Efficient Sentiment Correlation for Large-scale Demographics

Mikalai Tsytsarau
University of Trento, via Sommarive 14, Trento, Italy
tsytsarau@disi.unitn.eu

Sihem Amer-Yahia
CNRS, LIG, France
Sihem.Amer-Yahia@imag.fr

Themis Palpanas
University of Trento, via Sommarive 14, Trento, Italy
themis@disi.unitn.eu

ABSTRACT

Analyzing sentiments of demographic groups is becoming important for the Social Web, where millions of users provide opinions on a wide variety of content. While several approaches exist for mining sentiments from product reviews or micro-blogs, little attention has been devoted to aggregating and comparing extracted sentiments for different demographic groups over time, such as 'Students in Italy' or 'Teenagers in Europe'. This problem demands efficient and scalable methods for sentiment aggregation and correlation, which account for the evolution of sentiment values, sentiment bias, and other factors associated with the special characteristics of web data. We propose a scalable approach for sentiment indexing and aggregation that works on multiple time granularities and uses incrementally updateable data structures for online operation. Furthermore, we describe efficient methods for computing meaningful sentiment correlations, which exploit pruning based on demographics and use top-k correlations compression techniques. We present an extensive experimental evaluation with both synthetic and real datasets, demonstrating the effectiveness of our pruning techniques and the efficiency of our solution.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining — *correlations*

Keywords

sentiment analysis; sentiment demographics; correlation;

1. INTRODUCTION

Today, sentiment analysis has become a platform that provides valuable information on people's opinions regarding different topics, and is widely used by businesses [5] and social study institutions [17]. Sentiment extraction and aggregation has been applied in various domains, from *movie reviews* to product *reputation management*. While multiple efforts focused on developing machine learning and statistical methods for characterizing sentiment within large bodies of text or for brief opinions and tweets [11, 14], not much attention has been devoted to aggregating sentiment along users' demographics [15]. Studies along this direction have traditionally focused on an off-line analysis and aggregation of polling data for pre-determined demographic groups. However, it is difficult to organize polling on a large scale and conduct it regularly enough to analyze trends and correlations over time.

Nevertheless, the need to provide fine-grained analytics of social data is growing. Readily available users' demographics along with opinion data constitute a gold mine for extracting insights on what a particular user group thinks and how their opinion evolves over time and compares to opinions of others.

Some recent studies have already made a step along this direction. For instance, "A Demographic Analysis of Online Sentiment during Hurricane Irene" [6] revealed dynamic (temporal) sentiment differences between Southern USA and New England, and at the same time a constant (inherent) difference in the sentiments expressed by males and females, referred to as *sentiment bias*. Indeed, different demographic groups may have different points of reference when they express their sentiments for different topics. For example, while youngsters tend to prefer relatively cheap restaurants and are comfortable with a certain level of noise, pensioners generally prefer quieter and moderately priced restaurants. In addition, sentiments of demographic groups may evolve differently over time. For example, "French farmers" and "German farmers" had initially positive sentiments for the topic "organic farming", but later disagreed when the French government introduced additional taxes for organic goods superseding laws set by the European union and in disfavor of French farmers. In the above example, if the two groups have equal average sentiments for a specific topic during some time period, it will be hard to say if they really have the same attitude towards the events in that period, or if their equal sentiments are merely the result of an instantaneous convergence of otherwise diverse sentiments.

Our examples suggest the need for sophisticated methods that can compare and correlate sentiments of demographic groups over time regardless of their inherent biases. One important aspect in the design of appropriate methods is the definition of sentiment correlation as a function of aggregated sentiment over a time period. We explore Pearson's correlation using several variations of the average sentiment. Second, it is important to use efficient correlation methods, which allow online updates. Here again, we evaluate several ways of constructing a time interval of sentiment correlations and show that correlations remain meaningful and robust to noise when time intervals are assembled from smaller ones, which allows to apply efficient top-k and windowed correlation methods.

There are two computational challenges when implementing our methods: 1) finding demographic groups requires the exploration of all possible combinations of values for demographics attributes; 2) in order to find correlations between pairs of demographic groups, one potentially needs to explore all sub-intervals of the input time interval. We show that both challenges render traditional database

SIGMOD'13, June 22–27, 2013, New York, New York, USA.

approaches inefficient, and describe algorithms that exploit the *lattice structure induced by demographics attributes* in order to prune the search space. Our algorithms also make use of *hierarchical time aggregation* to achieve efficient and scalable indexing and retrieval of aggregated sentiments. The contributions of this work can be summarized as follows:

- We formalize a novel large-scale sentiment analytics problem, focusing on the efficient aggregation of sentiment and computation of high and significant sentiment correlations between maximal demographic groups within dynamically determined time intervals.
- Specific to demographics sentiments, we describe efficient correlation pruning methods based on the demographics lattice. Furthermore, we introduce two novel methods for correlation compression, which allow for the efficient implementation of our algorithms.
- We conduct an extensive set of experiments to validate our problem, and evaluate the performance of our solution. We use synthetic datasets, which contain large-scale artificial correlations with added noise, and the MovieLens real dataset, which comes with rich user demographics. The experiments demonstrate that correlated demographic groups can be identified very efficiently with the help of our specialized indexing storage and effective pruning. Finally, our evaluation provides interesting insights on correlations among real demographic groups in MovieLens.

This paper is organized as follows. Section 2 defines our framework and the problem we tackle, while Section 3 develops the properties of correlation with respect to our problem. Section 4 describes our algorithms for correlated groups and correlation compression. Our user study and performance experiments are reported in Section 5. Section 6 provides a summary of the related work. We conclude in Section 7.

2. FORMALISM

We are given a database $\mathscr{X}$ of records $x = (u,t,s,p)$ where $u \in \mathscr{U}$ denotes a user expressing sentiment $s \in$ [-1,1] on a topic $t \in \mathscr{T}$ in a time period p characterized with a start and end timestamps.

In our definitions we assume that sentiments are extracted for a given topic. For example, the record $x_1 = (u_1, Politics, 0.8, p_1)$ means that user u_1 expressed a positive sentiment (i.e., +0.8) for "Politics" during time period p_1. Such information can be extracted from the tweets of user u_1 during that time period. The record $x_2 = (u_2, Drama, -0.5, p_2)$ expresses a negative sentiment for "Drama" movies by user u_2 during time period p_2. This information can be computed from movie rating datasets such as MovieLens.

2.1 Definitions

We assume that each user $u \in \mathscr{U}$ is associated with a collection of values for a set of demographics attributes $\{a_i\}$. For example, *25* for attribute a_1 `:age`, *Student* for a_2 `:occupation`, and *Italy* for a_3 `:location`. Each attribute a_i is associated with a demographics hierarchy $\mathscr{D}_i$ whose nodes hierarchically partition the set of values for that attribute. Correspondingly, each demographics hierarchy node contains all users from $\mathscr{U}$ whose attribute values are covered by that node. The top left part of Figure 1 shows an example demographics hierarchy associated with attribute `location`. The top node covers users from all available geographic locations (corresponding to $\mathscr{U}$), and the descendant nodes partition those users into non-intersecting subsets according to their geographic locations.

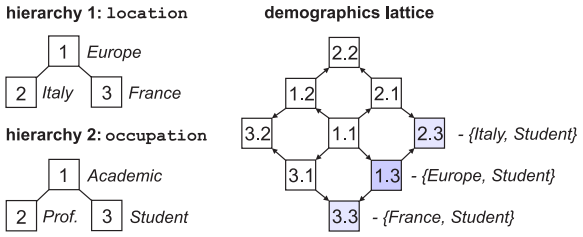

Figure 1: Two demographics hierarchies forming a lattice.

Definition 1. (Demographics Criteria) $\boldsymbol{d} = \{a_1 = d_1, \cdots, a_k = d_k\}$ is a set of predicates over demographics attributes a_i, where each predicate requires attribute's values to be contained in a node from a demographics hierarchy.

For example, $\boldsymbol{d} = \{$`age` : *Young*, `location` : *Italy*, `occupation` : *Student*$\}$ refers to a combination of predicates on user attributes `age`, `location` and `occupation`. To simplify our notation, we will use $\boldsymbol{d} = \{Young, Italy, Student\}$ to refer to the same set of predicates. Values in demographics criteria correspond to hierarchy nodes and are therefore a fixed set, which can be enumerated. We note that user attributes with continuous values (for example, `age`) can be transformed to categorical values in order to induce a hierarchy.

Definition 2. (Demographics Generality) Demographics criteria $\boldsymbol{d}$ is more (less) general than $\boldsymbol{d'}$ if: $\forall\,(d_i, d'_i):\quad d'_i \in d_i \quad (d_i \in d'_i)$. We denote these relationships as $\boldsymbol{d} \prec \boldsymbol{d'}$ $\quad(\boldsymbol{d} \succ \boldsymbol{d'})$.

All demographics criteria and their generality relationships form a *demographics lattice* L, with a size equal to the product of the hierarchies' sizes. We show an example of such a lattice in the right part of Figure 1, where a nine-element demographics lattice is formed by all node combinations of two hierarchies, each containing three nodes (shown left). The links that go from one lattice node to another indicate generality relations. For example, such criteria as $\{Italy, Student\}$ and $\{France, Student\}$ are less general than $\{Europe, Student\}$, which is itself less general than $\{Europe, Academic\}$.

