SVD Subspace Projections for Term Suggestion Ranking and Clustering

David Gleich [*]
Harvey Mudd College
Claremont,CA
dgleich@cs.hmc.edu

Leonid Zhukov
Yahoo! Research Labs
Pasadena, CA
leonid.zhukov@overture.com

ABSTRACT

In this manuscript, we evaluate the application of the singular value decomposition (SVD) to a search term suggestion system in a pay-for-performance search market. We propose a novel positive and negative relevance feedback method for search refinement based on orthogonal subspace projections. We apply these methods to the subset of Overture's market data and demonstrate the effect of SVD and subspace projections on search results.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Relevance feedback, Retrieval models; G.1.3 [**Numerical Linear Algebra**]

General Terms

Algorithms, Experimentation

Keywords

Latent Semantic Indexing, Singular Value Decomposition, Vector Space Model, Relevance Feedback

1. INTRODUCTION

In a pay-for-performance search market, advertisers compete in online auctions by bidding on search terms for sponsored listings in affiliated search engines. Because of the competitive nature of the market, each keyword may have bids from many advertisers and almost every advertiser bids on more than one search term. The practical goal of our work was creating a "term suggestion" tool, which, for any given search term, provides a sorted list of correlated terms and suggests them to the user. In this analysis, we consider applications of a singular value decomposition (SVD) [9] to term suggestion ranking. The advertisers, bidded search terms, and placed bids constitute the dataset.

One of the desired features in the term suggestion tool was to control the level of "generality" of suggested terms. To that end, we decided to use a vector space model and an SVD based approach. An alternative approach would be to cluster the data and return terms from the same cluster [5]. The vector space model-based approach avoids using a rigid clustering and allows each query to potentially form a "soft cluster" of relevant results.

Latent semantic indexing (LSI) [7, 8] uses an SVD based-method to expose and search semantic information within a dataset. Most papers cite the use of LSI to enhance text information retrieval systems, [2, 3, 6]. In this context, LSI is used to compute a document-query similarity score for each document in the collection. However, as noticed in [7, 11], we can also compute the similarity between documents and other documents, between documents and terms, and between terms and other terms. In this paper, we focus on using LSI for term-term similarity. For clarity of notation, when we refer to LSI, we are referring to the process of computing similarity scores for a query through projection into the SVD subspace.

According to the vector space model, every search term in a dataset is represented as a vector in a space of all advertisers with nonzero entries corresponding to advertisers bidding on this term. Then, the proximity between search terms can be measured as the cosine of the angle between corresponding term vectors. We can calculate the proximity between any given search term and the rest of the terms in the collection and then sort the retrieved terms according to this proximity measure.

Using the vector space approach guarantees retrieval of terms whose advertisers bid on the search term (exact match). Using LSI, we can also perform a conceptual match [7, 2] and might significantly expand the number of suggested terms and also change their ranking. In other words, LSI enables us to match terms globally, or conceptually, without the need for explicit connections.

While this paper is mainly experimental, it provides additional insight into LSI's properties and behavior. The demonstrated effect of the dimensionality reduction is rather general, since our dataset has a power law distribution, typical for any text collection data.

[*]Work performed while at Yahoo! Research Labs.

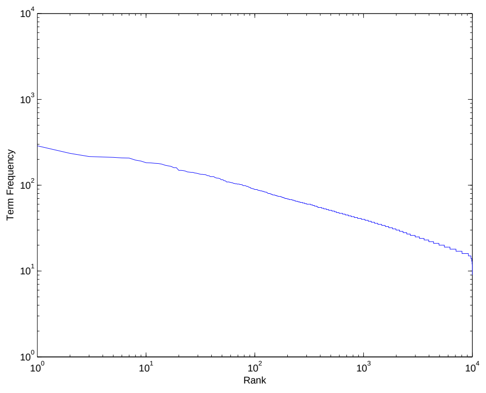


Figure 1: A log-log plot of the number of advertisers bidding on each term. The results are sorted so that the term with the most bids is first, and the term with the least bids is last. The plot indicates a power law distribution of the data.

2. DATA

In this study, we use a small, densely connected subset of Overture's United States term-advertiser data with 10,000 bidded search terms, 8,850 advertisers, and more than 250,000 bids.

