Semantic Similarity Between Search Engine Queries Using Temporal Correlation

Steve Chien
Microsoft Research, Silicon Valley Campus
Mountain View, CA
schien@microsoft.com

Nicole Immorlica*
Massachusetts Institute of Technology
Cambridge, MA
nickle@theory.csail.mit.edu

ABSTRACT

We investigate the idea of finding semantically related search engine queries based on their *temporal correlation*; in other words, we infer that two queries are related if their popularities behave similarly over time. To this end, we first define a new measure of the temporal correlation of two queries based on the correlation coefficient of their frequency functions. We then conduct extensive experiments using our measure on two massive query streams from the MSN search engine, revealing that this technique can discover a wide range of semantically similar queries. Finally, we develop a method of efficiently finding the highest correlated queries for a given input query using far less space and time than the naive approach, making real-time implementation possible.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Experimentation

Keywords

search engines, query stream analysis, semantic similarity among queries

1. INTRODUCTION

A continuing challenge facing modern search engines is that of determining and satisfying a user's needs based only on very short text queries. Such queries can be imprecise and often reflect a user's own ambiguity while performing a search. One frequently mentioned approach to addressing this problem is for search engines to help users refine their search by suggesting alternative queries in response to a user input. Unfortunately, this can be a difficult problem in itself, with many solutions relying on mining a large text corpus to uncover semantically similar terms.

In this paper we explore the possibility of discovering such semantically similar queries by instead mining the query stream received by a search engine. The central idea we use is to infer that two query strings are semantically related if they are *temporally correlated* (i.e., if their popularities over time behave similarly), thus reducing the need to understand query terms at a linguistic level.

We mention that like other techniques for finding semantically related queries, this approach has applications such as expanding user search and suggesting keywords to advertisers. However, we believe using temporal correlation for these purposes also has the following unique advantages:

- *Query context.* By using temporal correlation, our approach implicitly "understands" why a query is interesting at a particular time. For example, as the query `chocolate` becomes popular in February, our approach will suggest other Valentine-related queries, since those queries are also popular at the same time.

- *Rapid adjustment.* Similarly, our approach also has the ability to adjust quickly to news events, as queries related to an event will begin to appear immediately in a search engine query stream.

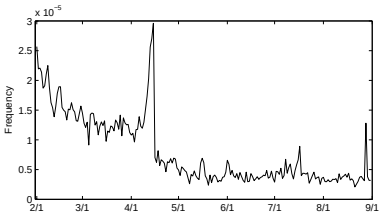
In order to automate the process of finding semantically similar queries using temporal correlation, we must first formalize a measure of temporal correlation, demonstrate that our measure effectively finds semantically related queries, and be able to compute our measure efficiently given the scale of a search engine query stream. Along these lines, we do the following:

Devise a formal measure of temporal correlation. We do this by first defining the *frequency* of a query q over a particular time unit i as the ratio of the number of occurences of q in i to the total number of queries in i. Our measure of the temporal correlation between two queries p and q over a span of many time units is then the standard *correlation coefficient* of the frequencies of p and q. This is a value between -1 and 1 with larger values indicating stronger correlation.

Evaluate the effectiveness of our measure. We evaluate our measure by testing it over two large query streams. The first query stream covers a seven month period in 2004 and includes the daily aggregate totals of each of 33,986,136 distinct queries (over 5 billion total) received during that period. The second data set covers an overlapping two month period in 2004 and, for each of 15,248,106 distinct queries (over 1 billion total) received in that period, lists the exact

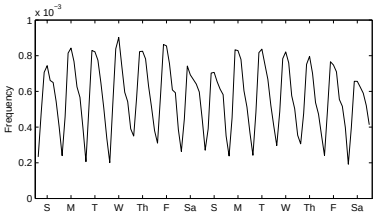
*Work completed while author was visiting Microsoft Research, Silicon Valley Campus.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.



income tax	www.irs.gov
tax returns	1040ez forms
h&rblock	efile
federal income tax	freetaxusa.com
taxact	

Figure 1: The frequency function of `income tax` and the queries with the highest correlation with `income tax`.



walmart	sam's club
target	best buy
petsmart	sears
circuit city	realtor.com
bed bath and beyond	

Figure 2: The frequency function of `walmart` and the queries with the highest correlation with `walmart`.

times that these queries were received [1]. We find that our measure of temporal correlation uncovers a wide range of semantically related queries, from the expected to the more surprising.

As might be expected, our measure allows us to identify semantically related queries that are "event driven"– those queries whose frequencies vary greatly near a specific event, such as a holiday or a news item. As one of many examples, we show in Figure 1 the frequency function of the query `income tax` over time as well as the other queries that best correlate with it. Intuitively, the pronounced change in frequency around an event (in this case April 15) makes finding semantic relationships possible.

More surprisingly, however, our technique can discover interesting relationships even among non-event driven queries whose frequencies do not change greatly over the long term. One example here is that of `walmart`, whose frequency function and highest correlated queries are shown in Figure 2.

While the frequency function of `walmart` may not appear unusual, showing only that it is more popular during the day than at night, it is in fact distinctive enough such that it correlates very well with other large retailers. We also find this to be true for queries in many other areas; for example, newspapers, airlines, and banks among others also tend to have high correlation among themselves.