Definition 3. (Demographic Group) $\mathscr{U}^{\boldsymbol{d}}$, defined by demographics criteria $\boldsymbol{d}$, is a set of users $u \in \mathscr{U}$, who satisfy predicates in $\boldsymbol{d}$.

An example of demographic group is European students defined by the demographics criteria $\{Europe, Student\}$, shown in Figure 1, right. In this paper, we only consider groups of users, that can be defined using demographics criteria and use demographics criteria to denote demographic groups in all our equations.

Definition 4. (Group Sentiment) Given a demographics criteria $\boldsymbol{d}$, a topic t and a time period p, we define the group sentiment of $\mathscr{U}^{\boldsymbol{d}}$ as an aggregation of sentiments s_x over records $x = (u_x, t_x, s_x, p_x)$ where $u_x \in \mathscr{U}^{\boldsymbol{d}}$, $t_x = t$, s_x is the sentiment of u_x for t, and $p_x \in p$: $s(\boldsymbol{d}, p) = \frac{1}{|x|} \sum s_x \mid \{u_x \in \mathscr{U}^{\boldsymbol{d}}, p_x \in p\}$.

In the rest of the paper, we assume that sentiments are aggregated and analyzed with respect to the same topic t, and therefore we omit t where applicable.

The main scope of this paper is to analyze time-behavior of sentiment. Therefore, we need to consider a time series of sentiments, aggregated using fixed intervals to allow meaningful comparisons of individual points within, as well as across time series.

Definition 5. (Sentiment Time Series) is a sequence of values s_i computed by aggregating sentiments on a time interval p, using fixed sub-intervals p_i of the same length: $s_i = s(\boldsymbol{d}, p_i)$.

Our further analysis of sentiment dynamics is centered on sentiment time series for a given group, topic and time period. Before going into details about the right time granularity and a sentiment correlation function $\rho()$, we consider the relations between demographic groups, and discuss how they affect the formulation of our problem.

Definition 6. (Maximal Demographic Group) Given a sentiment time series similarity function $\rho()$, a threshold θ and a time period p, we call a demographic group $\mathscr{U}^{\boldsymbol{d}}$ maximal, if and only if: $\nexists\, \boldsymbol{d}' \prec \boldsymbol{d},\ s.t.\ \rho(\boldsymbol{d}, \boldsymbol{d}', p) > \theta$.

Intuitively, the above definition says that a demographic group is maximal if there is no other, more general group, that for the time period of interest shares the same sentiment behavior with the given demographic group. We define maximal demographic groups with respect to their sentiment time series similarity, by analogy to the definition of maximal itemsets in frequent itemset mining.

Demographics relations may also be of a partial-overlap type, e.g., between $\{Europe, Students\}$ and $\{Italy, Academic\}$ (where *Europe* is a superset of *Italy*). However, we can argue that while for negative correlations all relationships between groups can be interesting, for positive correlations, generality and partial-overlap relations represent trivial cases of sentiment dependency. Sentiment correlations in this case can be caused by aggregating the same sentiments from the overlap for both groups. Therefore, we need to consider a disjoint type of relation.

Definition 7. (Demographics Disjointness) between two demographic criteria $\boldsymbol{d}$ and $\boldsymbol{d}'$ is strictly opposite to demographics generality: $\exists (d_i, d_i') : d_i \cap d_i' = \emptyset$. We denote these relationships as $\boldsymbol{d} \not\oslash \boldsymbol{d}'$.

In other words, disjointness on any of the attributes makes the entire criteria also disjoint. Based on the above observations, for the rest of this paper we limit the scope of possible relations to those between non-overlapping groups and fully-overlapping groups only.

2.2 Problem Definition

In this paper we are interested in finding strong and significant positive and negative correlations among the sentiments time series of demographic groups. For the sake of simplicity, we will only work with positive thresholds, specifying the sign of the correlation if needed.

Problem Statement. (Correlated Sentiment) Given a period of time p and correlation ρ_{min}, find pairs of maximal disjoint demographic groups $\{\boldsymbol{d}, \boldsymbol{d}'\}$ and the longest time interval $p' \in p$ where their sentiments correlate: $|\rho(\boldsymbol{d}, \boldsymbol{d}', p')| > \rho_{min}$.

Note that we can further restrict the answer set to only contain demographic groups whose correlation is statistically significant. That is, we require that $|\rho(\boldsymbol{d}, \boldsymbol{d}', p')| > \rho_{min}$ at a significance level r_{min}. We discuss this issue in more detail in the following sections.

A proper solution to the above-formulated problem will allow identifying sentiment behavior at a much finer level of detail than currently possible, finding cases that are counter-intuitive and can only be observed by processing huge amounts of data. We further discuss some interesting examples of such findings in Section 5, where we report the results of applying the proposed approach to the analysis of movie opinions.

3. SENTIMENT CORRELATION

Similarity of sentiments between two demographic groups is measured using a correlation coefficient of their sentiment time series.

Definition 8. (Sentiment Correlation) Correlation ρ of two time series s and s' of length n (with averages $\overline{s}$ and $\overline{s}'$) is defined as the normalized inner product $(s \circ s')_1^n$ of local deviations from averages:

$$\rho = \frac{(s \circ s')_1^n}{n \sigma_s \sigma_{s'}} = \frac{\sum_{i=1}^{n} (s_i - \overline{s}) \cdot (s_i' - \overline{s}')}{\sqrt{\sum_{i=1}^{n} (s_i - \overline{s})^2 \cdot \sum_{i=1}^{n} (s_i' - \overline{s}')^2}}$$

Indices $_1^n$ in the inner product denote that it is computed using data points from 1 to n. Correlation ρ takes values in the interval $[-1, 1]$, where a positive (resp., negative) sign indicates that sentiments are changing in the same (resp., opposite) way and the absolute value measures the strength of the correlation.

Sentiment average can be computed in different ways in order to reflect different dependencies between time series. In addition, the period of time during which correlation is computed may vary depending on on whether the whole period is considered at once or if it is processed into sub-intervals. In the next two sub-sections we discuss different sentiment average computation and interval processing. We use the examples in Figure 2.

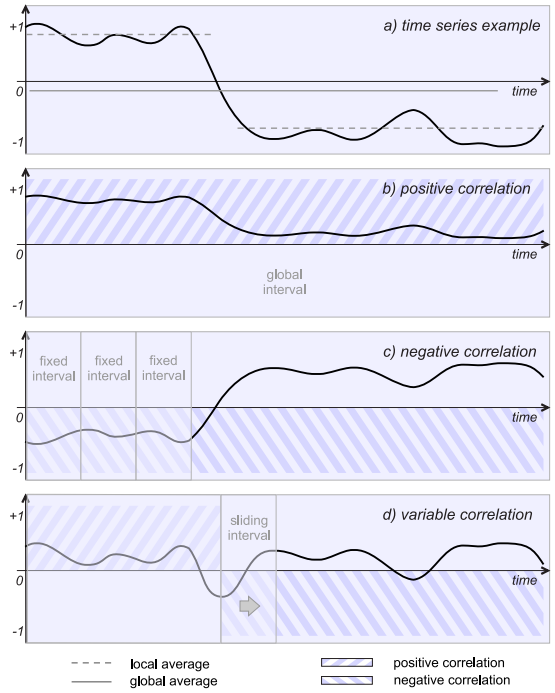

Figure 2: Examples of different correlation types.

3.1 Sentiment Averages

Global Average: This correlation detects co-variation of sentiments regardless of their sentiment bias. In Figure 2, a strong positive correlation between a and b is detected even though time series b is entirely positive.

Zero Average: When an average is substituted with a zero value, the correlation formula detects *polarity correlation* (e.g., between a and c in Figure 2). Polarity correlation indicates a much stricter dependency between sentiments: not only their local deviations, but also their signs (polarity) should be synchronized. As an additional benefit, it is easier to compute, as it does not need to compute average values of sentiments.

Local Average: If we compute correlation with local average (shown as dashed grey lines in time series a), we are able to detect correlation between local deviations. For example, the time series d has the same deviations of sentiment as the series a during the first period, and inverse deviations during the second period. These periods can be detected by computing correlation using sliding windows with the running average.

3.2 Time Intervals

Global Interval: We compute a single correlation value for the entire input time interval.

Fixed Interval: The input time interval is divided into fixed-length sub-intervals within which correlations are computed.

Sliding Interval: The input time interval is divided into variable-length sub-intervals in a way that maximizes correlations. This goal can be achieved by optimizing sizes of correlation intervals, or by using a greedy algorithm that identifies intervals on-the-fly.

We observe, that for very long time intervals, global average sentiment is likely to be close to zero, so global average and zero average effectively become the same. The same is true for local average, which becomes closer to the global one with increased interval sizes. We will use the above variations of the sentiment correlation in our algorithms, which we describe in the following sections.

3.3 Correlation Significance

When computing the correlation value between two time series, we also need a measure that expresses how confident we are that this value accurately captures reality (e.g., we can be more confident that the correlation value between two particular time series is true when these time series are long than when they are comprised of just a few data points). This measure is the correlation significance. Given the correlation coefficient ρ, computed from n samples, we consider two *Null Hypotheses* about the correlation, which have the following test statistics:

Hypothesis H1 ($\rho = 0$): the value $z = \rho\sqrt{(n-2)/(1-\rho^2)}$ is distributed as the *t-distribution* with $(n-2)$ degrees of freedom.

