it demonstrates the effectiveness of the Storygraph to help discover patterns and form hypotheses.

Figure 6 shows the storylines of seven Afghanistan based combat units from October 1-20, 2009. It can be seen that units $tfprotector$ and $tfcyclone$ were at the same location during October 5 - 7 marked by D in Figure 6 and also during 12 - 14. Similarly $E1$ and $E2$ shows the presence of two units alternated in that location. It is also clear that $tfwhiteeagle$ travelled a wider range of locations than the rest, $tflethal$ travelled a lesser range and $tfprotector$ didn't travel at all. Hence, besides showing the mobility of the units and how the unit interacted with other units, the greatest strength of the storylines based on Storygraph is it also shows the time period during which the unit was most mobile or least mobile.

6.2 Unexploded Ordnance Laos (1964-1973)

An ongoing problem in war torn regions is the persistence of unexploded ordinance. Established by the Lao government with support by various NGOs, the Lao National Unexploded Ordnance Programme (UXO Lao) addresses this ongoing problem so as to reduce the number of new casualties and increase the amount of land available for agriculture. By their estimates, approximately 80m unexploded bombs remain in Laos as a result of U.S. Air Force (USAF) bombing missions during the Vietnam Conflict. These bombs affect 25% of all villages, all 17 provinces, and have resulted in approximately 25k injuries and fatalities in the post-war period, 1974-2008 . This data set details bombings in Laos by the USAF during the period of the war, 1968-1975 and is also a military, descriptive log of the bombings. It consists of 60K reports documenting approximately 2m tons of ordinance. Figure 7 shows the graph obtained from this data set.

Figure 7 shows all the events in the UXO data set. The

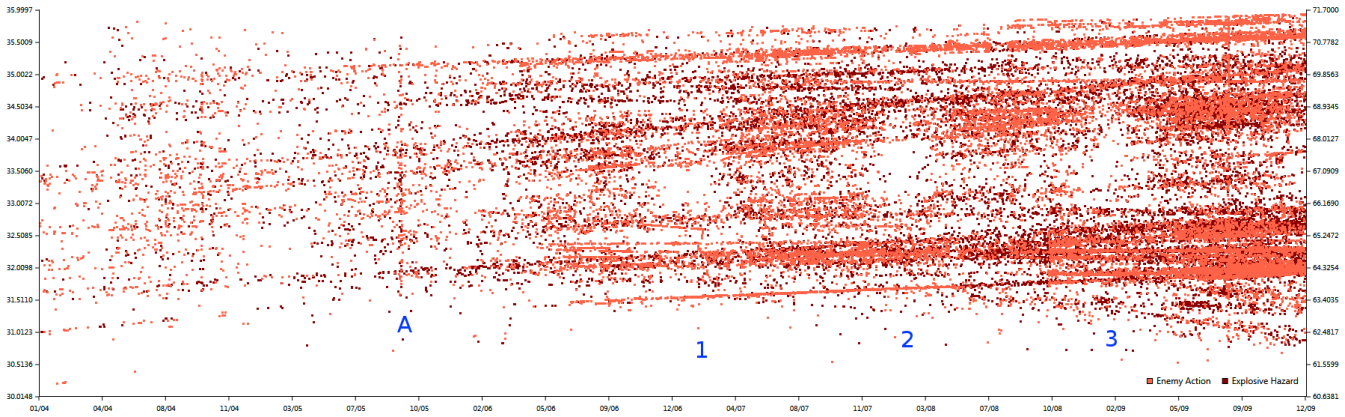


Figure 4: Storygraph showing all the events categorized as "Enemy Action" or "Explosive Hazard" during Afghanistan war from 2004 to 2009 across different regions of the country. Pattern marked by A shows that a lot of events took place during the same time period throughout the country. Patterns marked by $1-3$ show voids - indicating that the attacks were periodic in nature.