Our representation for the data is a term-advertiser matrix, A, whose columns correspond to advertisers and rows to bidded search keywords, that is, search terms. The number of rows in this matrix, m, is equal to the number of unique bidded keywords, and the number of columns, n, is the number of unique advertisers active on the market. This matrix is strictly non-negative and is also sparse, since the majority of the advertisers bid only on a small number of terms. Thus, every column of this matrix represents an advertiser vector described in the bidded keyword space and every row is a bidded keyword vector in the advertiser space. This arrangement follows the traditional vector space model [1] for a textual collection, where every column is a word distribution histogram for a document.

The matrix A is normalized using the binary frequency variant of term-frequency, inverse document frequency normalization [3],

$$A_{ij} = \frac{\chi_{ij} \log(n/n_i)}{\sum_{j=1}^{m} [\chi_{ij} \log(n/n_i)]^2}, \quad (1)$$

where χ_{ij} is 1 if advertiser j bids on term i, n is the total number of advertisers in the collection, and n_i is the number of advertisers which bid on term i.

Additionally, the term-advertiser data obeys a power law distribution, as seen in Figure 1. Thus, the results obtained in this study, could be applicable to other data that follow a power law distribution, such as textual data [1].

3. METHOD

In this section, we describe methods in our process. First, we depict projection operators from linear algebra. Second, we discuss the cosine distance measure and its interactions with projection operators. Next, we discuss projections in the latent semantic subspace and how we use these projections to rank term suggestions. Finally, we introduce a novel method of relevance feedback using subspace projections.

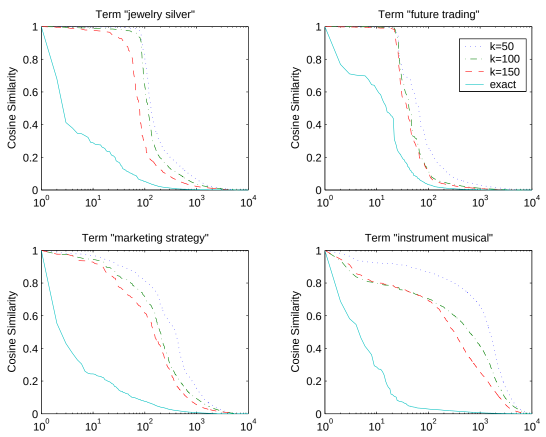


Figure 2: The cosine similarity results from four terms. The ordered terms are on the logarithmic horizontal axis and the similarity for four different techniques is shown. The solid line is the exact cosine similarity; the dotted lines show the cosine similarities in three SVD subspaces. See Table 1 for the top terms suggested.

3.1 Projection Operators

An orthogonal projection on a unit vector v is defined as $\hat{P} = v^T v$, where v is a column vector. An orthogonal projection on a subspace spanned by a collection of k vectors $v_1, v_2, \ldots, v_k$ is given by a projector operator

$$\hat{P}_k = \sum_{i=1}^{k} v_i v_i^T = V_k V_k^T. \quad (2)$$

Any projection operator has the property that $\hat{P}_k \hat{P}_k = \hat{P}_k$ and orthogonal projections have the additional property that $\hat{P}_k^T = \hat{P}_k$.

3.2 Cosine Distance Measure

For any two terms, t_i and t_j, we define the similarity metric as a cosine of the angle between corresponding vectors,

$$sim(t_i, t_j) = \cos(t_i, t_j) = \frac{t_i^T \cdot t_j}{||t_i||\ ||t_j||}. \quad (3)$$

In the subspace, defined by its orthogonal projection P_k, the similarity (cosine of the angle) between vector projections is

$$sim(P_k t_i, P_k t_j) = \cos(P_k t_i, P_k t_j) = \frac{(P_k t_i)^T \cdot (P_k t_i)}{||P_k t_i||\ ||P_k t_j||}. \quad (4)$$