Our approach does have several weaknesses that prevent it from being effective in isolation. For example, it cannot handle the large fraction of queries that appear only a small number of times, and generates false positives on others. Overall, however, we believe that this approach can be combined with complementary query refinement methods and text mining techniques to form a larger and more effective query refinement system.

Develop a method to efficiently implement our measure. Given a particular query q, we would like to find all other queries that have high temporal correlation with q efficiently, perhaps in real time when q is presented to a search engine. Unfortunately, the sheer size of these query streams makes this a challenge. The naive approach would require storing for each of n queries their frequencies for each of d time units, as well as making a linear pass through all of this data for each input query q, which could be prohibitively expensive in both space and time.

Borrowing from classical techniques in low-dimensional embeddings and nearest neighbor algorithms [4, 5, 6], we demonstrate a particularly simple and easily implemented approach to find approximate top-correlated queries in much less space and time. Our technique requires storing only 128 bits of data for each query, and a linear pass through these 128 bits for an expected $\frac{1}{776}$ fraction of the queries. This makes real-time processing of input queries feasible.

1.1 Related Work

The idea of using temporal similarity to find semantic similarity was previously and independently studied by Vlachos et al. [8]. They use a measure of temporal similarity based on the Euclidean distance between the demands over time of two queries, and describe an approach to find the most similar queries to a given query using the several best Fourier coefficients of each query's demand function. They also describe a method for detecting bursts in a query's demand function. Their emphasis is on their techniques and they report only limited experimental results on the semantic relationships they are able to find. While our measure is similar, our contributions include a much more extensive evaluation of this measure as well as an efficient method for computing our measure.

We also list some examples of other recent approaches to defining and extracting semantic similarity in queries. Daumé and Brill [3] suggested that queries are related if they share a large fraction of their search results. Earlier, Beeferman and Berger [1] suggested using clickthrough data to group queries, saying that two queries are similar if users selected the same document after searching for each of them. Wen et al. [9] and Cui et al. [2] also used clickthrough data to find correlations between terms in user queries and terms in the documents that are selected from those queries (as well as other techniques), while Kraft and Zien [7] suggested an approach based on crawled anchortext. The results produced by these techniques are mostly complementary to ours. However, our techniques only require a time-stamped query stream as input; no linguistic analysis or additional information regarding the search results is necessary.

1.2 Paper Organization

We first define and motivate our measure of temporal correlation in Section 2. We go on to evaluate its effectiveness in Section 3, and then describe how to efficiently implement it in Section 4. Finally, we discuss future work in Section 5.

[1]Both of these data sets exclude queries below a certain threshold of popularity; for details see Section 3.

2. SIMILARITY MEASURE

In this section we motivate and define our measure of temporal similarity.

We first introduce the notion of the frequency function of a query. Assume some discretization of time into time units. Let $X_{q,i}$ be the *frequency* of query q during the ith time unit, or $\frac{n_{q,i}}{N_i}$, where $n_{q,i}$ is the number of occurrences of query q during the ith time unit and N_i is the total number of queries during the ith time unit.

Definition: The *frequency function* of a query q over d time units is the d-dimensional vector $X_q = (X_{q,1}, \ldots, X_{q,d})$.

If we think of the frequency $X_{q,i}$ of a query in a particular time unit as a random variable, we can then define the similarity of two queries as the correlation coefficient of their frequency functions:

Definition: For a particular query q, let $X_{q,i}$ be its frequency function, $\mu(X_q)$ be its mean frequency, and $\sigma(X_q)$ be the standard deviation of its frequency. Then the similarity of two queries p and q is the correlation coefficient of their frequency functions, or

$$\frac{1}{d}\sum_i \left(\frac{X_{p,i}-\mu(X_p)}{\sigma(X_p)}\right)\left(\frac{X_{q,i}-\mu(X_q)}{\sigma(X_q)}\right).$$

The correlation of two random variables is a standard measure of how strongly two variables are linearly related. It always lies between -1 and 1, with 1 indicating an exact positive linear relationship between them and -1 indicating an exact negative linear relationship. The correlation of a variable with itself is 1, while the correlation of two independent random variables is 0.

Correlation therefore naturally captures our intuitive notion of temporal similarity. To illustrate this, in Figure 3, we plot the frequency functions of two pairs of queries and note that they show high correlation coefficients.

While this definition appears straightforward, it does contain some important features. First, note that the frequency function of a query is the *density* of that query in a given time unit as opposed to simply the *number of occurrences* of that query in a given time unit. This normalizes out the effects of the natural variation over time of query stream volume, and prevents some pairs of queries from showing artificially high correlation simply because the total number of queries per time unit is much larger during the day than overnight, and much larger on weekdays than on weekends. As an example, we compare in Figure 4 the highest correlated queries with `southwest airlines` our techniques (see Section 4) find when we use density and when we use the number of occurrences.

Another important point about our measure is that we use the *correlation* of the frequency functions as opposed to the *covariance*, another common measure of similarity. The covariance fails to normalize for the variance in frequency functions, and thus queries with high variance falsely appear to be temporally related to many other queries.

3. EXPERIMENTAL ANALYSIS

In this section we evaluate the effectiveness of our measure using query data from the U.S. market of the MSN search engine. We had access to two data sets from this query stream: The first data set runs from February 1, 2004 through August 31, 2004 and includes the *daily* aggregate totals of each of the 33,986,136 distinct queries that appeared at least 10 times on at least one day. The second, more detailed, data set runs from August 1, 2004 through September 25, 2004[2] and, for each of the 15,248,106 distinct queries that appeared at least 10 times in this period, gives the exact times that these queries were received. The results that we extract from these data sets demonstrate that our technique is able to detect a variety of often surprising semantic relationships among queries, though with some shortcomings. We first discuss the details of our experimental technique in Section 3.1, before presenting a summary of some of these results in Section 3.2.