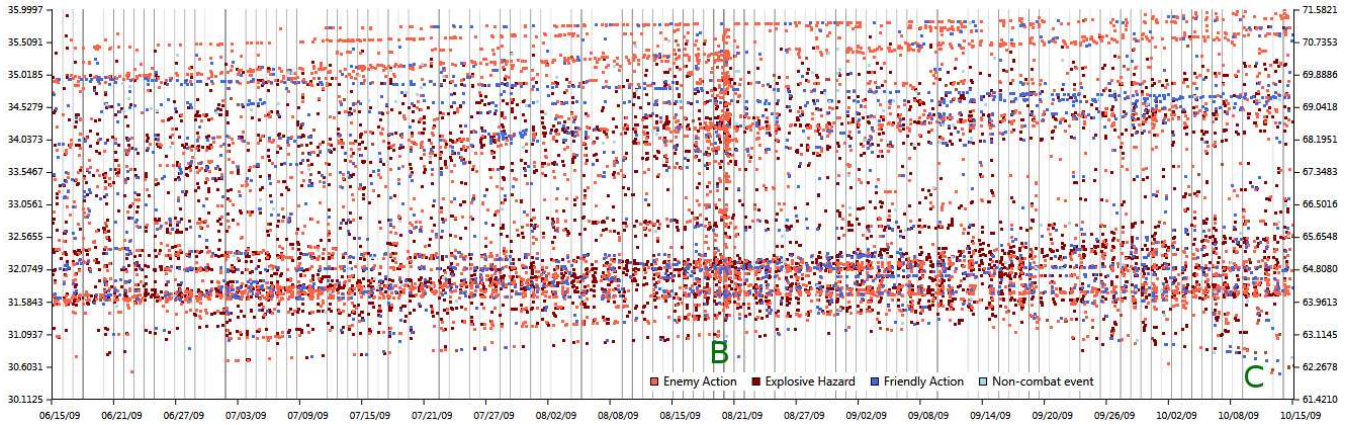


Figure 5: The Afghanistan war log from June 2009 to October 2009 with Cablegate cables within the same time frame plotted overlayed. The figure shows four types of events "Enemy Action", "Explosive Hazard", "Friendly Action" and "Non-combat event". Pattern B, similar to pattern A in Figure 4 show a number of events taking place in the entire region within the small time frame. Pattern C shows events taking place in near regular intervals (not induced due to the lack of precision in time).

markers are colored according to the number of bombs dropped. Patterns marked by $A1$ and $A2$ show that the bombings intensified during this period. A common error in interpreting Storygraph arises in cases like $A2$ where the users might be tempted to argue that more bombs were dropped in a certain location during that time. This kind of confusion resulting from crossing location lines can be alleviated in our implementation by zooming into the region. Figure 8 shows the zoomed view of region marked by $A2$ in Figure 7. The alpha value of the location line has been changed according to the number of bombings so that the higher number of bombings are highlighted. The figure shows two crossing bands implying that these locations were constantly bombed (as opposed to bombings concentrating at one location).

The Storygraph obtained from this data set shows three clear patterns.

1. Events clustering on location lines to form distinct lines can be seen from the start to June 1966. These lines signify that the bombings were focused at certain locations at almost regular intervals.

2. The bombings reduced drastically after March 1972 when most US troops left Vietnam.

3. The patterns of bombing data are interspersed with periodic bands of white.

We have two hypothesis for those voids: either they could mean that bombings were paused during that range of time, it could also mean that the data correlated to those raids during that period was redacted from the set. Like much military data, classified operations such as those by special forces are frequently not contained within general operation activity reports. One way to test this second hypothesis would be to correlate bombing data to peace talks, truces, and ceasefires as documented in other sources. Identifying the focal locations of bombing campaigns helps groups like UXO Lao address the areas most in need of re-mediation.

