TrendQuery: A System for Interactive Exploration of Trends

Niranjan Kamat
Computer Science & Engineering
The Ohio State University
kamatn@cse.osu.edu

Eugene Wu
Computer Science
Columbia University
ewu@cs.columbia.edu

Arnab Nandi
Computer Science & Engineering
The Ohio State University
arnab@cse.osu.edu

ABSTRACT

The surfacing of *trends* from data collections such as user-generated content streams and news articles is a popular and important data analysis activity, used in applications such as business intelligence, quantitative stock trading and, social media exploration. Unlike traditional content analysis, trend analysis includes an additional vital time dimension: a *trend* can be defined as a *temporal* pattern over a group of semantically related items. The unsupervised discovery of trends is often not sufficient, either due to inadequacies in the trend analysis algorithm, or because the data collection itself does not possess all of the information to identify the trend. Thus, it is necessary for an expert human-in-the-loop to be involved in the process of trend analysis.

To this end, we introduce *TrendQuery*, a system designed towards iterative and interactive surfacing of trends. Our system provides a set of trends to the expert, and enumerates iterative operations to curate the result. This process continues until the expert is satisfied with the surfaced trends. Since the space of possible tweaks to the result can be extremely large, the system continually provides feedback and guidance to the expert to prioritize possible operations. Our system allows interactive curation of trends providing better insights than a purely unsupervised approach.

1. INTRODUCTION

Due to the inherent historical nature of many data sources, computing measures such as item frequency over time is useful in a large number of domains – stock prices, sales amount, tweets, etc. Beyond inspecting temporal patterns for a single item, it is also interesting and useful to consider groups of items, such as semantically related words *"xquery"* and *"xml"*. We consider this generation of groups of terms that share similar temporal and semantic patterns as *trend analysis*. Analyzing trends over data is tremendously useful both from a business perspective and for knowledge gathering purposes.

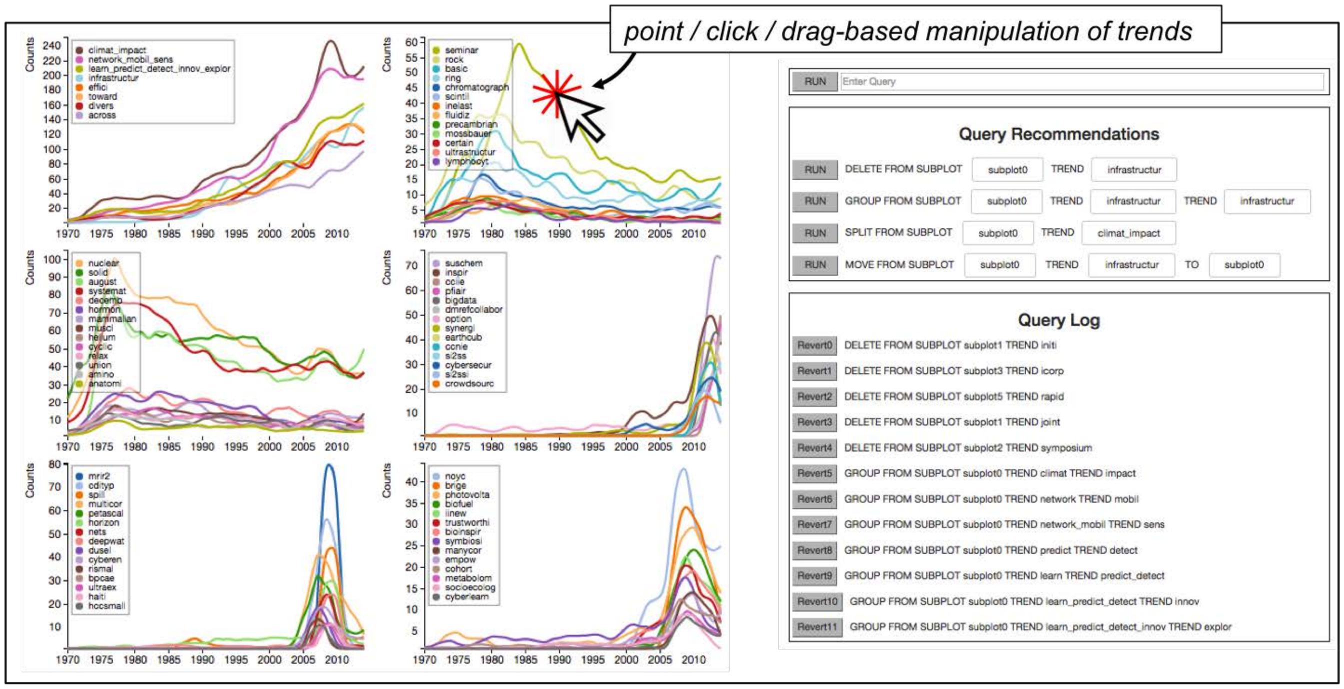
HILDA'16, June 26 2016, San Francisco, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4207-0/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2939502.2939514