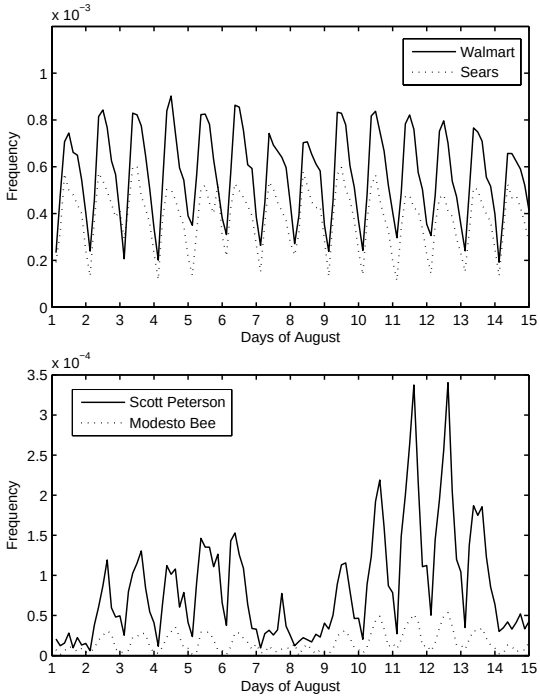


Figure 3: (top) The frequency functions of `sears` and `walmart` in August, correlation 0.92; and (bottom) the frequency functions of `scott peterson` (accused murderer) and `modesto bee` (his local newspaper), correlation 0.94.

`orbitz`	`sprintpcs.com`
`travelocity`	`att`
`united airlines`	`mortgage calculators`
`american airlines`	`holiday inn`
`frontier airlines`	`expedia`
`expedia`	`continental`
`www.fafsa.ed.gov`	`wells fargo`

Figure 4: Top correlated queries with `southwest airlines` using (L) frequency, and (R) number of occurrences.

[2]This data set is missing a two day period from the 17th through 18th of August due to a data collection error.

3.1 Experimental Parameters

Before we can run our experiments, we need to fix the parameters of our measure. The main parameters are:

- Definition of time unit. Our measure assumes a discretization of time, but how long should each unit be? one hour? one day? one week? Too fine a scale threatens to drop all correlations well below any reasonable threshold, but too coarse a scale creates meaningless correlations.

- Dimension d. Our measure runs for a certain length of time (or, equivalently, over a certain number of time units). What length of time should we study? one week? one month? one year?

- Definition of meaningful correlation. Once the pairwise correlations have been computed, what value of correlation suggests semantic similarity?

Our detailed dataset allows us to experiment with the length of the time unit. In Figures 21 through 26 (at the end of this paper), we consider two input queries – `disney` and `republican national convention` – for three, six, and twenty-four hour time units over a period of two months. For each length of time unit and input query, we plot the frequency function and list the most correlated queries along with their correlations. Notice that the results for event driven queries, such as `republican national convention`, seem to be better with longer time units, while queries with more periodic frequency functions, such as `disney`, perform best with shorter time units. This makes intuitive sense, since the important changes in frequency for event driven queries are visible only over long time scales, while the interesting variations for periodic queries tend to occur on a daily or weekly basis.

The period of time over which we compute our measure can also affect the results, though less so than the length of time unit. In general, we found that event driven queries, such as `greeting cards` (Figure 6), produced better results over longer time frames while more periodic queries, such as `sears` (Figure 17), performed better in the shorter ones. This is because a longer time frame is needed to capture the peaks for an event driven query such as `greeting cards`, while the pattern for `sears` is established in a short amount of time. In fact, longer time frames may be harmful for periodic queries, since an outlier in a single time unit can severely diminish the correlation coefficient.

Finally, the quality of our results varies as we fine-tune our notion of meaningful correlation. In general, we found that a correlation of 0.9 is a necessary (but not sufficient) predictor of semantic similarity.

3.2 Experimental Results

We first present a sample of the positive results returned by our technique. We again roughly divide these into those that are event driven, and those that are more periodic over time.

Figures 5 through 9 present the frequency functions and top correlated results for six event driven queries. In each of these cases, the frequency function of the query spikes near some event (as described in the captions). Intuitively, these distinctive patterns make it easy for our technique to isolate semantically related (though linguistically unrelated) queries, whose frequency functions show similar behavior, and our technique appears to work particularly well for these queries.

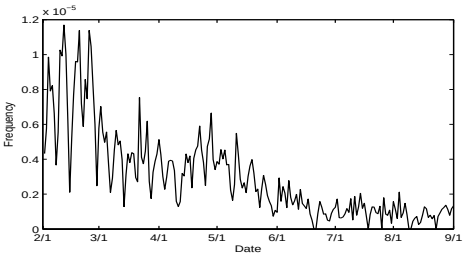
To expand on several examples, note that as February is Black History Month, the query `alice walker` (Figure 5) correlates exclusively with other famous African Americans. The query `greeting cards` (Figure 6), whose frequency function has large peaks around holidays, is temporally related to online electronic greeting card sites. The best correlated results for convicted murderer `scott peterson` (Figure 7) include `court tv`, `laci peterson` (his murdered wife), and `modesto bee` (his hometown newspaper). Finally, the query `superbowl` (Figure 8) correlates with other Super Bowl-related people and events, including terms that are not linguistically similar, such as `tom brady` (a football player), `janet jackson` (the halftime show performer), and `pepsi commercial` (a commercial aired during the game).