Hypothesis H2 ($\rho < \rho_{min}$): the value $z = (Z(\rho) - Z(\rho_{min}))\sqrt{n-3}$ is distributed as the *standard normal*, where $Z(\rho) = artanh(\rho)$.

Definition 9. (Correlation Significance) r is defined as the probability at which the considered null hypothesis is supported. For large n, the (one-tailed) significance of H1 and H2 is computed using the cumulative distribution function of the standard normal:

$$r = 1 - \Phi(z) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{x^2}{2}} dx$$

While the first hypothesis is intended to verify if there exists any correlation between the two time series, the second hypothesis tests if they are correlated at least as high as ρ_{min}. For example, if $\rho = 0.9$, $\rho_{min} = 0.7$ and $n = 10$, we have that $r_{H1} \approx 0.0002$ and $r_{H2} \approx 0.06$. Relying on the significance threshold $r_{min} = 0.05$, H1 can be rejected as improbable, meaning that the two time series are indeed correlated. On the contrary, H2 cannot be rejected, signifying that ρ can be smaller than 0.7 and thus it is subject for pruning.

4. METHOD AND ALGORITHMS

In this section, we present our methods for storing sentiment time series for demographic groups and efficiently extracting correlations. We begin with the outline of our sentiment storage, followed by a description of our algorithms. Finally, we present smart pruning and compression techniques which take a full advantage of our storage and allow efficient problem solving.

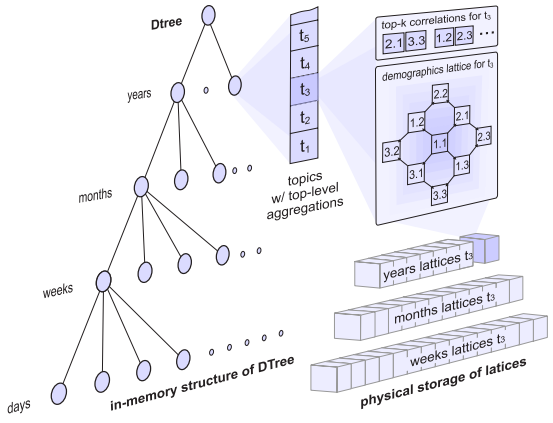

Figure 3: The DTree, a time-indexed sequential storage of aggregated demographics sentiments over multiple topics.

4.1 Data Storage and Management

Our problem requires ad-hoc navigation of time series at different aggregation granularities, and fast access to sentiments for demographic groups. Therefore, it makes sense to organize the data storage around a time-indexed and aggregating structure named *Demographics Tree (DTree)*, which at its nodes provides access to aggregated sentiment values via the demographics lattice. We demonstrate this structure in Figure 3 and describe below.

DTree is a hierarchically organized balanced tree, where each level in the hierarchy stores information relevant to years, month, weeks, and days. Each node in the tree corresponds to one of these intervals, and is connected to the parent and children nodes in the hierarchy, as well as to the adjacent nodes at the same level. Each DTree node stores statistical aggregations of sentiments for different topics for the specific time interval: *(count, sum, sum of squares)*$_t$, where topic $t \in \mathscr{T}$. These aggregations allow us to reconstruct sentiment mean, variance, volume and their derivatives, and they are also incrementally maintainable, allowing the easy update of the DTree as new data come in. In addition, DTree nodes store top-k correlations for the particular time interval and topic, in order to facilitate query answering. We provide more details on the construction of top-k correlations in Section 4.3. DTree nodes maintain physical aggregations only for the top-level demographic groups for each topic (e.g., only for group (1.1) in Figure 3). Detailed aggregations for all individual groups are accessible by following a pointer to a separate structure, the sequential file storage for lattices. This pointer indicates an offset in the file that contains the demographics lattice snapshot with the aggregations for all demographic groups for the particular topic and time interval. By traversing this sequential file storage structure, we can simultaneously reconstruct the sentiment time series for all demographic groups for a particular topic and time aggregation level.

Thanks to this layout, a time index with high-level aggregates and pointers remains compact and can be kept in main memory (Figure 3, left), while sentiment time series can be organized as a collection of individual files (Figure 3, right). The additional benefit of this organization is that it ensures fast sequential access for time series of sentiments, compared to relational databases [16].

We note that the number of sentiment values monotonically decreases as we navigate down the demographics lattice and down the DTree levels, so many of the demographics leaf nodes will contain zeroes at lower time granularities. This allows storing sentiment values in a more compact way, by storing only non-zero values (e.g., using run-length encoding methods [19]).

4.2 Overview of Algorithms

Our algorithm for extracting significant correlations is based on the DTree storage and is listed in Algorithm 1. Due to lack of space, we present our algorithm for the sliding time interval and conventional correlation formula (see Section 3). The other approaches, using fixed or global intervals, can be reduced from it by considering subsequent time intervals of constant sizes (fixed), or the entire time series (global).

The process of mining sentiment time series is performed in a top-down fashion, going from higher to lower granularities in a DTree and from root to leaf nodes in a demographics lattice. Our approach achieves hierarchical correlation management and pruning through remembering for every pair of groups which of the identified time intervals to `refine` or `exclude` at the next granularity level. Lines 13 and 16 of the algorithm only show the invocation of these functions, assuming that the corresponding time interval pruning takes an immediate effect on the next iteration.

In the second loop, we sequentially access the time series, while computing correlations between demographic groups in the third loop, where candidates are evaluated going from higher order nodes to their children. If a high correlation is detected for some candidate groups (line 12), then their positively correlated children are excluded, meaning that the corresponding lattice branches are not revisited in subsequent iterations of the loop. This pruning asserts the maximality of identified correlations and also reduces the candidate set.

The described algorithm employs the sliding interval approach, where correlation interval boundaries are determined in a greedy fashion: by comparing correlation coefficients between a forward (sliding) interval w_{next} of a fixed size, and an interval that runs from the previous boundary w_{prev}. We note, that while the sliding time interval w_{next} is updated for all demographics pairs as we scan the time series, the correlation time intervals w_{prev} are computed and maintained for each demographics pair individually, and their starting boundaries do not necessarily coincide (however, all their ending boundaries border with w_{next} while it slides). When global or fixed time interval approaches are used, it is possible to prune candidate demographic groups on-the-fly according to their estimated value of correlation, so that less and less computations are needed as we advance along the time series.

Finally, for all detected pairs of groups and correlation intervals, the algorithm can start the greedy generalization step, described in Algorithm 2. It iteratively supersedes groups with their maximal parents until they are disjoint and highly correlated.

Computing and storing correlation coefficients for all combinations of demographics nodes is only possible for small lattices, since it requires a quadratic space on the size of a lattice. But since we are interested in finding only high and significant correlations, it is possible to compute and store only such values, while still being able to answer queries with a good precision. In the following sections, we describe how correlation pruning and compression enable efficient implementation of our method.

4.3 Computing Correlations

In this section we describe an efficient way for computing correlations: first, by discarding insignificant results, and, second, by discarding correlations of children groups according to the maximality principle given in Definition 6. Furthermore, we propose efficient methods of storing precomputed correlation values for fixed time intervals. We note that the proposed hierarchical pruning and correlation compression methods are applied on top of correlation values, and can be used in combination with various correlation algorithms. Some existing correlation methods [21, 7] can also be applied to our case, but are otherwise orthogonal to the pruning and compression methods discussed in this paper.

Algorithm 1: Sliding algorithm for discovering sentiment correlations. Employs pruning using correlation estimates based on Lemma 1 and significance threshold based on Lemma 2.

Input : Time interval p, significance r_{min}, correlation ρ_{min}, lattice L, sliding interval size m
Output: demographic groups and correlation intervals

```
1  for granularity = max ... 1 do
2      for p_i = {p_1 ... p_n} ∈ p do
3          w_next = w_next + p_i − p_{i−m}; //push next, pop last
4          //w_prev are individual for each candidate
5          //candidates are ordered by height(d, d') top-down
6          for (d, d') ∈ L × L | d ⊅ d' do
7              w_prev = w_prev + p_{i−m}; //update previous interval
8              ρ_prev = correlation(d, d', w_prev);
9              ρ_next = correlation(d, d', w_next);
10             if |ρ_prev − ρ_next| > ρ_min then
11                 //correlation interval is detected
12                 if r(ρ_prev < ρ_min) < r_min then
13                     refine (d, d', w_prev, granularity - 1);
14                     //exclude all correlated children groups
15                     for (d_1 ∈ d, d_2 ∈ d', ρ > ρ_min) do
16                     exclude (d_1, d_2, w_prev, granularity);
17                 end
18                 w_prev = ∅;
19             end
20             //prune for the next granularity using Lemma 1
21             w_BW = p_{i−2..i};   ρ_BW(d, d', w_BW) = Lemma1(n);
22             if r(ρ_BW > ρ_min) < r_min then
23                 exclude (d, d', w_BW, granularity - 1);
24             end
25         end
26     end
27 end
```

Algorithm 2: Group generalization algorithm

Input : Correlation ρ, time interval p, demographic groups d, d', maximality threshold θ
Output: Maximal demographic groups

```
1 //start from initial demographic groups complying to criteria
2 while d ⊅ d' & correlation(d, d', p) ≥ ρ do
3     d = arg max {correlation(d, parent(d), p) > θ};
4     d' = arg max {correlation(d', parent(d'), p) > θ};
5 end
6 return the last (d, d') complying to criteria;
```

4.3.1 Pruning Correlations

To find a pair of demographic groups with correlated sentiment, we have to evaluate all pairs of nodes in demographics lattice. However, we observe that correlation holds certain regularity properties on a demographics lattice and on time granularities, which are useful for pruning. We can apply pruning based on correlation estimates from the higher-granularity data (*vertical pruning*), and based on the observed part of the time series (*horizontal pruning*), as described below.