Using Eq. 2, the scalar product between vectors can be expressed as

$$(P_k t_i)^T \cdot (P_k t_j) = t_i^T P_k t_j = (V_k^T t_i) \cdot (V_k^T t_j), \quad (5)$$

jewelry silver		**future trading**	
jewelry silver	1	future trading	1
necklace silver	0.998	future option	0.999
jewelry silver sterling	0.996	commodity market	0.999
fine jewelry	0.994	commodity future trading	0.999
gold jewelry	0.993	future quote	0.999
jewelry wholesale	0.992	commodity trader	0.999
necklace pearl	0.992	commodity trading	0.998
bracelet silver	0.991	future online trading	0.998
jewelry pearl	0.991	commodity option	0.998
silver sterling	0.990	commodity online trading	0.998

Table 1: The top ten suggestions and their cosine similarity for two terms from Figure 2 using LSI with $k = 100$.

and the vector norm in the subspace is given by

$$||P_k t||^2 = (P_k t)^T (P_k t) = t^T P_k t = ||V_k^T t||^2. \quad (6)$$

3.3 SVD Subspace Projections

We first decompose term-advertiser matrix A using the singular value decomposition,

$$A = USV^T. \quad (7)$$

The first k columns of the matrix V form the truncated orthogonal subspace, V_k. Note, that we are using V_k instead of U_k since we are interested in terms space, not advertiser space, and thus it might be easier to think about $A^T = VSU^T$ factorization instead. The number of columns in V_k is equal to the rank k of the subspace we use. Every column of V_k is a basis vector, and any search term in the dataset can be represented as a linear combination of V_k and $S_k U_k^T$, where the columns of $S_k U_k^T$ are the coefficients of the terms in the V_k subspace. The Eckart and Young theorem [9] guarantees that top k singular vectors provide the best (closest in the L_2 norm sense) approximation of the data vectors A_k to A in any basis of the order k.

3.4 Search term ranking

A query q is a search term represented in the advertiser space, or in other words, a query is a column a_i of the matrix A^T; alternatively, it is a row of the matrix A. Mathematically, it is convenient to express $q_i = a_i = A^T e_i$, where e_i is a column vector of all zeros except for a position corresponding to the column of interest in the matrix A^T, or row in the matrix A. The angle between a query vector and any other term vector in the matrix is

$$sim(t_i, q_j) = \cos(t_i, q_j) = \frac{a_i^T a_j}{||a_i||\,||a_j||} = \frac{(A^T e_i)^T (A^T e_j)}{||a_i||\,||a_j||} = \frac{(AA^T)_{ij}}{||a_i||\,||a_j||}, \quad (8)$$

which is a normalized inner product of A^T columns, or A matrix rows.

The similarity for the same vectors in the SVD orthogonal subspace is given by

$$sim(V_k^T t_i, V_k^T t_j) = \cos(V_k^T t_i, V_k^T q_j) = \frac{(V_k^T A^T e_i)^T (V_k^T A^T e_j)}{||(V_k^T A^T e_i)^T||\,||(V_k^T A^T e_j)||} = \frac{(SU_k^T)_i^T (SU_k^T)_j}{||(SU_k^T)_i^T||\,||(SU_k^T)_j||}. \quad (9)$$

The above result of SVD decomposition is equivalent to the use of eigen-decomposition (i.e. principal component analysis) on a correlation (affinity) term-term matrix AA^T, where $\Lambda_i = S_i^2$ and U_i are eigenvectors, that is,

$$(AA^T)U_i = U_i \Lambda_i. \quad (10)$$

Figure 2 demonstrates the cosine similarity scores for all terms in the dataset to four different search terms and how the similarity scores change when projecting into various SVD subspaces. Table 1 displays the top 10 suggested terms. Finally, Figure 3(a) elaborates on the differences between the exact cosine and SVD subspace cosine similarities.

3.5 Relevance Feedback

We can iteratively refine the results when the user chooses terms from the returned results to reinforce or reject the ranking, thus performing *positive* or *negative* refinements.

By positive refinement, we mean the user selecting positive, reinforcing examples to his query from the provided term list. Then, instead of using q_0 as a query, we can construct a new extended query, that spans the space $\{q_0, a_{j1}, a_{j2}, ..a_{jp}\}$, where a_j is the j-th column of A^T corresponding to the term that the user choses to reinforce the query. Notice, that we are not computing the centroid for a new query as in [3, 2], but rather are measuring the angle between the terms and an extended query subspace. If the space is formed by non-orthogonal vectors, the projection operator on that subspace is given by [13].

$$\hat{P}_q = Q(Q^T Q)^{-1} Q^T. \quad (11)$$

When vectors q_i forming the subspace are orthogonal, i.e. $Q^T Q = I$, then the projection operator reduces to Eq. (2).