More interestingly, our technique is also able to produce interesting results for some classes of non-event driven queries, and we show ten such examples in Figures 10 to 19. These results might be considered surprising, given that the frequency functions of these queries are not as visually distinctive as those of event driven queries. However, it turns out they still contain enough structure for us to find interesting correlations. A comparison of `cnn` (Figure 12), `disney` (Figure 14), and `sears` (Figure 17), for example, shows that the news service is most popular on weekdays (August 1 was a Sunday), the children's entertainment source is more popular on weekends, and the large retailer is fairly constant, thus allowing us to separate these categories. We can also see more subtle differences: note that though both the Atlanta Journal-Constitution newspaper `ajc` (Figure 10) and `cnn` are weekday queries, `ajc` is most popular in the morning, while searches for `cnn` are spread throughout the day, allowing our technique to separate newspapers from television news sources. Thus even these less pronounced temporal features are useful in uncovering semantic relationships.

Other examples we show include queries for banks (`bankone`, Figure 11), airlines (`southwest airlines`, Figure 18), and reference services (`dictionary`, Figure 13, and `yellow pages`, Figure 19), among others.

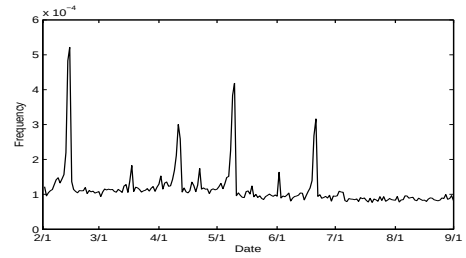
While these examples show that temporal correlation shows promise in finding semantic similarity, it is important to note that it does not provide a complete solution to the problem of query refinement. In order to develop a general measure of the effectiveness of our technique, we analyzed the 300,000 most popular queries during the month of August using three hour time units (this includes all queries that were searched for at least 500 times). Among these, a weighted 20% reported a correlation above 0.9 with at least one other query (the weighting is done by popularity of the queries). Of the 100 most popular of these queries, for a weighted 70%, at least three of their top ten correlations were judged to be in fact semantically related[3]. Thus while we can make reasonable suggestions for some fraction of input queries, many others are not temporally similar enough to other queries for our technique to be useful. This is particularly the case for queries with low total frequency. Further, as our analysis shows, our approach also suggests many false positives. Overall, then, temporal correlation would be

[3]As semantic similarity is inherently subjective, we could find no systematic way to evaluate the results and thus classified them manually.



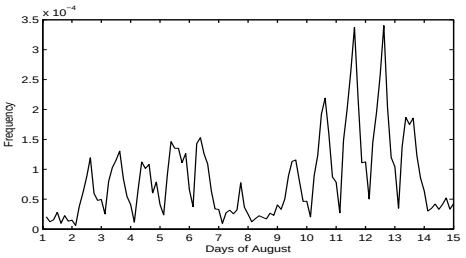
results: michael jordan
martin luther king jr
jackie robinson
ella fitzgerald
malcolm x
frederick douglass
maya angelou
langston hughes
gwendolyn brooks

Figure 5: `alice walker` **results for 2/1-8/31, 24 hour time unit; all listed results are famous African Americans. (Most popular in February, which is Black History Month.)**



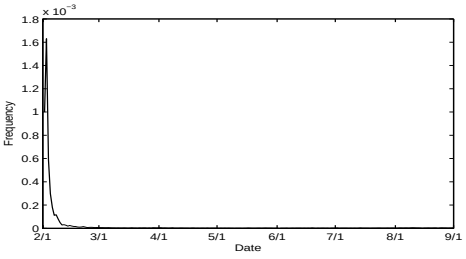
results: free e-cards
egreetings.com
free cards
american greetings
bluemountain.com
yahoo cards
www.hallmark.com
email cards
msn greetings

Figure 6: `greeting cards` **results for 2/1-8/31, 24 hour time unit; bluemountain is an online greeting card site. (Spikes occur on holidays.)**



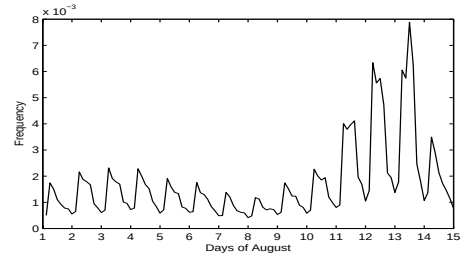
results: court tv
laci peterson
dictionary
modesto bee
scott peterson trial
peterson trial
washington mutual bank
free clip art
thesaurus

Figure 7: `scott peterson` **results for 8/1-8/31, 3 hour time unit; the Petersons were from Modesto. (Spikes occur on days of important testimony in the trial.)**