Vertical Pruning:

Given the DTree, we would like to be able to estimate correlations for a smaller time granularity based on the averages computed for a higher time granularity. This is possible using the Spruill and Gastwirth correlation estimation method [13], which relies on the Bartlett and Wald regression estimator.

LEMMA 1. *The estimate ρ_{BW} of correlation and its asymptotic standard deviation are computed using the following formula:*

$$\rho_{BW}(\boldsymbol{d},\boldsymbol{d}',p) = \frac{s'_{U3}-s'_{L3}}{s_{U3}-s_{L3}} \cdot \frac{\sigma(\boldsymbol{d},p)}{\sigma(\boldsymbol{d}',p)}, \quad \sigma(\rho_{BW}) = c\frac{1-\rho^2}{\sqrt{N}}$$

In the above equations, s_{U3} (s_{L3}) and s'_{U3} (s'_{L3}) are the averages of intermediate aggregates $s(\boldsymbol{d},p_i)$ and $s(\boldsymbol{d}',p_i)$ computed for $i \geq 2n/3$ ($i \leq n/3$), where n is the number of intermediate aggregates. The factor c is linearly depending on ρ and n and is estimated using the tabulation data given in [13]. We note that all standard deviations and intermediate aggregates used in this formula are directly accessible in the DTree at every granularity level.

Horizontal pruning:

If both the correlation threshold ρ_{min} and the time interval p of size n are known, then for every subinterval $p_1 \dots p_k$, $k < n$, with the corresponding inner product $(s \circ s')_1^k$, we can compute the upper bound of the correlation coefficient over p. We can then use this estimate to prune small correlations as more and more points of p are observed.

LEMMA 2. *If δ_s and $\delta_{s'}$ are the maximum sentiment deviations and the inner product of sentiment deviations $(s \circ s')_1^k$ at point k is less than $(n\rho_{min}\sigma_s\sigma_{s'} - (n\text{-}k)\delta_s\delta_{s'})$, then $\rho(s,s') < \rho_{min}$.*

PROOF.

$$\rho(s,s') = \frac{(s \circ s')_1^k + (s \circ s')_{k+1}^n}{n\sigma_s\sigma_{s'}} < \frac{(s \circ s')_1^k + (n-k)\delta_s\delta_{s'}}{n\sigma_s\sigma_{s'}} < \frac{n\rho_{min}\sigma_s\sigma_{s'}}{n\sigma_s\sigma_{s'}} = \rho_{min}$$

We note that estimating maximum deviations is only possible for bounded time series. This is true in our case, where sentiments are distributed between [-1,1]. Thus, we can set $\delta_s = |\overline{s}| + 1$, which occurs when a sentiment is the most distant from the mean value (deviating in the opposite direction).

As in the case of vertical pruning, the standard deviations and mean values of time series, used in the above estimation, are stored at a higher granularity level in the DTree and thus directly available.

4.3.2 Compressing Correlations

Although the pruning techniques help to efficiently compute top-k correlations, the number of these correlations can sometimes grow very large. There exists a tradeoff between the precision and recall of storing top-k correlations, which is defined by size k. Improving both characteristics is only possible for larger top-k sizes. In contrast, performance and scalability requirements demand top-k size to be small. This problem can be addressed by compressing top-k correlations, as described below.

We propose two algorithms of top-k compression: a greedy algorithm of triangulation correlation compression, TCC, and clustering correlation compression, CCC, based on density clustering. Nevertheless, other existing methods of clustering and graph compression can be adapted to compress top-k correlations.

Triangulation correlation compression TCC

Given correlation coefficients between two demographics nodes and a third one, we can estimate upper and lower limits for the correlation between them. Based on the correspondence between correlation coefficients and angles of vectors, representing local deviations of time series to their mean, we can apply the triangular inequality, which gives us the following lemma:

LEMMA 3. *If $\rho(\boldsymbol{d},\boldsymbol{d}') = \rho_1$, $\rho(\boldsymbol{d},\boldsymbol{d}'') = \rho_2$ and $\rho(\boldsymbol{d}',\boldsymbol{d}'') = \rho$ then*
$\rho_1\rho_2 - \sqrt{(1-\rho_1^2)(1-\rho_2^2)} \leq \rho \leq \rho_1\rho_2 + \sqrt{(1-\rho_1^2)(1-\rho_2^2)}$.

The detailed proof can be found in [3]. From the above inequality it follows that the transitivity of a positive and a negative correlation holds only if $\rho_1^2 + \rho_2^2 > 1$. This property requires absolute correlation values between two time series to be above 0.7 in order for the inequality to have any valuable prediction power. We note that this property is naturally achievable between nodes in a demographics lattice thanks to regularity and monotonicity of aggregated data. Therefore, Lemma 3 suits to our needs to compactly store correlations and recover missing values.

The simple greedy compression algorithm is listed in Algorithm 3. It removes elements from the top-k, which can be approximated using the triangulation principle. The compression process starts with a sorted list of correlations, which size is larger than k. Correlations are removed from the list one by one, being replaced with the next candidate in the list $(k+1)$ until the removal of any correlation introduces an error, larger than the one gained by adding a candidate. TCC algorithm can be further optimized by removing several correlations at once, until their approximations do not depend on each other. Such an optimization leads to a considerable performance benefit, since the approximation errors are not recomputed at every modification of a top-k list. However, the algorithm may become less optimal in this case. For the lack of space, we evaluate only the basic version of TCC, leaving possible extensions of this method for a future work.

Algorithm 3: Triangulation correlation compression TCC. Removes correlations which can be approximated using Lemma 3.

```
   Input  : demographics lattice L, number k
   Output: Top-k correlations
1  for (d,d') ∈ L × L do
2  |  add ρ(d,d') to topk;
3  end
4  sort topk descending by ρ;
5  while find (d,d',d'') ∈ topk s.t.
6  |     err = min|ρ(d,d') - Lemma3(d,d',d'')| do
7  |  if err < |topk[k+1]| then
8  |  |  remove ρ(d,d') from topk;
9  |  |  add topk[k+1] to topk;
10 |  |  keep ρ(d,d'') and ρ(d',d'') in the topk;
11 |  else trim topk to the size k; break ;
12 end
```

Clustering correlation compression CCC

Correlation coefficient between time series of sentiment can be transformed to Euclidean Distance [21]. Relying on this distance metric, we can identify groups of time series, which are highly-correlated on the same fixed time interval. We propose to apply unsupervised clustering to find such groups and to compactly store only their average and pairwise correlations. Since the space needed to allocate pairwise correlations is quadratic on the number of lattice nodes, replacing the correlations of individual nodes with those of their clusters can yield a significant compression ratio.

Because of the transitivity property of high correlations (according to Lemma 3), any set of highly correlated nodes is going to be densely packed in the Euclidean space, with a good cluster separation. Therefore, we find density-based algorithms of clustering more suitable, as their complexity in our case becomes asymptotically proportional to the number of elements in a cluster. Our clustering method uses the density-based algorithm DBSCAN [4], although any other distance-based algorithm can be used as well.

The compression process, described in Algorithm 4, starts with grouping lattice nodes into clusters based on their pairwise correlations. Clustering is performed using the absolute correlation

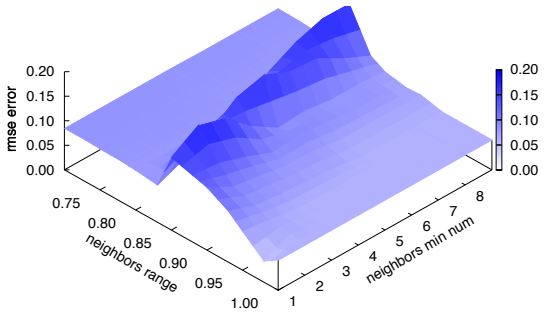

Figure 4: DBSCAN parameters space and optimization.

values, and the sign of each node's correlation with respect to its cluster is stored and later recovered. Unlike in euclidean spaces, where a cluster has a mean value or a centroid, in the correlation space there are only distances between nodes available. Therefore, we replace individual correlations between nodes with average correlations between their clusters: for different clusters the average is computed from pairwise correlations between their nodes, and for nodes in the same cluster the average is computed across all pairwise correlations within that cluster. Finally, correlations between outlier nodes and clusters or between outlier nodes are added to the output list which is trimmed to fit the top-k size.

Algorithm 4: Clustering correlation compression CCC. Replaces correlations with cluster averages.