Why is Interactivity Important in Trend Analysis?: Trend analysis consists of a series of steps, with each step having the potential to introduce errors or inaccuracies: Frequency data is inherently noisy and highly fluctuating. Feature construction and the algorithms built on top of them, although necessary, have the potential to model insight incorrectly and further induce errors. The limitations of the image display (screen resolution, color) further results in information loss. Presence of outliers is a recurring phenomenon in any type of data analysis, with smaller datasets worsening the signal to noise ratio as well. These different error sources have a compounding effect on the overall error. It is often too difficult or impossible to automatically fix these errors, e.g., constructing features and algorithms often require manual fine-tuning. Application of various time-series analysis techniques on erroneous data results in erroneous results as well. Thus, intervention by the user at each step has the potential to improve the quality of analysis results.

Secondly, the successful completion of any analysis task is marked by the user being satisfied with the analysis output. In the case of unsupervised algorithms, in the absence of ground-truth data, it is apparent that different users can have different preferences for output. In the trend detection paper by Twitter [10], the authors have noted the absence of a single best trend-detection algorithm with different approaches having different trade-offs. As in most of the mining tasks, tuning a number of parameters to obtain optimal values is an extremely difficult and time consuming task.

In these circumstances, we can improve the overall analytical process by giving the user the ability to inspect the results of the initial analysis and change them on the fly – enabling rapid integration of human input. Empowering the user in this fashion can thus be vital in interactive analytics. However, exposing all the tuning knobs of the trend analysis pipeline could overwhelm the user. Thus, providing exposing the right set of operations to the end user is critical for effective, interactive curation of trends. In this endeavor, *TrendQuery* provides a set of operations that the user can improve the result with and helps the user choose amongst them using a mixed-initiative user interface. This assists the users in fixing the errors, and cleaning the data, while improving the trend extraction output.

Trends occur at varying granularities in the data – from the minutiae of individual words to multiple words grouped together hierarchically under varying weights. Thus, trend analysis consists of interacting with the data at different resolutions. Therefore, the overall interactive trend analysis process will consist of the following two phases – offline