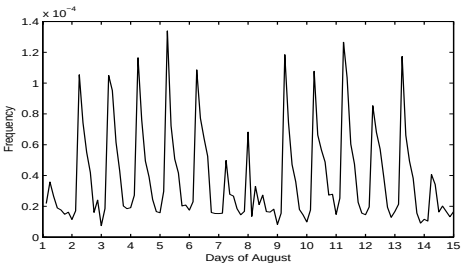
results: superbowl commercials
superbowl halftime show
superbowl xxxvii
superbowl 2004
nfl half time show
superbowl-ads.com
janet jackson
pepsi commercial
tom brady

Figure 8: `superbowl` **results for 2/1-8/31, 24 hour time unit; Tom Brady was the Super Bowl MVP. (The Super Bowl was played on February 1.)**



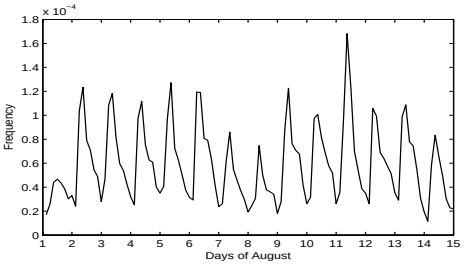
results: weather channel
national weather service
weather underground
intellicast.com
accu weather
twc
nbc6.net
click10
msn weather

Figure 9: `weather` **results for 8/1-9/25, 3 hour time unit; twc is the Weather Channel. (Hurricane Charley made landfall on August 13.)**



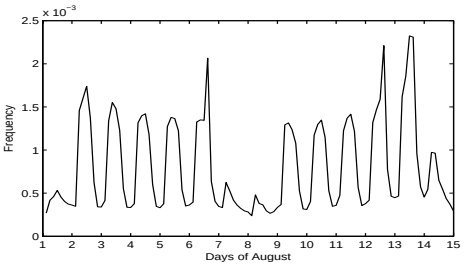
results: cincinnati enquirer
times union
boston herald
baltimore sun
pittsburgh post gazette
bb&t
ny daily news
detroit free press
washingtonpost.com

Figure 10: `ajc` **(Atlanta Journal-Constitution) results for 8/1-9/25, 3 hour time unit**



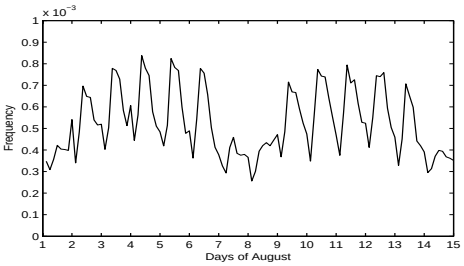
results: fleet
charter one bank
usajobs
wachovia bank
chevy chase bank
bankofamerica
usaa
air tran
amsouth

Figure 11: `bankone` **results for 8/1-9/25, 3 hour time unit; all results other than** `usajobs` **and** `air tran` **are banking related.**



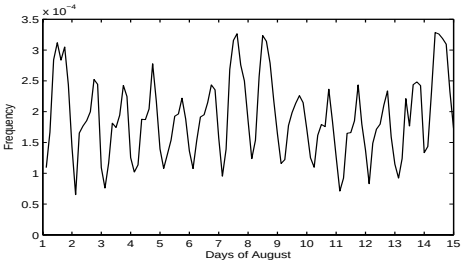
results: fox news
cbs news
drudge
abc news
msnbc news
grainger
wpvi
aol
www.ups.com

Figure 12: `cnn` **results for 8/1-9/25, 3 hour time unit**



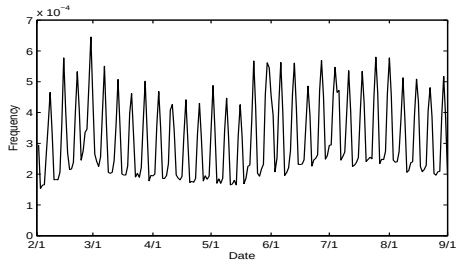
results: websters dictionary
thesaurus
free translation
dictionary.com
register to vote
spanish dictionary
free clip art

Figure 13: `dictionary` **results for 8/1-9/25, 3 hour time unit**



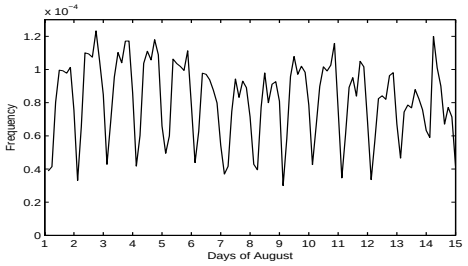
results: barbie.com
postopia.com
noggin.com
cartoon network
neopets.com
pbs kids
nickjr
bratz.com
yugioh.com

Figure 14: `disney` **results for 8/1-9/25, 3 hour time unit; all listed results are children's sites**



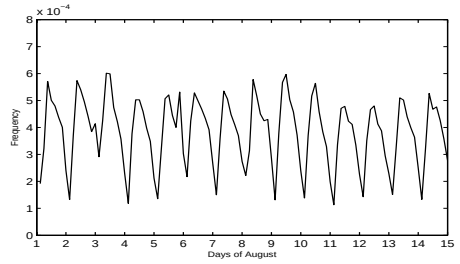
results: amc
ravemotionpictures.com
movie listings
www.movifone.com
regalcinema.com
movies.com
local movies
harkins.com
www.movietime.com

Figure 15: `movies` **results for 2/1-8/31, 24 hour time unit; AMC and Harkins are theater chains.**