Input : demographics lattice L, number k
Output: Top-k correlations

$clusters = \texttt{DBSCAN}(L \times L)$;
for $i = 0; i < |clusters|; i{+}{+}$ **do**
  for $j = i; j < |clusters|; j{+}{+}$ **do**
    $\rho(i,j) = 0$; //add cluster-cluster correlations
    for $(\boldsymbol{d},\boldsymbol{d}') \in clusters[i] \times clusters[j]$ **do**
      $\rho(i,j)$ += $\rho(\boldsymbol{d},\boldsymbol{d}')$;
    end
    $\rho(i,j) = \rho(i,j)/|clusters[i] \times clusters[j]|$;
    add $\rho(i,j)$ to $topk$;
  end
  for $\boldsymbol{d} \notin clusters$ **do**
    $\rho(i,\boldsymbol{d}) = 0$; //add outlier-cluster correlations
    for $\boldsymbol{d}' \in clusters[i]$ **do**
      $\rho(i,\boldsymbol{d})$ += $\rho(\boldsymbol{d},\boldsymbol{d}')$;
    end
    $\rho(i,\boldsymbol{d}) = \rho(i,\boldsymbol{d})/|clusters[i]|$;
    add $\rho(i,\boldsymbol{d})$ to $topk$;
  end
end
for $(\boldsymbol{d},\boldsymbol{d}') \in L \times L, \boldsymbol{d},\boldsymbol{d}' \notin clusters$ **do**
  add $\rho(\boldsymbol{d},\boldsymbol{d}')$ to $topk$; //add outlier-outlier correlations
end
while $|topk| > k$ **do**
  remove the lowest;
end

To achieve a good clustering, the density parameter should be set to a correct value, which is not known a-priori. As a minimum density parameter, DBSCAN uses a combination of neighbors range (which we substitute for minimal correlation) and their minimum number. We note that a lower minimum correlation corresponds to a broader neighbors range, unlike in original DBSCAN implementation. Figure 4 demonstrates an example of our parameters space and their corresponding compression errors for a fixed top-k size. We observe that for a broader range (Figure 4, left valley), DBSCAN tends to aggregate all nodes into one cluster. The error in this case remains constant, since we approximate all correlations with a single value. When we increase the density parameter, the clustering error tends to grow due to outliers and since there are still not many clusters. Finally, for the optimum parameters, all the highly correlated nodes are clustered together and all the smaller correlations are represented by cluster distances, significantly reducing the compression error (Figure 4, right valley). Since it is not known which of the valleys contains the global optimum, we propose to broadly scan the space of possible DBSCAN parameters and then refine the optimum value using the gradient descent method. Nevertheless, DBSCAN is a one-pass method that relies on a precomputed distances index, and multiple clusterings used for optimization do not result in a significant performance degradation.

We note that top-k list can hold correlations not only between nodes, but also between clusters and between nodes and clusters. Depending on their presence in the top-k list and attribution to clusters, we retrieve correlation values in the way, described in Algorithm 5. It is also possible to apply a triangulation compression for clustering distances, creating a hybrid method that takes advantage of both TCC and CCC.

Algorithm 5: Top-k correlation retrieving method

Input: demographics pair $(\boldsymbol{d},\boldsymbol{d}') \in L \times L$
//Determine a cluster id for each node (if clustered).
if *clustered* **then** $\boldsymbol{d} = cluster(\boldsymbol{d})$; $\boldsymbol{d}' = cluster(\boldsymbol{d}')$;
//If the value for a pair of ids is present, return it.
if $topk(\boldsymbol{d},\boldsymbol{d}') \neq null$ **then return** $topk(\boldsymbol{d},\boldsymbol{d}')$;
//If the value is not present, estimate using Lemma 3.
$\rho_{low} = -1$; $\rho_{high} = +1$;
for *all* $(\boldsymbol{d},\boldsymbol{d}',\boldsymbol{d}'')$ **do**
  $(\rho'_{low},\rho'_{high}) = \texttt{Lemma3}(\boldsymbol{d},\boldsymbol{d}',\boldsymbol{d}'')$;
  if $\rho_{low} < \rho'_{low}$ **then** $\rho_{low} = \rho'_{low}$;
  if $\rho_{high} > \rho'_{high}$ **then** $\rho_{high} = \rho'_{high}$;
end
return $(\rho_{low} + \rho_{high})/2$;

5. EXPERIMENTS

We ran experiments using both synthetic and real data. We first experiment with the synthetic data to evaluate the efficiency and performance of our algorithms, following it with the qualitative evaluation of correlations detected on a real dataset.

We implemented our algorithms in Java, and ran the experiments using Java JRE 1.7.0 on a machine with dual core 2.53 GHz CPU and 1.5 Gb of main memory.

5.1 Datasets

Synthetic Dataset

In order to accurately measure the precision of our system in identifying sentiment correlations, we conduct a series of experiments on synthetic data, containing time series of sentiments with artificially added positive and negative correlations, level biases and sentiment noise as demonstrated in Figure 5. We describe the layout of our dataset below.

Hierarchies. As a preliminary step, we generated a set of demographics hierarchies, for such attributes as age, gender, occupation, and location, containing 8, 3, 4 and 65 nodes respectively. Each node in every hierarchy was randomly assigned with a weight and a bias probability. Weights of nodes are distributed according to a *Zipf's* distribution, and normalized to add up to 1 at every level of a

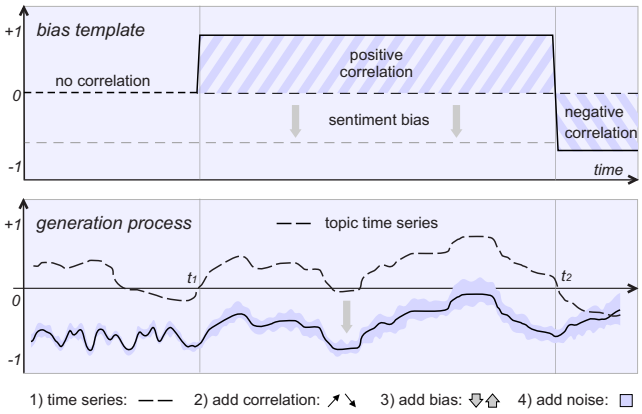

Figure 5: Generating biased sentiment time series.

hierarchy. Nodes from different hierarchies, when combined, form a lattice of 6240 demographic groups as shown in Figure 1. The sentiment volume of each demographic group was taken as a multiplication of weights of attribute nodes, and its bias as a weighted sum of their individual biases. That way, we achieve natural regularity in the demographics lattice, providing a natural distribution of sentiments, which is necessary for a proper evaluation of extracting maximal groups and pruning.

Topic Sentiment. The dataset itself contains time series of sentiments generated independently for multiple topics over a time span of 8 years. Each topic is represented by a unique *topic time series*, produced using a *random walk* method, which aggregates *uniformly* distributed sentiments, whose timestamps follow *Poisson* distribution. We vary the parameter of rate for timestamps to produce faster or slower changing time series for each topic. Since sentiments for the topic time series are sampled uniformly, its mean value is close to zero in a long run, meaning that it crosses the zero line a few times (for example, at points t_1 and t_2 in Figure 5). The original topic time series is stored for a randomly chosen demographic node, and we generate time series for other lattice nodes using individual *bias templates* for each of them. Bias templates contain sentiment bias levels and intervals of correlation with the topic time series (positive, negative or zero, randomly changed at "zero sentiment" points). The goal of our evaluation is then to correctly extract these correlation intervals between topic and biased time series, generated as described below.

Biased Sentiment. We produce a biased time series in a correspondence with the template, by copying (inverting) the topic time series in the case of positive (negative) correlation, or outputting randomized data otherwise (as seen in Figure 5, bottom). Following that, we add a certain positive or negative bias to the whole time series (shifting all the values) and the uniformly-distributed noise (for each value). Finally, we scale the time series to make sure that sentiments lay within the boundaries of [-1,1]. After generating a biased time series for the node, we insert raw sentiment data into the index in a proportion corresponding to node's volume. Then, we proportionally distribute these sentiments for all node's children (Figure 1), ensuring the maximality of that node.

MovieLens Dataset

MovieLens dataset[1] consists of 1 million ratings left by 6 thousand users on 4 thousand movies. It also comes with rich demographics attributes: age, gender, occupation, and location, which we directly imported to our application. These attributes result in a lattice of over 30 thousand nodes, making almost *half a billion* possible pairwise combinations. We extracted the geographical location from postal codes, however the number of ratings for many nodes in this hierarchy was exceedingly small. Since we aim at extracting only significant results, we disabled the use of the location attribute in this experiment. We used five-star MovieLens ratings as sentiments, by mapping them to [-1,1] continuous sentiment scale, where one star corresponds to a highly negative (-1) sentiment, and five stars correspond to a highly positive (+1) sentiment, and other ratings are distributed evenly.

Since comments for movies usually appear during a period of their showtime and then fade out, we propose using genres as topics, thus providing a stream of sentiments with rather constant rate, where new movies serve a role of events, leading to sentiment changes. The dataset has 18 genres, and most of movies belong to several genres at once, with their ratings contributing equally to all of them. This results in a certain regularity of sentiments across topics and demographic groups and challenges the detection of interesting correlations.

5.2 Evaluation Methodology

Efficiency evaluation is conducted on a synthetic dataset constructed as described in Section 5.1. It contains 10 topics with 400 biased time series for each of the topics, excluding children copies, while the much larger fraction of time series are generated randomly. We vary the level of noise added to time series in this dataset from 0.0 to 0.4 in absolute values, resulting in the same signal-to-noise ratios as sentiments are distributed on [-1,1]. We apply the scaling of time series after the noise was added.