RUN Enter Query

Query Recommendations

RUN DELETE FROM SUBPLOT subplot0 TREND infrastructur

RUN GROUP FROM SUBPLOT subplot0 TREND infrastructur TREND infrastructur

RUN SPLIT FROM SUBPLOT subplot0 TREND climat_impact

RUN MOVE FROM SUBPLOT subplot0 TREND infrastructur TO subplot0

Query Log

Revert0 DELETE FROM SUBPLOT subplot1 TREND initi

Revert1 DELETE FROM SUBPLOT subplot3 TREND icorp

Revert2 DELETE FROM SUBPLOT subplot5 TREND rapid

Revert3 DELETE FROM SUBPLOT subplot1 TREND joint

Revert4 DELETE FROM SUBPLOT subplot2 TREND symposium

Revert5 GROUP FROM SUBPLOT subplot0 TREND climat TREND impact

Revert6 GROUP FROM SUBPLOT subplot0 TREND network TREND mobil

Revert7 GROUP FROM SUBPLOT subplot0 TREND network_mobil TREND sens

Revert8 GROUP FROM SUBPLOT subplot0 TREND predict TREND detect

Revert9 GROUP FROM SUBPLOT subplot0 TREND learn TREND predict_detect

Revert10 GROUP FROM SUBPLOT subplot0 TREND learn_predict_detect TREND innov

Revert11 GROUP FROM SUBPLOT subplot0 TREND learn_predict_detect_innov TREND explor

Figure 1: *TrendQuery* UI: Users can employ a direct manipulation approach to point / click / drag individual trend lines, performing actions such as deleting trends, moving trends from one subplot to another, grouping, splitting them, etc. Further actions & their illustrations are provided in Section 2.2. Recommendations are provided on the top right, and a query log is shown on the bottom right for debug / undo purposes.

analysis, followed by online iterative curation.

Offline: The temporal and semantic distances (Section 2) are first used to cluster the input. Most representative words in each cluster are presented, with each initial cluster occupying a different subplot. While we would like to investigate techniques such as topic modeling, biclustering, etc., clustering was found to give interesting results for our datasets.

Online: The user is now capable of manipulating the trends. As part of the online interactive phase, the initial trend data can be curated using actions presented in Section 2.2. The user can choose an action from the ranked list of action suggestions, or enter it in the interactive text editor manually, or use direct manipulation operations on the trends themselves. He continues operating in this interactive fashion till he is satisfied with the overall output.

Task Example: Consider John, a university grant supervisor, who wants to understand the trends in NSF research grants. He is presented with an initial clustered output using *TrendQuery*. He proceeds to delete glaring outliers from different subplots ($Q0$, $Q1$, $Q2$, $Q3$, $Q4$). Now ”climate” and ”impact” strike at him, which he groups ($Q5$). He detects another trend consisting of ”network”, ”mobil” and ”sens” ($Q6$, $Q7$). Next, he glances at numerous similar terms – ”learn”, ”predict”, ”detect”, ”innov”, ”explor” ($Q8$, $Q9$, $Q10$, $Q11$). At this stage, he is not quite sure what would be an appropriate course of action – maybe the first and the third trends need to be grouped together signifying increasing governmental focus on innovating and exploring climate change solutions or maybe the third trend is a general disparate trend. He changes his focus to other subplots, which in turn might help him better understand these trends as well. Interestingly, governmental efforts towards ”bigdata” and ”cybersecurity” also seem to be gathering steam over the last decade – providing another avenue for investigation. Figure 1 represents the *TrendQuery* UI at this exploration stage. Thus, using a sequence of actions, the user is not only able to clean the output and improve the result but also understand the result better in the process.

1.1 Contributions

Our contributions can be enumerated as follows:
1. We provide a principled database-inspired approach to operators for interactive curation of trends.
2. We provide a mixed-initiative interface, co-designed with and exposing our operators. An ordered list of query suggestions is provided to the user, which takes into consideration the query session, using well-defined metrics.

2. THE TRENDQUERY SYSTEM

The atomic unit of a trend consists of a *timeseries* representing its temporal and semantic information. The temporal component can be defined by the following `SQL` query: `SELECT term, year, COUNT(*) FROM trend_table GROUP BY term, year`.

A trend can be defined as a set of timeseries, grouped together in a hierarchical fashion with varying weights assigned to individual groups. We define the distance between two trends using linear combination of their temporal and semantic distance due to their orthogonality as follows:

$$dist(t1, t2) = \alpha \times temp(t1, t2) + \beta \times sem(t1, t2) \qquad (1)$$

Metric learning [13], based on user actions, can then be used to learn these parameters. *TrendQuery* supports different quickly computable temporal metrics such as euclidean [12], shape-based [20], and cDTW [23]. We currently use the largest synset path similarity to estimate the semantic similarity [2, 25]. We would like to investigate different semantic metrics more thoroughly in the future, in conjunction with their combination with temporal metrics.