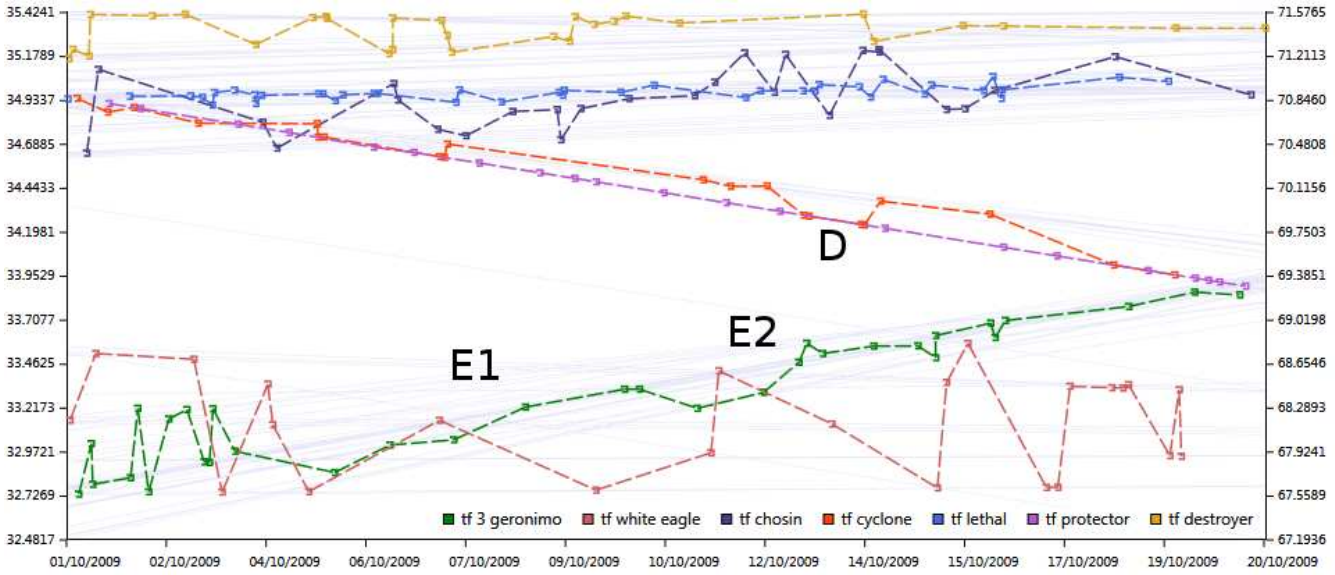


Figure 6: Storylines of seven Afghanistan based combat units from October 1 - 20, 2009. The pattern marked by D shows co-location in space and time - meaning that the two units were at the same location. Pattern marked by $E1$ and $E2$ show that co-location in time - meaning that one unit was present in the location before another.

7. CONCLUSION

In this paper, we have presented our novel visualization technique, Storygraph, which provides an integrated view containing locations and time. We also presented storylines based on Storygraph. Storygraph addresses the problems in previous 3D and integrated 2D spatio-temporal data visualization by minimizing glyph occlusion and cluttering. This improved design help users better correlate events on both the spatial and temporal dimensions. Storylines help track the movement of characters and the interactions between them. In the future we plan to extend the visualization to incorporate uncertainty in location and time. We also intend to implement data clustering algorithms to handle large scale data sets.

8. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1209172. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] G. Adrienko, N. Adrienko, M. Mladenov, M. Mock, and C. Politz. Identifying place histories from activity traces with an eye to parameter impact. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):675–688, May 2012.

[2] W. Aigner, S. Miksch, W. Muller, H. Schumann, and C. Tominski. Visualizing time-oriented data - a systematic view. *Computers and Graphics*, 31(3):401 – 409, 2007.

[3] W. Aigner, S. Miksch, W. Muller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47 –60, jan.-feb. 2008.

[4] G. Andrienko, N. Andrienko, and S. Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46, 2007.

[5] N. Andrienko, G. Andrienko, and P. Gatalsky. Visualization of spatio-temporal information in the internet. In *Proceedings of 11th International Workshop on Database and Expert Systems Applications*, pages 577 –585, 2000.

[6] A. O. Artero, M. C. F. de Oliveira, and H. Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 81–88. IEEE, 2004.

[7] A. Asgary, A. Ghaffari, and J. Levy. Spatial and temporal analyses of structural fire incidents and their causes: A case of Toronto, Canada. *Fire Safety Journal*, 45(1):44 – 57, 2010.

[8] C. Brunsdon, J. Corcoran, and G. Higgs. Visualising space and time in crime patterns: A comparison of methods. *Computers, Environment and Urban Systems*, 31(1):52 – 75, 2007.

[9] U. Demsar and K. Virrantaus. Space-time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, 24(10):1527–1542, 2010.

[10] D. Fisher, A. Hoff, G. Robertson, and M. Hurst. Narratives: A visualization to track narrative events as they develop. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*, pages 115 –122, oct. 2008.