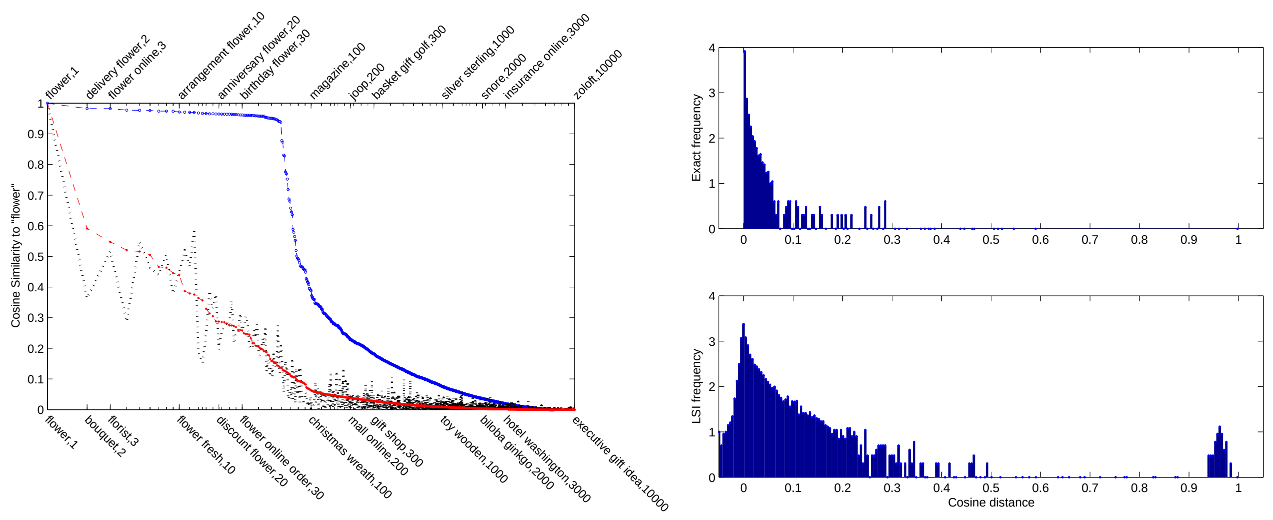
The angle between a term and the positive term subspace is defined as the angle between a term and its orthogonal projection on that subspace. Formally,

$$sim(t_i, q') = \frac{t_i^T \hat{P}_q t_i}{||t_i||\,||P_q t_i||}. \quad (12)$$

This feedback mechanism works in both the entire space and SVD subspace. For the SVD subspace, instead of t_i we use $t_{ik} = V_k V_k^T t_i$ and Q is formed using $a_{ik} = V_k V_k^T a_i$. Table 2 displays the change in search term ranking after positive refinement, and Figure 4 shows the change in similarity scores with positive refinement.

Negative refinement allows users to choose irrelevant documents and force the search results to be orthogonal to them. Thus, we are looking for a vector term in the collection with the smallest angle with the query and, at the same time, orthogonal to the negative term vectors specified by the user. Again, we want to emphasize that our method will produce results orthogonal to the entire subspace spanned by negative examples and not to only those terms[1]. In this case, we need to build a complementary projector to the negative

[1]The subspace spanned by examples means that the returned result will be orthogonal to any possible linear combination of negative documents



(a) Rank ordered similarity scores. (b) Histograms of similarity scores.