2.1 System Architecture

Our system consists of a trend generation backend, and an interactive frontend to curate them. The frontend consists of subplots for displaying different trends, ranked list of action suggestions, an editor to input the next query manually, and the query log. Some components are briefly explained below.

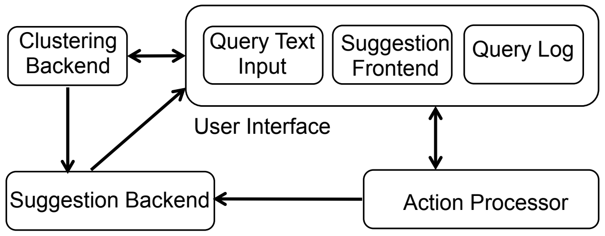


Figure 2: System Architecture

Trend Clustering Backend: The input distance matrix, consisting of pairwise distances (Equation 1), is provided to a clustering algorithm. During the query session, the user can also trigger a new clustering to be performed which takes into consideration the user preferences thus far.

Action Suggestion Frontend: A ranked list of suggestions is presented in a parameterized form via a drop down menu. A novel ranked parametrization technique is used to increase number of possible suggestions (Section 2.3.3).

Action Processor: This module performs the action, updates the subplots, and calls the *Action Suggestion Backend* to update the list of suggestions.

Action Suggestion Backend: This module enumerates the different possible actions and ranks them (Section 2.3).

2.2 Manipulation Operations

In this section, we list the available actions – how to perform them using the point and click approach or an actual query. *TrendQuery* uses a one-to-one mapping between user input and actions. Although this reduces the flexibility of user actions, it increases robustness and helps provide the user with a better intuitive idea of their effect. In our preliminary user studies, we found that users were able to quickly mentally map user actions with the corresponding operations. We believe our operations constitute building blocks of any interactive trend manipulation system. Further, they were found to be expressive enough to detect trends and clean the resultset in our preliminary user studies.

Delete: This action can be performed on an outlying *timeseries*, by right clicking the trend.
Query: `DELETE [FROM SUBPLOT subplot] TREND[S] [X | WHERE {OUTLIER_SCORE > threshold | OUTLIER_RANK < rank}]`.

Move: If a user believes that a timeseries is better suited to being in another subplot, he can drag it to the other subplot.
Query: `MOVE FROM SUBPLOT subplot_a TREND x TO subplot_b`.
A timeseries having a high outlier score indicates that it might be a good candidate for either the *Delete* or *Move* actions. While more complex approaches such as ROF [8], LOCI [19], etc. can be used for this purpose, we use the Mahalanobis distance and the corresponding Chi-Square test due to its speed of computation.

Group: If the user considers different timeseries within a subplot to be similar, he can group them together by clicking one, followed by the other.
Query: `GROUP [FROM SUBPLOT subplot] TREND[S] [X[,...]] [WHERE {SIMILARITY_SCORE > threshold} | {SIMILARITY_RANK < rank}]`.

Split: The user can split a group by middle clicking it.
Query: `SPLIT [FROM SUBPLOT subplot] TREND[S] [X[,...]] [WHERE {SIMILARITY_SCORE < threshold} | {SIMILARITY_RANK > rank}]`.

Importantly, each of these actions are can be undone. Thus, the overall session results in being undoable as well without needing to save the entire program state. For example, a *Group* action has a corresponding *Split* action. This is extremely helpful as undoing actions is vital in analytics. In the future, we would like to improve our action set by enabling user defined functions.