[11] A. Fredrikson, C. North, C. Plaisant, and B. Shneiderman. Temporal, geographical and

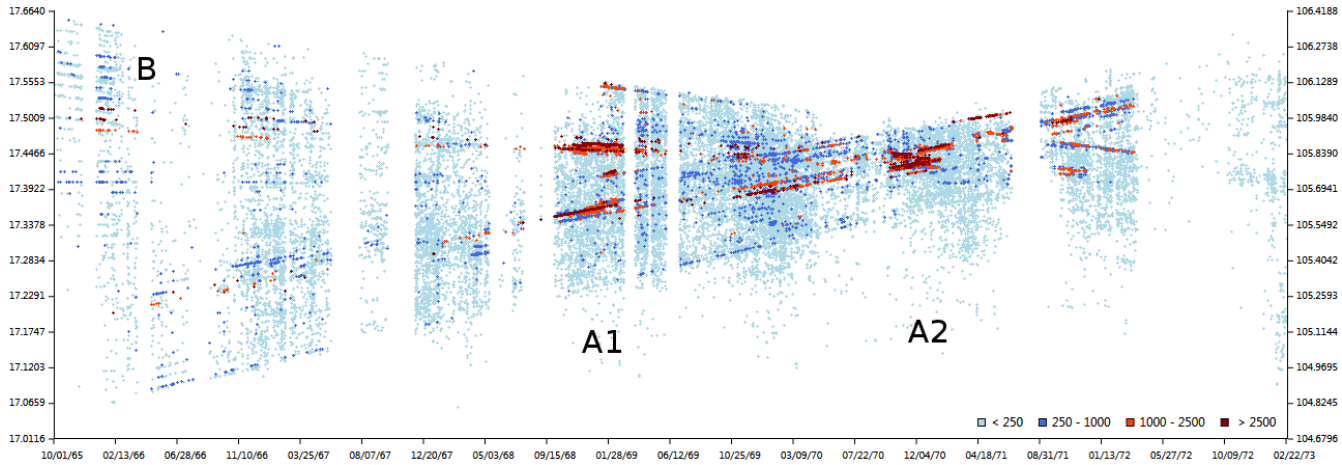


Figure 7: Storygraph generated from Laos data set. Each event corresponds to an instance of bombing. Patterns marked by $A1$ and $A2$ showing bombing intensified during the period. Pattern B shows that some places were bombed regularly. Also evident in the figure are the vertical bands of white either meaning missing data or the stopped bombings.

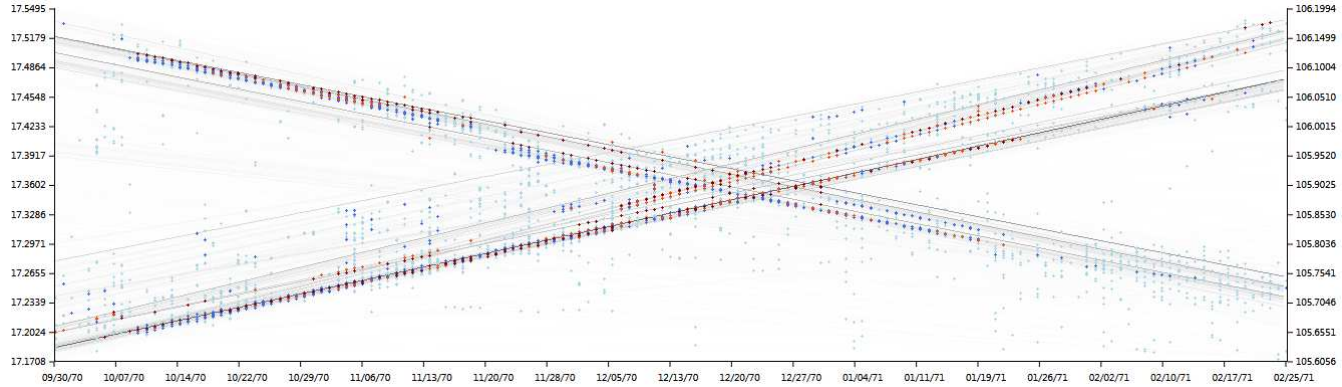


Figure 8: Area marked by $A2$ in Figure 7 zoomed together with location lines. The alpha value of the lines were set according to the number of bombs dropped so that more prominent areas and bands of location are highlighted.

categorical aggregations viewed through coordinated displays: a case study with highway incident data. In *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM internation conference on Information and knowledge management*, NPIVM '99, pages 26–34, New York, NY, USA, 1999. ACM.

[12] P. Gatalsky, N. Andrienko, and G. Andrienko. Interactive analysis of event data using space-time cube. In *Proceedings. Eighth International Conference on Information Visualisation*, pages 145 – 152, july 2004.

[13] Z. Geng, Z. Peng, R. Laramee, R. Walker, and J. Roberts. Angular histograms: Frequency-based visualizations for large, high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2572 –2580, dec. 2011.