Figure 3: The suggested results for the term "flower" from the exact (non-LSI) cosine method and the LSI method with $k = 100$. On the lower horizontal axis of (a), we include the suggestions at various points on the logarithmic scale from the LSI results; these terms correspond to the upper curve. The upper horizontal axis has suggestions from the exact approach, corresponding to the lower curve. The dotted line in the middle shows the exact results plotted with the ordering from LSI, i.e the lower curve ordered on the lower axis. In (b), the horizontal axis is the similarity score and the vertical axis shows the logarithm of the number of results with that similarity score. Since the vertical axis is logarithmic, the area underneath the histograms is equal, although it appears uneven.

examples space,

$$\hat{P}_{qn} = I - \hat{P}_q. \tag{13}$$

Then the new similarity score becomes

$$sim(t_i, q') = \frac{t_i^T(I - \hat{P}_q^T)t_i}{||t_i||\ ||(I - \hat{P}_q^T)t_i||}. \tag{14}$$

4. IMPLEMENTATION

We developed two programs for this work. The first is a program to compute the truncated SVD of a sparse matrix using the Implicitly Restarted Lanczos Method implemented in ARPACK [12], and ARPACK++ [10] libraries. The second is a Java program to query a dataset and retrieve results between general and specific associations (Figure 6 shows the user interface in the Java application).

While the formulas presented in the previous section provide a compact description of the operations, they are extremely inefficient as written. For example, while the original matrix A is around 3 MB in a sparse matrix representation, the matrix AA^T is more than 300 MB. Thus, we needed to plan the application solely using sparse and dense matrix vector multiplications.

Instead of directly computing the final form of Equation 8, we first row-normalize the sparse matrix A to $\hat{A}$ then compute the intermediate vector $a_i = \hat{A}^T e_i$ through a sparse matrix-vector multiply. This operation extracts the query vector, row i from matrix A, which is already normalized. Then we compute the matrix vector product Aa_i, which simultaneously computes the cosines for each term in the collection. We perform a similar operation using the precomputed dense matrix U_kS to compute the cosines in the LSI subspace as in Equation 9.

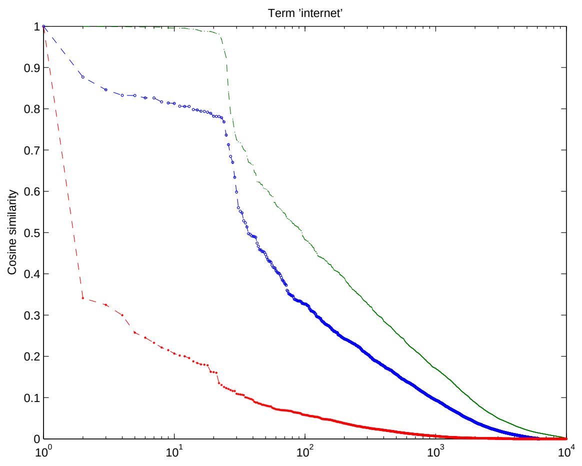
Finally, before we perform any projection operations, we orthogonalize vectors using the Gram-Schmidt process [9]. This operation is computationally more efficient and stable than inverting the matrix in the non-orthogonal case.

"internet"	**"internet" + "cheap isp"**
internet	cheap isp
computer internet	internet
free internet	low cost isp
internet service	cheap internet service
access	national isp
isp	unlimited internet access
isps	cheap internet service provider
internet provider	cheap internet access
isp provider	isp provider
dial up internet access	internet access provider

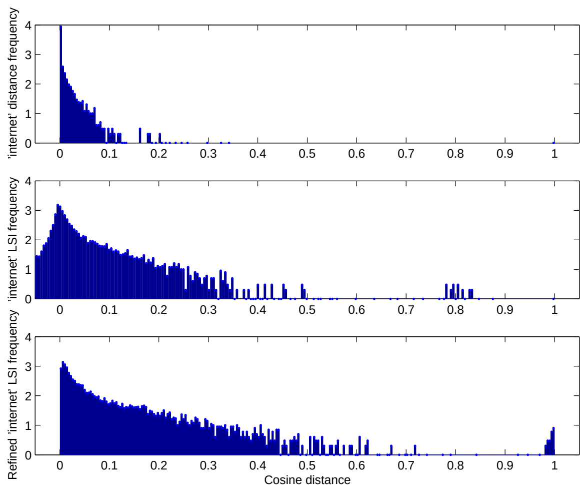
Table 2: Changes in the top 10 suggested terms after positive refinement on the term "cheap isp," that is, inexpensive internet service providers. "Cheap isp" was the 18th result.