We measure the average accuracy, precision and recall over all topics and all bias templates by measuring the correctness of correlation values over the extracted time intervals. For each of the time series, the extracted correlations are mapped to the binary scale $[-1, 0, 1]$ according to a 0.5 threshold, and compared to binary values stored in the corresponding template.

We report additional measurements, such as precision and recall, to break down the observed performance for a more detailed analysis. Precision is computed as the percentage of the length of extracted high-correlation intervals, which are found in the template as such (this is relevant for either +1 or -1 correlations). Recall is computed as the percentage of the length of high correlation intervals from the template, which were extracted as high correlations. Accuracy is computed as the precision of extracting all kinds of intervals from the template, including zero-correlation intervals. Finally, the root mean squared error (RMSE) is computed by measuring the actual differences between extracted correlation values and those stored in templates. It is computed as a square root of the average of squared errors, where the average is computed by weighting errors according to their time interval lengths.

5.3 Efficiency Evaluation

We conduct the evaluation of efficiency to demonstrate the properties of the proposed correlation extraction methods, and their usefulness when applied on noisy data. We evaluate our methods against noise for time granularities of 1 day and 10 days to demonstrate the effects that sentiment aggregation and time interval coarseness have on the accuracy of correlation. The size of fixed and sliding windows in both cases was equal to 10 samples. The observed behavior is not specific to our implementation alone. Rather, it marks the best possible performance for computing correlations using various fixed or sliding interval methods at particular levels of aggregation and noise.

[1] http://www.grouplens.org/node/73

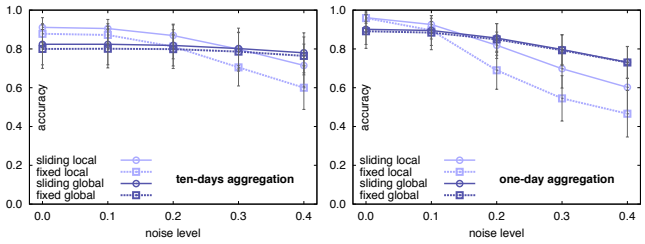

Figure 6: Accuracy of baseline correlations vs aggregation.

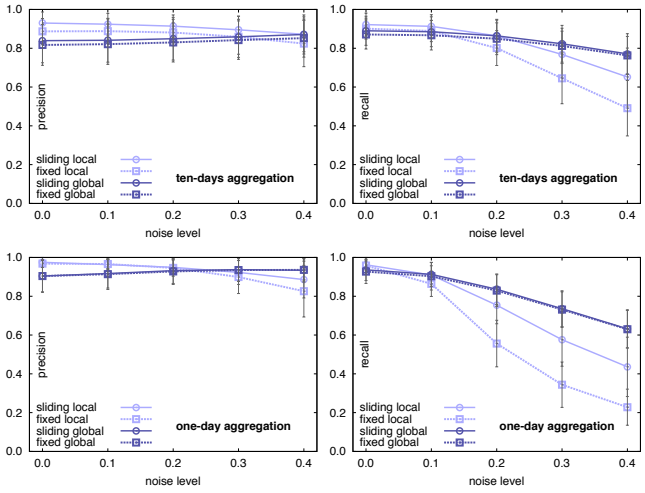

Figure 7: Precision and recall of baseline correlations.

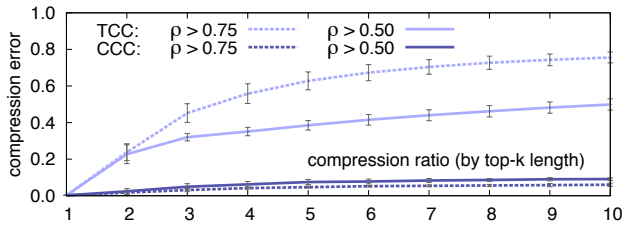

Figure 8: Compression error for top-k correlations.

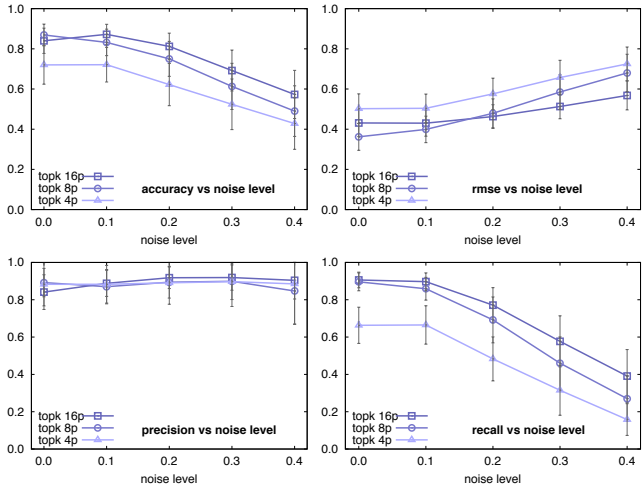

Figure 9: Accuracy of top-k correlations, 10 days aggregation.

5.3.1 *Baseline Correlation Methods*

The results achieved by baseline correlation methods are depicted in Figures 6 and 7, respectively. We observe that the best accuracy[2] is achieved in the case of local average methods, albeit, only at a proper aggregation granularity (in our case, ten days). Using same methods at granularities that are affected by noise results in a substantial performance drop, as can be seen in Figure 6, right. Very high levels of noise, usually present at low granularities, can lower or even reverse the actual correlation between the time series, affecting precision and recall. On the other hand, increased granularity may reduce precision because of the discretization errors when identifying correlation interval boundaries. This can be observed by comparing zero-noise accuracy values for one- and ten-days aggregations: the latter shows slightly lower accuracy solely by the coarseness of time intervals. Using global average (computed for the whole period and same for all windows) has proven to be more noise-resistant although it is not always as precise as local average, and cannot be applied in ad-hoc scenario, requiring pre-specified global time interval. From the discussion above it is evident that correlations must be analyzed using sliding or fixed windows and at varying aggregation granularities.

5.3.2 *Top-K Correlation Method*

We evaluate the individual performance of *TCC* and *CCC*, by measuring their average compression error and its variance while varying top-k sizes from 1 to 10 times of their initial length. To construct the top-k list, we computed correlations over disjoint pairs of demographic nodes for fixed time intervals (with local averages), taken from 17 different topics in MovieLens. Then, we filtered correlations according to the significance and minimum correlation criteria, obtaining the lists of (approximately) 12K high ($\rho > 0.50$) and 2K very high ($\rho > 0.75$) top-k correlations for an initial set of 140K disjoint pairs.

Compression error is computed as the root mean squared error (RMSE) between actual correlations and those retrieved according to Algorithm 5. We note that an optimal compression (with the smallest error) for CCC clustering method was sometimes achieved with a size, smaller than that required by the compression ratio parameter. In such cases the remaining space was filled with the highest non-clustered correlations.

In Figure 8 we present the results of our evaluation. We observe that TCC triangulation compression shows better performance when it is able to fit all the high correlations necessary for describing the rest of correlations, what happens in the case of large initial top-k list ($\rho > 0.50$). In the case when all correlations are high ($\rho > 0.75$) and there is a high compression ratio, there is a large portion of correlations which do not fit into the compressed top-k list and neither can be triangulated from the correlations present in the list. The error in this case is the highest. On the other hand, CCC clustering compression benefits from compressing higher correlations as soon as there is enough space in top-k to store an optimal number of clusters. In this case most of the high correlations appear within clusters and the amount of correlations which are not approximated by cluster-cluster distances becomes relatively small. Nevertheless, CCC can become inefficient due to the clustering information overhead if there are many distanced small clusters of 3 items. Since TCC method is able to approximate the third correlation using the remaining two, it can be a good companion in such cases. We recommend to use hybrid CCC+TCC method for compressing correlations as the most universally applicable, especially in the case of moderate compression ratios.

[2]The best achieved accuracy is not 100%, because some correlation intervals are smaller than the minimal correlation window.

We now look at the efficiency of correlation extraction of our top-k method using the same synthetic dataset, we used to evaluate the baseline methods. We calculated lists of high correlations ($\rho_{min} = 0.5$) for each of the fixed time intervals, containing 20-50K top-k values out of 15M (disjoint) and 40M (total) group pairs, and compressed them using the hybrid CCC+TCC method with clustering parameters optimized individually for the specific top-k size. We varied top-k sizes from 4 to 16 four-kilobyte disk pages (each page can hold up to 800 correlations or cluster distances). Figure 9 demonstrates that with the sufficient list of top-k correlations, computed for fixed-windows, it is possible to match the accuracy of conventional methods. A more detailed inspection shows that the drop in accuracy for smaller top-k sizes is caused mainly by a decreasing recall, due to the inability of top-k to fit all the high correlations, which our synthetic dataset is mainly composed of.

5.4 Scalability Evaluation

Performance of Indexing Sentiments

When new sentiments are being inserted into the DTree, they go through a number of levels in the tree index. First, the updates are aggregated in buckets, corresponding to the lowest time granularity. Second, the aggregated values are inserted in a single update, populating nodes of the index from top to bottom. For the DTree, updates are accumulated and inserted for each demographic group individually. Nevertheless, this method still improves the performance since disk pages are being accessed only once for each batch. During every update and at every granularity level, sentiment values are inserted into corresponding demographics lattices, for the specified group and *all of its parents*.