2.3 Action Suggestion

The manipulation operations have two orthogonal effects – they improve the quality of trends and change the system state. These effects are taken into consideration by the *Action Suggestion Backend* module in order to improve suggestions, using the query session. This module consists of the submodules, *Action Enumeration* and *Action Ranking.*

2.3.1 Action Enumeration

For each action type, all possible query instantiations are first enumerated. For example, the *Delete* action can be applied to every trend and is, thus, linear in the number of trends. However, for some action types, *Group* and *Move*, number of possible applications can be exponentially large. Hence, we use a novel query ranking based parameterization technique in order to successively expose possible queries based on the currently entered incomplete user query (Section 2.3.3). Note that more complex instantiations of these actions can be constructed iteratively, in line with our underlying principle of iterative processing. We provide below the list of possible applications for different action types.

Action	Enumerated List
Delete	All timeseries
Move	All (timeseries, subplot) pairs
Group	All timeseries pairs
Split	Each grouped trend

Table 1: Enumerated Actions

2.3.2 Action Ranking

Query Session and Effect Localization: Given our operation set, we can notice that specifying the subplot helps reduce the action scope, whereas, using a `WHERE` predicate and not providing specific trends can result in increased scope. As the session progresses, a user's understanding of the data and the underlying trends keeps improving. Therefore, we prefer for the user to employ actions with smaller, localized effects initially, and followed later by more complex ones. Hence, we initially lower the score of global actions, and progressively reduce this dampening effect.

User Cues: Understanding the user intent is crucial in improving the query suggestions. However, quantifying this abstract notion is inherently difficult. We use the two following orthogonal cues to do so:
1. Subplot: A user focusing on a subplot indicates his dissatisfaction with its current state and, hence, the associated operations are given a higher priority.
2. Action Type: Greater importance is also given to the action types recently used by the user. For example, if he employs the *Group* action, it indicates that the plots currently possess a higher granularity. Similarly, if the user removes a trend, it indicates a preponderance of outliers prompting higher priority for *Delete* actions.
Using cues about the user preference for plots, and action types, and localization effects, different enumerated actions are ranked, in a similar fashion as Wrangler [11] and DataPlay [1]. An interesting opportunity exists here to explore better integration between interaction gestures and the underlying algebra beyond our form-based interface.

2.3.3 *Query Parameterization*

Numerous queries exist which differ by a single parameter. For example, the `DELETE` queries can be parameterized as `DELETE FROM SUBPLOT {subplot_0 ... | subplot_n} TREND {trend_0, ...trend_m}`, where the trend instances are varied based on the subplot selected in the earlier part of the query specification. This simple technique of updating the successive parameter helps us provide the user with a larger number of ranked queries while expending fewer pixels.

3. RELATED WORK

Analysis of trend data is a popular analytical task. Twitter has published their work on trend detection [10], attempting to discern rise in new trends over differing timescales. TwitterMonitor [17] allows users to interact with the system by using different criteria to order trends. In contrast with these systems, we present the idea of *iterative*, *exploratory*, and *interactive* analysis with a human-in-the-loop, which can reside on top of any of these systems.

The efficacy of iterative data cleaning and preparation has been well illustrated by Potter's Wheel [21], Wrangler [11], and DataXFormer [18]. While some other tools like Bellman [6] help users understand the quality and structure of the database, others such as Toped++ [24] transform the data as well. Unlike these tools, our system uses the clustering output, which is highly unstructured, as its input data.

There has been extensive review of visual query systems by Catarci et. al. [4] and El-Mahgary, et. al. [7]. DataPlay [1] has enabled interactive tweaking of SQL queries using a graph view of databases. From a text processing perspective, VINERy [15] introduces the concept of interactive tweaking using rule generation for information extraction. Vegemite [16] uses direct manipulation and programming-by-demonstration to populate tables with information from disparate sources. Chiticariu et al. [5] present easy-to-use capabilities for refining schemas. This line of work has influenced our direct trend series manipulation actions.

There have also been efforts to incorporate user interaction into clustering [3,22], especially in domains such as gene analysis in bioinformatics (BiCluster Viewer [9]) and document exploration (Lighthouse [14]). However, unlike this work we not only provide user actions but also query ranking in an easy to use interface. Recent work by Paparrizos et al. [20] has improved temporal clustering.