5. EXPERIMENTAL RESULTS

Examination of the results from the SVD subspace projection and the exact cosine similarity reveals two interesting features. First, projecting into the SVD subspace appears to form clusters of data for certain terms. Second, the distance distortion between the results returned by LSI and the exact



(a) Rank ordered similarity scores.



(b) Histograms of similarity scores.

Figure 4: Similarity result curves with user feedback. The bottom curve shows the exact similarity results for the term "internet;" the middle curve is for $k = 100$ LSI similarity; and the upper curve is for $k = 100$ similarity with positive refinement on the term "cheap isp." The vertical scale in (b) is logarithmic. The reason the LSI results display negative cosine values is that in we normalize the results when projecting into the SVD subspace, thus the vectors lie on a hypersphere, so the maximum angle is π. In the exact space, the matrix is positive so the vectors are all in the same quadrant,i.e. the maximum angle is $\pi/2$. When positively refining, we project a vector onto a subspace and measure the angle between the projected vector and the original vector. This angle is always less than or equal to π.

cosine method is quite small.

5.1 Clustering Behavior

Figure 3(a) demonstrates the clustering behavior that occurs with LSI, but not with the exact cosine similarity. The plateau at the top of the LSI curve represents a set of results whose similarity scores are high. If we take the histogram of the distances, like in Figure 3(b) this behavior becomes even more apparent. The cluster at the right of the LSI histogram represents the "flower" cluster in the data. It is interesting to note, that the steep decline of the LSI curve corresponds to the end of related terms.

This behavior, however, does not occur for all terms in the dataset. In Figure 2, the terms "marketing strategy" and "instrument musical" do not display any clear clustering behavior. Additionally, the histogram for the term "internet" in Figure 4(b) only shows a weak cluster near the similarity score 0.8. Note that with positive refinement on the term "cheap isp" from "internet" a strong cluster of terms emerges.

5.2 Distance Distortion

The dotted line in the middle of Figure 3(a) represents the distortion between the distances in LSI subspace and in complete space. That is, we plot the similarity values from the exact cosine sorting results using the ordering from the LSI results. Of note in this figure is that there is very little distortion of the results beyond the plateau. This fact indicates that LSI only performs a local reordering of the results.

Number of Dimensions	$\sum_i \cos(q, a_i)^2$
$k = 25$	935.9
$k = 50$	284.8
$k = 75$	217.8
$k = 100$	190.5
Exact	7.17

Table 3: The sum of squared angle-cosines between the query term "flower" and all other terms in the dataset. As per Brand's polarization theorem, this sum is strictly decreasing as k increases. The exact result is the sum in the original space.

6. DISCUSSION

Our results indicate that as the data representation is truncated, the angles between vectors become smaller causing the cosine values to increase as in Figures 2, 3, and 4. One explanation for these results comes from [4]. In that paper, the author suggests that as the dimensionality of the representation is reduced, the distribution of cosines (similarity measures) migrates away from zero to ± 1. This behavior is easily seen in Figures 3 and 4. Mathematically, this statement follows from a *polarization theorem* [4] that states when a positive matrix A is projected to successively lower dimensions $A_{D-1}, \ldots, A_k, \ldots, A_1$, the sum of squared angle-cosines between projected column vectors, $\sum_{i \neq j} [\cos(t_i, t_j)]^2$, is strictly increasing. Table 3 shows that our numerical results confirm this statement.

Thus, the singular value decomposition of A exaggerates the structure of A by reducing the angle between similar vectors and increasing the angle between dissimilar vectors.