Demographic lattices can be stored on disk using different structures. The simplest of such structures is a *fixed array*, suitable for constant demographics lattices. It allows fast indexed access to the lattice values. The other is a *binary tree* of variable size and structure, which has reasonably fast $\log|L|$ access time and allows demographics lattices to be extended. However, this structure requires more space for storage and extra processing time.

We evaluate update performance for both structures by measuring index throughput versus main memory cache size. Smaller cache sizes require disk pages to be flushed more often, while larger cache sizes allow to have a smaller number of sequential updates. With the cache size of one page, the system becomes persistent, i.e. all changes are immediately stored on disk. Larger memory cache sizes demonstrate higher throughput, which is ultimately bounded by disk writing speed, as the processing time is sufficiently smaller than the input-output time. Another advantage of a larger cache is that in-memory updates are extremely fast if a cache has enough space. Nevertheless, the update rate drops back to nominal once the cache is full, and until it is flushed to disk, there is a risk of loosing the data in the case of memory or system fault. The results of our evaluation are presented in Figure 10. We observed that disk writes occur every time the DTree is updated up until when the cache size reaches the maximum number of parent nodes in the tree. In our experiment, the updating rate stayed constant until 4 pages. After that, when the cache is further increased, we see a linear improvement in performance, which is then asymptotically reaches the maximum value, bounded by the disk write speed. Binary tree (dynamic) storage features smaller update rate compared to the fixed array storage, since its disk pages occupy twice more space and since a binary tree is dynamically constructed.

Performance of Extracting Correlations

In Figure 11 we compare the time needed to extract correlation intervals using the same setup as in our accuracy evaluation.

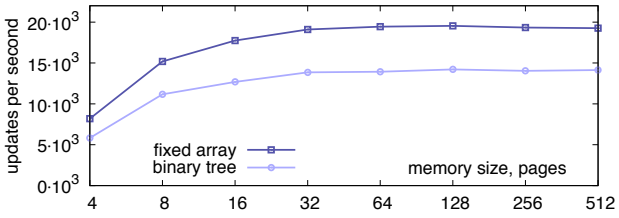

Figure 10: Performance of DTree versus memory cache size.

In the top chart, we report average times for the proposed methods using sliding and fixed time intervals (left), and the top-k technique (right). The time needed to compute correlations using sliding time intervals is approximately one third larger than the time taken by a fixed-interval method, since the prior needs to incrementally compute and compare correlations for two intervals: one is a fixed-length interval, sliding in front of the cursor, and another one is a dynamically expanding interval behind the cursor. The time of baseline methods remains fairly large since they compute correlations for a set proportional to $|L \times L|$.

In the bottom chart, we demonstrate the effect of hierarchical pruning for fixed-interval correlations and compare it to the grid-hashing correlation pruning used by StatStream [21]. We note that StatStream cannot be applied for sliding-interval correlations when sliding intervals among time series are of different lengths, as in our method. Moreover, the kind of time series approximation used in StatStream can not be used for bounded sentiments, where it results in numerous false positives at shorter interval lengths.

Both methods lead to significantly improved execution times in comparison to baseline methods, with hierarchical exhibiting better relative performance. The advantage of our method on highly correlated data is more pronounced with lower correlation thresholds, when maximality constraints allow earlier pruning. On the other hand, StatStream's performance benefits from better selectivity of higher thresholds and when time series are sparsely correlated, making it a good complementary approach.

Overall, we observe that the best performance is achieved by top-k correlations, which we report in both charts (with and without hierarchical pruning) for varying top-k sizes (16, 8 and 4 disk pages). It is evident, that top-k method is much faster even in the case of larger top-k sizes, where it matches the accuracy of direct methods. Furthermore, the top-k method demonstrates sub-linear scalability, sustaining almost the same performance even with exponentially increasing top-k sizes.

5.5 Usefulness Evaluation

To demonstrate the usefulness of our approach, we automatically identified the highest (i.e., exceeding 0.9) maximal significant correlations between disjoint demographic groups in the MovieLens dataset. We note that a naive solution to this problem will require computing and comparing almost half a billion of time series. In Figures 12-13 we represent positive and negative correlations organized using a graph-like structure. Correlations are visualized as edges between demographic groups, labeled according to topics (genres). For brevity, we visualize only a small fraction (up to 10) of high correlations identified for each topic. We note that due to this filtering, some of the correlation edges are not present in the graph. This does not necessarily mean that such correlations are below the specified threshold. Finally, we do not report correlations between some highly-overlapping (but still disjoint) demographic groups, such as between {*Under 18*} and {*K-12 student*} or {*56+*} and {*retired*}.

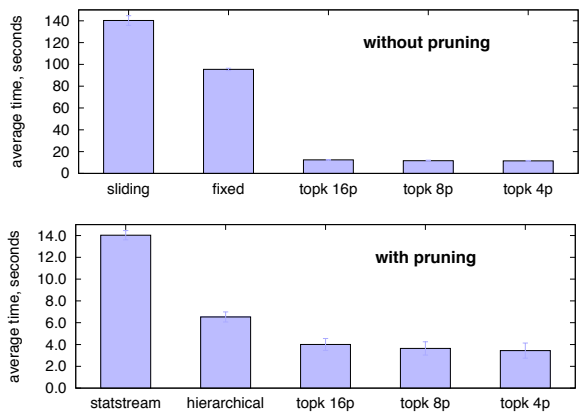

Figure 11: Comparison of correlation methods performance

We highlight nodes with the most unusual and interesting correlations, which took our particular attention, and provide their additional details in Table 1. For instance, in the top right corner of Figure 12 we observe a cluster of correlations on topic *animation* between the three very different groups: {*F, 56+*}, {*M,customer service*} and {*M,grad student*}. A more careful examination of this figure reveals that {*F, retired*} think of *thrillers* as {*M, 18-24, programmer*} - a short of a surprise, knowing that they have the same attitude to *romance* as {*M, K-12 student*}. Finally, {*academic*} and {*writer*} people have shown the most vibrant and unusual behavior in MovieLens dataset, producing many of the anti-correlations we observed in Figure 13.

Such results can be used to drive the work in a number of directions: in sociology, researchers may investigate why these unexpected correlations exist, and examine more carefully and in greater detail the interests of users; in marketing, knowledge of group sentiments, how they change over time and how they are related to other groups could help expand existing markets, by influencing similar target groups, gaining a better understanding of the fan base, and monitoring the reaction of opposing groups; in collaborative filtering, new systems may expand their range of recommendations on particular topics with results from correlated groups, resulting in an enhanced user experience.

6. RELATED WORK

The problem of identifying sentiment behavior in demographic groups has been traditionally addressed by polling. Polling requires long-term monitoring of a large sample of the population in order to allow for a meaningful comparison of sentiments among demographic groups. However, this process is rather expensive and error-prone [8, 6]. Therefore, many scientists look towards evaluating online sentiments, especially considering their existing correlation with actual opinions.

Online sentiments monitoring has been approached by scientists using a variety of data mining algorithms, from trend monitoring [18] to contradiction detection [11, 16], although these studies were not specifically accounting for relationships between demographic groups, their sentiment's correlation and hierarchical organization.

Recently, the work of Das et al. [2] introduced complex mining of sentiment data in the form of ratings, where the authors aimed at extracting meaningful demographic patterns. Our work differs in that we study the complementary problem of extracting groups with correlated sentiments over time, that is, groups that react similarly over time to external events. Then, this kind of analysis can also help provide a more meaningful interpretation for the biases observed in the sentiments expressed by the different groups.

Zhang et al. [20] introduced a sentiment aggregation and visualization system, which interactively displays sentiments based on the selected geographical location. The system represents a world map featuring a time evolution of sentiments expressed in news articles coming from different geographical regions. It automatically retrieves and displays sentiments around some particular time period for *ad-hoc* queries, aggregating them over different locations as the user navigates the map, or zooms in and out. However, it only targets small-scale data aggregation using a single demographics hierarchy. A work of Mandel et al. [6] is a step up in this direction, featuring sentiment aggregation over time for demographic groups formed by gender and location attributes. The authors analyzed Twitter messages for hurricane *Irene* and revealed sentiment differences among demographic groups. Their study suggested a necessity to account for classification errors (sentiment noise) and sentiment biases, thus forestalling our analysis, which addresses both of these problems.

Problems related to the identification of correlations among multiple time series have been studied by the data streams community, using a variety of techniques. These techniques focused on the efficient computation [21, 7], hidden variables [9], local correlations [10], pruning of candidate pairs [1], and lagged correlations [12]. Among them, StatStream [21] is the one that is closest to our work. StatStream computes correlations using sliding time intervals of specified sizes, composed of a number of sub-intervals of fixed length. It employs the *Discrete Fourier Transformation (DFT)* to compute correlations in an approximate and incremental manner. Our solution is different from the above works in a number of ways: (a) it analyzes time series using multiple aggregation granularities and detects correlations on ad-hoc time intervals; (b) it applies effective top down pruning both on time and demographics hierarchies; and (c) it uses correlation compression techniques to achieve efficiency and scalability.