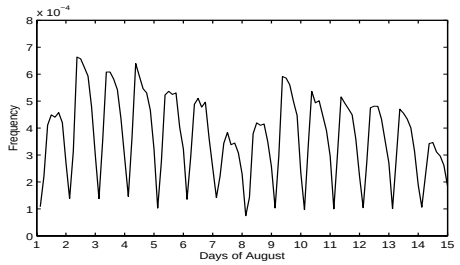
results: cheap tickets
hotwire
www.orbitz.com
hotels.com
boat trader
southwest airlines
www.costco.com
niagra falls
ata airlines

Figure 16: `priceline` **results for 8/1-9/25, 3 hour time unit**



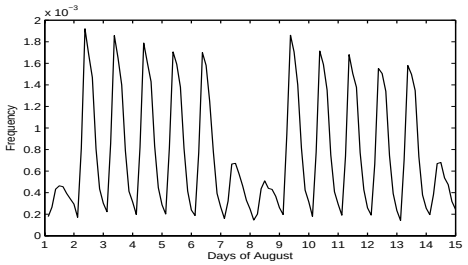
results: barnes and noble
walmart
jcpenny
dell.com
comp usa
circuit city
u haul
office max
lowe's

Figure 17: `sears` **results for 8/1-9/25, 3 hour time unit**



results: orbitz
travelocity
united airlines
american airlines
frontier airlines
expedia
www.fafsa.ed.gov
cheap tickets
netflix

Figure 18: `southwest airlines` **results for 8/1-9/25, 3 hour time unit**



results: zip codes
msn yellow pages
www.whitepages.com
yahoo yellow pages
us postal service
switchboard
social security
federal express
anywho.com

Figure 19: `yellow pages` **results for 8/1-9/25, 3 hour time unit; switchboard and anywho are Internet phone directories.**

most successful when deployed in conjunction with complementary query refinement techniques, and combined with text mining techniques to help filter out false positives.

4. IMPLEMENTATION

In this section, we describe our approach to efficiently find all queries that have high correlation with a given input query. A naive approach to this problem would be to simply compute all correlations with the input query. For a query stream with n queries and d time units, this would require storing dn values and making a linear pass over all of this data for each input query. In order to find these correlations in real time, we adapt techniques from the theory of embeddings [6] and nearest neighbor algorithms [5] to find the *approximate* correlations of queries and thus dramatically reduce the time and space complexity of the computation. We stress that in developing our approach, the emphasis is on simplicity and practicality of implementation.

Recall that the correlation of two queries p and q is:

$$\frac{1}{d}\sum_i \left(\frac{X_{p,i}-\mu(X_p)}{\sigma(X_p)}\right)\left(\frac{X_{q,i}-\mu(X_q)}{\sigma(X_q)}\right),$$

where $X_{p,i}$ is the frequency of p in time i. This can be interpreted as the dot product $\tilde{X}_p \cdot \tilde{X}_q$ of the scaled and normalized frequency vectors

$$\tilde{X}_{p,i} = \frac{1}{\sqrt{d}}\frac{X_{p,i}-\mu(X_p)}{\sigma(X_p)}.$$

Note that this is $\cos\theta_{pq}$, where θ_{pq} is the angle between $\tilde{X}_p$ and $\tilde{X}_q$. Our goal can be restated as finding, for a given input query q, those queries p for which $\cos\theta_{pq}$ is largest, or equivalently, for which θ_{pq} is smallest. Our approach proceeds in two steps:

1. Reduce the data from d frequencies per query to δ bits per query, for a reasonably small constant δ, by defining a mapping $v(\cdot) : \mathbb{R}^d \to \{0,1\}^\delta$ such that the fraction of bits where $v(\tilde{X}_p)$ and $v(\tilde{X}_q)$ agree is roughly proportional to $\tilde{X}_p \cdot \tilde{X}_q$, the correlation of queries p and q. (We chose $\delta = 128$, i.e., 16 bytes per query, for a total of only 240 MB for 15,000,000 queries, which fits easily into memory.)

2. Devise a technique that requires comparing the bits of only a small fraction of queries ($\frac{1}{776}$ in our case) in finding the top correlations for a given input query q, by examining only those queries p that seem likely to have high correlation with q based on the first $k < \delta$ bits of $v(\tilde{X}_p)$.

4.1 Reducing the Data

Our approach first defines a mapping $v(\cdot) : \mathbb{R}^d \to \{0,1\}^\delta$ that represents each frequency vector X_q as a string of δ bits, $v(\tilde{X}_q)$, for some $\delta \ll d$ (the parameter δ depends on the desired accuracy of the approximation but not on the original dimension d; we chose $\delta = 128$). This reduces the amount of storage required from $\Theta(dn)$ integers to a mere δn bits. We do this using random hyperplanes in a manner similar to that of [4, 5, 6]. Our mapping $v(\cdot)$ is defined by δ random vectors $\{r_1, \ldots, r_\delta\} \in \mathbb{R}^d$ drawn from a Gaussian distribution, which we interpret as normal vectors to hyperplanes. For a particular query q, the ith coordinate of $v(\tilde{X}_q)$ is the *sign* of the dot product $\tilde{X}_q \cdot r_i$. The sign essentially records the side of the random hyperplane on which the vector $\tilde{X}_q$ falls.