[14] T. Gschwandtner, W. Aigner, K. Kaiser, S. Miksch, and A. Seyfang. CareCruiser: Exploring and visualizing plans, events, and effects interactively. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis)*, pages 43 –50, march 2011.

[15] Guardian.co.uk. Afghanistan war logs 2003-2010, July 2010.

[16] T. Hägerstrand et al. Time-geography: focus on the corporeality of man, society, and environment. *The science and praxis of complexity*, pages 193–216, 1985.

[17] M. Harrower and C. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *Cartographic Journal, The*, 40(1):27–37, 2003.

[18] K. Hinum, S. Miksch, W. Aigner, S. Ohmann, C. Popow, M. Pohl, and M. Rester. Gravi++: Interactive information visualization of highly structured temporal data. *Workshop IDAMAP 2005 Intelligent Data Analysis in Medicine and Pharmacology*, pages 67 – 72, 2005.

[19] T.-H. Hsieh, J.-J. J. Chen, L.-H. Chen, P.-T. Chiang, and H.-Y. Lee. Time-course gait analysis of

hemiparkinsonian rats following 6-hydroxydopamine lesion. *Behavioural Brain Research*, 222(1):1 – 9, 2011.

[20] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927 –934, nov.-dec. 2010.

[21] M. Jern and J. Franzen. GeoAnalytics - exploring spatio-temporal and multivariate data. In *Proceedings of Tenth International Conference on Information Visualization*, pages 25 –31, july 2006.

[22] T. Kapler and W. Wright. Geotime information visualization. *Information Visualization*, 4(2):136–146, 2005.

[23] M. Krstajic, E. Bertini, and D. Keim. CloudLines: Compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432 –2439, dec. 2011.

[24] T. V. Landesberger, S. Bremm, N. Andrienko, G. Adrienko, and M. Tekusova. Visual analytics for categoric spatio-temporal data. In *Proceedings of IEEE Conference on Visual Analytics Science and Technology (VAST 2012)*, pages 183–192, October 2012.

[25] M. Lanzenberger, S. Miksch, and M. Pohl. The Stardinates - visualizing highly structured data. In *Proceedings of Seventh International Conference on Information Visualization (IV 2003)*, pages 47 – 52, july 2003.

[26] M. Lanzenberger, S. Miksch, and M. Pohl. Exploring highly structured data: a comparative study of stardinates and parallel coordinates. In *Proceedings. Ninth International Conference on Information Visualisation*, pages 312 – 320, july 2005.

[27] U. LAO. Lao national unexploded ordnance programme, 2011.

[28] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. Cleveland, S. Grannis, and D. Ebert. A visual analytics approach to understanding spatiotemporal hotspots. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):205 –220, march-april 2010.

[29] A. Murray, Y. Liu, S. Rey, and L. Anselin. Exploring movement object patterns. *The Annals of Regional Science*, pages 1–14, 2012. 10.1007/s00168-011-0459-z.

[30] D. Orellana, A. K. Bregt, A. Ligtenberg, and M. Wachowicz. Exploring visitor movement patterns in natural recreational areas. *Tourism Management*, 33(3):672 – 682, 2012.

[31] C. Plug, J. C. Xia, and C. Caulfield. Spatial and temporal visualisation techniques for crash analysis. *Accident Analysis and Prevention*, 43(6):1937 – 1946, 2011.

[32] L. N. U. O. Programme. Annual report 2007, 2008.

[33] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko. Visually driven analysis of movement data by progressive clustering. *Information Visualization*, 7(3-4):225–239, 2008.

[34] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2679–2688, 2012.

[35] C. Tominski, P. Schulze-Wollgast, and H. Schumann. 3D information visualization for time dependent data on maps. In *Proceedings. Ninth International Conference on Information Visualisation*, pages 175 – 181, july 2005.

[36] E. R. Tufte. *Beautiful evidence*, volume 23. Graphics Press Cheshire, CT, 2006.

[37] K. Vrotsou, J. Johansson, and M. Cooper. ActiviTree: Interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945 –952, nov.-dec. 2009.

[38] Wikileaks. Cablegate's cables, 2010.

[39] J. Zhao, F. Chevalier, E. Pietriga, and R. Balakrishnan. Exploratory analysis of time-series with ChronoLenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422 –2431, dec. 2011.

Author Index