4. CONCLUSION & FUTURE WORK

We have presented the *TrendQuery* user interface and backend. It allows users to analyze trends in time-series data using a mixed-initiative user interface. The user actions and the grammar for the corresponding queries provide a powerful tool for interactive curation of trends. In the future, we would like to perform a comprehensive user study to analyze not only the helpfulness of our frontend, but also the effectiveness of our query suggestion backend. We plan on evaluating our system by comparing output a data scientist provides (ground truth) with those from numerous *TrendQuery* users. We plan on incorporating incremental mining algorithms during online curation. We would also like to expand our grammar to the underlying clustering algorithms.

5. REFERENCES

[1] A. Abouzied, J. Hellerstein, and A. Silberschatz. DataPlay: Interactive Tweaking and Example-Driven Correction of Graphical Database Queries. *UIST*, 2012.
[2] S. Bird. NLTK: The Natural Language Toolkit. *COLING/ACL*, 2006.
[3] J. R. Brandt, J. Chong, and S. Rosenbaum. Interactive Clustering for Data Exploration.
[4] T. Catarci et al. Visual Query Systems for Databases: A Survey. *JVLC*, 1997.
[5] L. Chiticariu, P. G. Kolaitis, and L. Popa. Interactive Generation of Integrated Schemas. *SIGMOD*, 2008.
[6] T. Dasu et al. Mining Database Structure; or, How to Build a Data Quality Browser. *SIGMOD*, 2002.
[7] S. El-Mahgary et al. A Form-Based Query Interface for Complex Queries. *JVLC*, 2015.
[8] H. Fan, O. R. Zaïane, A. Foss, and J. Wu. A Nonparametric Outlier Detection for Effectively Discovering Top-N Outliers from Engineering Data. *KDD*, 2006.
[9] J. Heinrich et al. BiCluster Viewer: A Visualization Tool for Analyzing Gene Expression Data. *ISVC*, 2011.
[10] S. Hendrickson, J. Montague, J. Kolb, and B. Lehman. Trend Detection in Social Data. *Twitter Blog*, 2015.
[11] S. Kandel et al. Wrangler: Interactive Visual Specification of Data Transformation Scripts. *SIGCHI*, 2011.
[12] E. Keogh et al. Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research. *Knowledge and Information Systems*, 2005.
[13] B. Kulis. Metric Learning: A Survey. *Foundations and Trends in Machine Learning*, 2012.
[14] A. Leuski and J. Allan. Lighthouse: Showing the Way to Relevant Information. *InfoVis*, 2000.
[15] Y. Li, E. Kim, M. A. Touchette, et al. VINERy: A Visual IDE for Information Extraction. *PVLDB*, 2015.
[16] J. Lin, J. Wong, J. Nichols, et al. End-User Programming of Mashups with Vegemite. *IUI*, 2009.
[17] M. Mathioudakis and N. Koudas. TwitterMonitor: Trend Detection over the Twitter Stream. *SIGMOD*, 2010.
[18] J. Morcos, Z. Abedjan, I. F. Ilyas, et al. DataXFormer: An Interactive Data Transformation Tool. *SIGMOD*, 2015.
[19] S. Papadimitriou et al. Loci: Fast Outlier Detection Using the Local Correlation Integral. *ICDE*, 2003.
[20] J. Paparrizos and L. Gravano. k-Shape: Efficient and Accurate Clustering of Time Series. *SIGMOD*, 2015.
[21] V. Raman and J. M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. *VLDB*, 2001.
[22] M. Rasmussen et al. gCLUTO: An Interactive Clustering, Visualization, and Analysis System. *UMN-CS TR-04*, 2004.
[23] H. Sakoe et al. Dynamic programming algorithm optimization for spoken word recognition. *ICASSP*, 1978.
[24] C. Scaffidi et al. Intelligently Creating and Recommending Reusable Reformatting Rules. *UIST*, 2009.
[25] T. Slimani. Description and Evaluation of Semantic Similarity Measures Approaches. *arXiv*, 2013.