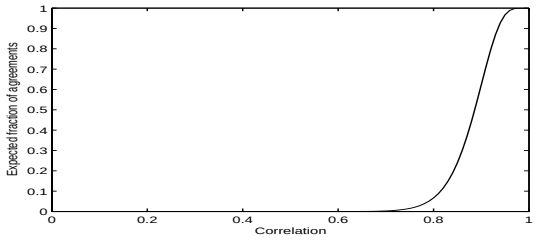


Figure 20: The probability a query is returned by our technique versus the correlation of that query with the input query.

We claim that this embedding preserves our ability to find highly correlated queries, since for any two vectors $\tilde{X}_p$ and $\tilde{X}_q$, the random variable τ_{pq}, denoting the fraction of bit agreements between $v(\tilde{X}_p)$ and $v(\tilde{X}_q)$, has expectation $1 - \frac{\theta_{pq}}{\pi}$ and low variance. To verify the expectation, note that the probability that $v(\tilde{X}_p)$ and $v(\tilde{X}_q)$ agree in the ith component (i.e., that $\text{sign}(\tilde{X}_p \cdot r_i) = \text{sign}(\tilde{X}_q \cdot r_i)$) is simply $1 - \theta_{pq}/\pi$, or the probability that two vectors with an angle θ_{pq} between them are on the same side of a random hyperplane. The low variance follows from the observation that each random vector r_i is chosen independently and a standard Chernoff bound argument.

Thus if we are interested in finding queries whose correlation is at least a given threshold λ with an input query, we can simply return those queries whose embeddings agree on a fraction $p_\lambda = 1 - \arccos(\lambda)/\pi$ of bits. As an example, when $\lambda = 0.9$ (the threshold used throughout this paper), we have $p_\lambda = 0.8564$. Figure 20 shows the probability that two queries p and q of a particular correlation $\cos\theta_{pq}$ have $\tau_{pq} \geq 0.85$ for $\delta = 128$. Note the somewhat sharp threshold: for two queries p and q with correlation at least 0.9, $\Pr[\tau_{pq} \geq 0.85] \geq 0.62$ while for two queries p and q with correlation at most 0.8, $\Pr[\tau_{pq} \geq 0.85] \leq 0.07$.

Online Implementation: One important feature of our proposed embedding is that it can be implemented in an online fashion using just $O(\delta n)$ integers. In each time unit, we keep the exact value of the frequency of each query. At the end of the jth time unit, our algorithm generates the next random coordinate r_{ij} of each of the δ random vectors r_i. It stores the current value of all δn dot products of the form $r_i \cdot X_p$, i.e., it updates the sum $\sum_{k=1}^{j} r_{i,k} X_{p,k}$. It also tracks the mean frequency of each the n queries after each time unit. This information is sufficient to compute $\text{sign}(\sum_{k=1}^{d} r_i \cdot (X_p - \mu(X_p)))$. Since normalizing the random vector r_i and dividing the sum by the variance of X_p do not affect the sign of the result, the method described does in fact compute the mapping $v(\cdot)$ defined above.

We mention that all results presented in this paper were computed using the technique presented in this section.

4.2 Finding Top Correlations Efficiently

Using the embedding technique above, we can now find the (approximate) highest correlated queries above a certain threshold to an input query q using just $O(\delta n)$ bit compar-

isons by simply computing the number of bit agreements between $v(\tilde{X}_p)$ and $v(\tilde{X}_q)$ for all queries p.

Considering that we only want to return queries with correlation above the given threshold λ, we can reduce the running time of our technique even further by limiting the number of queries p that we examine. In particular, we examine only those queries p that seem likely to have high correlation with q based on the first k bits of $v(\tilde{X}_p)$. We create 2^k buckets indexed by all strings in $\{0,1\}^k$ and store each query q in the bucket indicated by the first k bits of its embedding $v(\tilde{X}_q)$. For each bucket b, we define a bucket b' as *close* to b if their indices agree in at least a ρ fraction of the bits, for some parameter $\rho \leq p_\lambda$. Given an input query q, we only search for high-correlation queries p in the bucket of q and in buckets close to the bucket of q. (Note that we still compare all δ bits for the queries in these buckets, so that we eliminate false positives generated by this process.) The intuition is that queries whose embeddings agree in at least a fraction p_λ of their bits will likely agree in at least a fraction ρ of their first k bits, so that we still find most correlated queries. By only searching through close buckets, however, we greatly reduce the number of queries for which we have to perform bit comparisons.

For the parameters we recommend, with $k = 20$, $\lambda = 0.9$, and $\rho = 0.85$, each query with correlation above λ will fall into a close bucket with probability 0.68. Combined with the loss induced by the correlation estimate, this implies that an expected 0.42 fraction of queries with correlation 0.9 are returned by our technique (with better expectations for higher correlations). Further, for any given bucket, only 1351 of the $2^{20} = 1048576$ buckets are close, meaning that in expectation we only need to examine about 1/776 of the queries. For $n = 15,000,000$, this comes out to only $19,330$ queries, and we note that in our experiments we did not generally search through more than $40,000$ queries.

5. FUTURE DIRECTIONS

There are a number of future directions to consider for this work. In our current approach, we divide a query stream into bins, where bins correspond to time units, and aggregate query information within these bins. It would be interesting to generalize this approach to other notions of bins beyond time units, possibly as follows:

- *Regionalization.* Using the client IP address information available in query logs, we can further restrict the idea of bin to be not just a time unit, but also a geographic region. This may allow us to find correlations unique to a particular area.