7. CONCLUSIONS AND FUTURE WORK

In this work we approach the novel problems of characterizing sentiment evolution in a demographic group and identifying correlated groups, which address the large-scale sentiment aggregation. We design efficient algorithms for sentiment aggregation based on a careful indexing of time and demographics into hierarchies and demonstrate that our problems can be solved effectively on a large scale using clever pruning, top-k and compression methods.

Our approach allows observing sentiment behavior at a much finer level of detail than currently possible, helping to identify cases that are counter-intuitive and can only be observed by processing large amounts of data. Moreover, it enables an unprecedented scale-up of traditional social studies and raises new data analysis opportunities, useful for sociology and marketing researchers.

We outline some interesting problems and extensions of this paper, which we plan to work on. We consider only a disjoint type of relation, although it is possible to expand the notion of relations between groups to any arbitrary path in a demographics lattice, and use it as a filtering argument to our problems. Also we are investigating the case where disjoint groups appear to be the same sets of users due to a strict dependency among attributes. Filtering high correlations between such groups is possible when their sets of users are known and can be done as a preprocessing step. Alternatively, we can compare the volume of sentiments between these groups, which becomes possible since our DTree storage preserves this information.

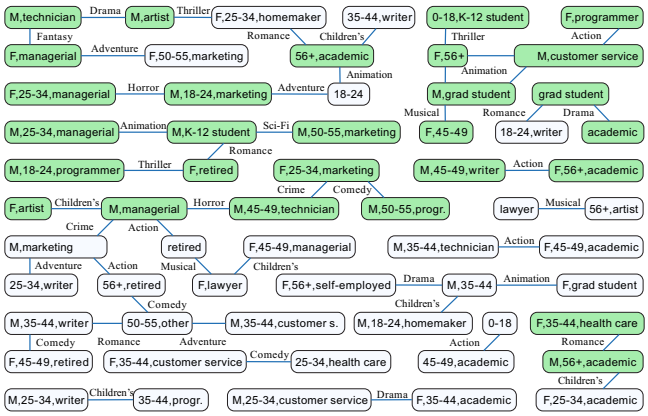

Figure 12: Positive sentiment correlations in MovieLens.

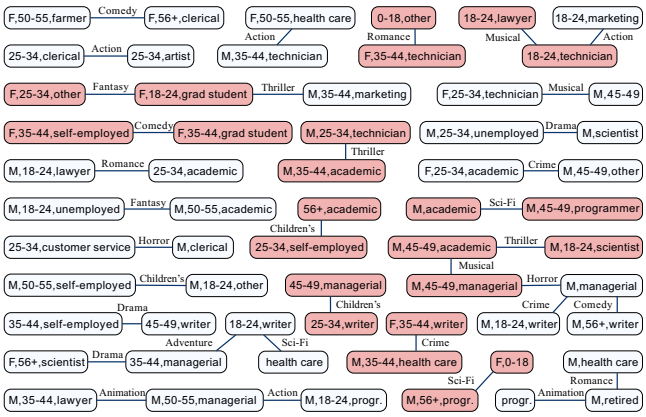

Figure 13: Negative sentiment correlations in MovieLens.

8. REFERENCES

[1] R. Cole, D. Shasha, and X. Zhao. Fast window correlations over uncooperative time series. In *KDD*, pages 743–749, 2005.

[2] M. Das, S. Amer-Yahia, G. Das, and C. Yu. Mri: Meaningful interpretations of collaborative ratings. In *VLDB*, 2011.

[3] N. S. Eric Langford and M. Owens. Is the property of being positively correlated transitive? *TAS*, 55(4):322–325, 2001.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[5] IBM. Social sentiment index. *IBM Smarter Analytics*, 2012.

[6] B. Mandel, A. Culotta, J. Boulahanis, D. Stark, B. Lewis, and J. Rodrigue. A demographic analysis of online sentiment during hurricane irene. In *LSM 2012*, pages 27–36, 2012.

[7] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time-series data. In *SIGMOD*, pages 171–182, 2010.

[8] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM*, 2010.

[9] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[10] S. Papadimitriou, J. Sun, and P. S. Yu. Local correlation tracking in time series. In *ICDM*, pages 456–465, 2006.

[11] A. M. Popescu and M. Pennacchiotti. Detecting controversial events from twitter. In *CIKM*, pages 1873–1876, 2010.

[12] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[13] N. L. Spruill and J. L. Gastwirth. On the estimation of the correlation coefficient from grouped data. *JASA*, 77(379):614–620, Sept. 1982.

[14] M. Thelwall, K. Buckley, and G. Paltoglou. Sentiment in twitter events. *JASIST*, 62(2):406–418, 2011.

[15] M. Tsytsarau and T. Palpanas. Survey on mining subjective data on the web. *DAMI*, 24(3):478–514, 2012.

[16] M. Tsytsarau, T. Palpanas, and K. Denecke. Scalable detection of sentiment-based contradictions. In *DiversiWeb, WWW 2011*, 2011.

[17] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *ICWSM*, 2010.

[18] I. Varlamis, V. Vassalos, and A. Palaios. Monitoring the evolution of interests in the blogosphere. In *ICDE Workshops*, 2008.

[19] F. Wang, N. Helian, and Y. J. Yip. Run-length encoding applied to grid space storing and indexing. In *ICWI*, pages 461–471, 2002.

[20] J. Zhang, Y. Kawai, T. Kumamoto, and K. Tanaka. A novel visualization method for distinction of web news sentiment. In *WISE*, 2009.

[21] Y. Zhu and D. Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.

Topic	Group 1	Group 2	Begin	End	Correlation	Significance H1	Significance H2
Animation	{M,college/grad student}	{F,56+}	30.07.00	17.11.00	0.96	8.68E-07	9.02E-02
Animation	{M,K-12 student}	{M,25-34,executive/managerial}	07.11.00	07.03.01	0.94	2.67E-05	2.13E-01
Children's	{M,executive/managerial}	{F,artist}	30.07.00	07.11.00	0.94	2.03E-05	2.41E-01
Children's	{25-34,writer}	{45-49,executive/managerial}	08.10.00	15.02.01	-0.95	1.64E-05	1.28E-01
Children's	{25-34,self-employed}	{56+,academic/educator}	21.04.00	29.08.00	-0.93	4.28E-05	2.78E-01
Comedy	{F,25-34,sales/marketing}	{M,50-55,programmer}	17.11.00	13.09.01	0.97	4.11E-08	6.36E-04
Comedy	{customer service}	{45-49,artist}	18.10.00	26.01.01	0.96	3.25E-06	1.05E-01
Comedy	{F,35-44,self-employed}	{F,35-44,college/grad student}	07.12.00	16.04.01	-0.95	1.33E-05	1.28E-01
Fantasy	{M,technician/engineer}	{F,executive/managerial}	18.10.00	05.02.01	0.92	6.50E-05	3.71E-01
Fantasy	{F,18-24,college/grad student}	{F,25-34,other}	08.09.00	17.11.02	-0.92	6.80E-05	1.53E-01
Romance	{F,35-44,doctor/health care}	{M,56+,academic/educator}	17.12.00	02.11.01	0.98	4.23E-04	4.40E-06
Romance	{F,retired}	{M,K-12 student}	07.11.00	15.02.01	0.96	3.27E-06	1.05E-01
Romance	{M,artist}	{56+,academic/educator}	08.09.00	27.12.00	0.96	5.40E-06	9.02E-02
Romance	{F,35-44,technician/engineer}	{Under 18,other}	27.11.02	07.03.03	-0.96	3.69E-06	1.05E-01
Drama	{college/grad student}	{academic/educator}	18.09.00	27.12.00	0.96	4.34E-06	1.05E-01
Crime	{F,35-44,writer}	{M,35-44,doctor/health care}	05.02.01	29.08.02	-0.92	4.17E-03	1.95E-01
Action	{M,45-49,writer}	{F,56+,academic/educator}	30.07.00	26.01.01	0.96	3.55E-06	3.33E-02
Action	{F,programmer}	{M,customer service}	07.12.00	17.03.01	0.96	4.80E-06	1.05E-01
Thriller	{F,retired}	{M,18-24,programmer}	28.10.00	25.02.01	0.96	8.00E-06	7.76E-02
Thriller	{Under 18,K-12 student}	{F,56+}	17.11.00	25.02.01	0.95	9.07E-06	1.71E-01
Thriller	{M,25-34,technician/engineer}	{M,35-44,academic/educator}	17.11.00	27.03.01	-0.95	1.02E-05	1.28E-01
Thriller	{M,18-24,scientist}	{M,45-49,executive/managerial}	27.11.00	05.07.01	-0.95	3.75E-06	5.85E-02
Horror	{M,45-49,technician/engineer}	{M,executive/managerial}	17.11.00	07.03.01	0.96	3.48E-06	9.02E-02
Horror	{M,18-24,sales/marketing}	{F,25-34,executive/managerial}	21.04.00	07.03.03	0.96	3.28E-04	8.59E-07
Sci-Fi	{M,50-55,sales/marketing}	{M,K-12 student}	05.02.01	21.04.02	0.96	5.79E-06	1.21E-03
Sci-Fi	{M,45-49,programmer}	{M,academic/educator}	21.04.00	07.03.03	-0.98	3.07E-04	2.24E-01
Sci-Fi	{M,56+,programmer}	{F,Under 18}	16.01.01	11.01.02	-0.97	1.84E-04	1.94E-02

Table 1: Positive and negative sentiment correlations identified in MovieLens dataset.