- *Time zones.* Again using client IP addresses, we can instead count how often queries occur within *local* time intervals (e.g., how often a query q occurs between 6 am and 9 am local time), allowing us to study and possibly remove the effect of time zones.

It is also interesting to consider how to determine which pairs of highly correlated queries are semantically related and which are false positives. Our current approach uses a 0.9 threshold to determine semantic similarity and admits many false positives. An alternative approach might be to run a clustering algorithm on the weighted complete graph whose nodes are queries and whose edges are the correlation estimates of the corresponding endpoints. Unfortunately, this may be prohibitively expensive computationally.

Further, one weakness of our current approach is that it is limited to queries that are common enough to have non-trivial frequency functions. An open question is whether it is possible to produce meaningful results for those queries that occur very rarely.

More generally, we note that because query streams are often proprietary, their potential for improving search is still largely untapped, and that much research is still left to be done in this area.

Acknowledgements

We are grateful to Robert Ragno for processing and providing us with the two month query stream. We would also like to thank Dennis Fetterly, Mohammad Mahdian, Mark Manasse, Frank McSherry, Chris Meek, Marc Najork, Madhu Sudan, and Kunal Talwar for useful discussions, as well as the anonymous referees for very helpful suggestions.

6. REFERENCES

[1] Doug Beeferman and Adam L. Berger. Agglomerative clustering of a search engine query log. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, USA, 2000.

[2] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Eleventh International World Wide Web Conference (WWW)*, Honolulu, Hawaii, 2002.

[3] H. Daume and E. Brill. Web search intent induction via automatic query reformulation. In *Human Language Technology Conference / North American chapter of the Association for Computational Linguistics (HTL/NAACL)*, Boston, MA, USA, 2004.

[4] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42(6):1115–1145, 1995.

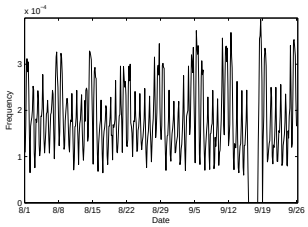
[5] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Thirtieth Annual ACM Symposium on the Theory of Computing*, Dallas, Texas, USA, 1998.

[6] W. Johnson and J. Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[7] Reiner Kraft and Jason Y. Zien. Mining anchor text for query refinement. In *13th international conference on World Wide Web (WWW)*, New York, NY, USA, 2004.

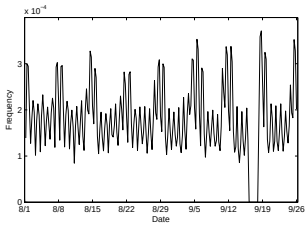
[8] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identification of similarities, periodicities and bursts for online search queries. In *International Conference on Management of Data (SIGMOD)*, Paris, France, 2004.

[9] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In *Tenth International World Wide Web Conference (WWW)*, Hong Kong, China, 2001.



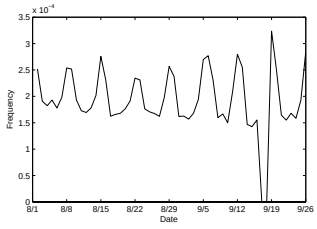
barbie.com (0.96)
postopia.com (0.94)
noggin.com (0.93)
cartoon network (0.93)
www.neopets.com (0.91)
pbs kids (0.91)
nickjr (0.91)
bratz.com (0.91)
yugioh.com (0.90)

Figure 21: disney **3 hour time unit**



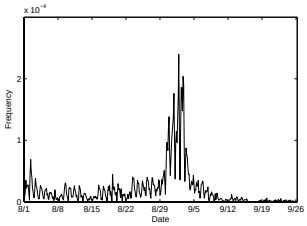
cartoonnetwork.com (0.96)
nickjr.com (0.96)
barbie.com (0.96)
pbs kids (0.95)
www.lego.com (0.94)
www.myscene.com (0.93)
polly pocket (0.93)
noggin (0.93)
nickelodeon.com (0.93)

Figure 22: disney **6 hour time unit**



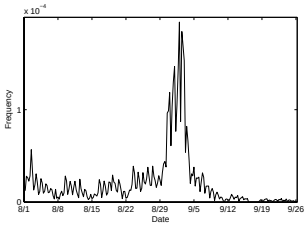
free ringtones (0.99)
xanga (0.99)
music videos (0.99)
icq (0.99)
free porn (0.99)
download.com (0.99)
chemical products (0.99)
xxx (0.98)
sex (0.98)

Figure 23: disney **24 hour time unit**



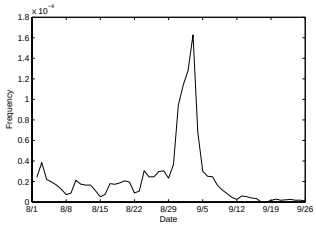
republican convention (0.94)
hurricane francine (0.91)
gop (0.90)

Figure 24: republican national convention **3 hour time unit**



republican convention (0.99)
rnc (0.99)
gop convention (0.94)
latin grammy (0.89)

Figure 25: republican national convention **6 hour time unit**



republican convention (0.99)
rnc (0.99)
gop convention (0.99)
rnc convention (0.99)
national republican convention (0.98)
hurricane florence (0.98)
2004 republican convention (0.95)
cspan (0.94)
projected path of hurricane frances (0.94)

Figure 26: republican national convention **24 hour time unit**