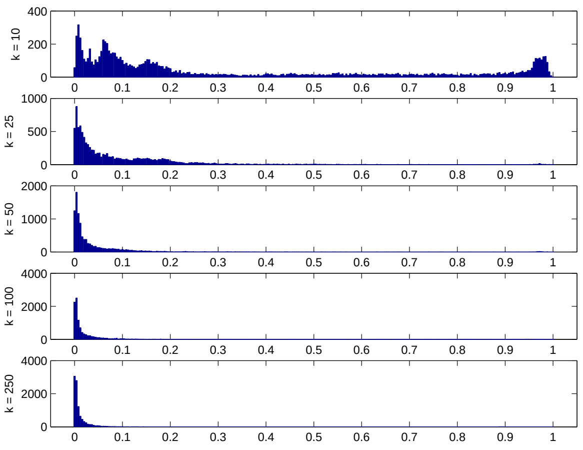
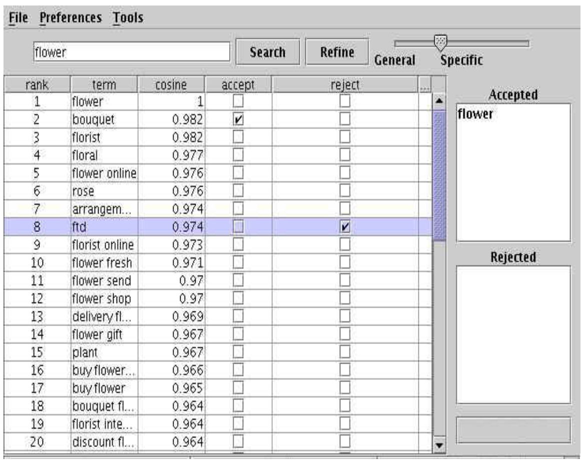


Figure 5: Histograms of the distortion amounts for the term "flower" with increasing k. The distortion measures how much the results change between the exact cosine similarity and the LSI similarity.



flower

rank	term	cosine	accept	reject
1	flower	1	☐	☐
2	bouquet	0.982	✓	☐
3	florist	0.982	☐	☐
4	floral	0.977	☐	☐
5	flower online	0.976	☐	☐
6	rose	0.976	☐	☐
7	arrangem...	0.974	☐	☐
8	ftd	0.974	☐	✓
9	florist online	0.973	☐	☐
10	flower fresh	0.971	☐	☐
11	flower send	0.97	☐	☐
12	flower shop	0.97	☐	☐
13	delivery fl...	0.969	☐	☐
14	flower gift	0.967	☐	☐
15	plant	0.967	☐	☐
16	buy flower...	0.966	☐	☐
17	buy flower	0.965	☐	☐
18	bouquet fl...	0.964	☐	☐
19	florist inte...	0.964	☐	☐
20	discount fl...	0.964	☐	☐

Accepted: flower

Figure 6: This is a screenshot of the user interface to our term suggestion tool. The table shows the suggestions for the word "flower," using a fairly specific level of generality ($k > 100$).

Returning to Figure 3(a) and 3(b), the plateau in the results from LSI clearly shows how term vectors similar to "flower" are almost collapsed to the same point. The terms, "jewelry silver" and "future trading" from Figure 2 also demonstrate this effect. However, there is no such result for "marketing strategy" and "instrument musical." In these cases, we believe that no set of similar vectors exists.

Since the steep decline in the LSI curve corresponds to the end of the related terms for the query "flower," identifying this cutoff suggests a natural way to cluster a dataset using SVD subspace projections. For the terms, "marketing strategy" and "instrument musical," then, there is no good cluster in the data for these terms. We observed that a topic generally does not correspond to a single axis in the SVD subspace and accurate clustering requires working in subspaces.

The local distortion effect identified in the previous section reinforces our belief in LSI's clustering ability. The largest distortion in the angles is between vectors that are already similar (the left side of the curve in Figure 3(a)). Thus, projecting into the SVD subspace causes clusters of vectors to become tighter. In Figure 5, we computed histograms of the amount of distance distortion at various similarity scores. The histograms showed that for $k > 50$, the distortion tends to be small, i.e. the reordering is local. Thus, the projections with more than 50 dimensions are strong enough to separate the topics, but still cause only small distance distortion and reordering.

7. SUMMARY AND CONCLUSIONS

We investigated the effect of SVD subspace projections on data from a pay-for-performance advertising market. We developed a tool to suggest related terms at varying levels of generality by varying the rank of the SVD subspace. Additionally, we developed a novel relevance feedback system for positive and negative examples using vector subspaces.

The results from our system, along with the polarization theorem [4] for reduced dimensional projections, suggest that projection into the SVD subspace clusters the dataset. The unique falloff pattern in the similarity scores suggests a method to build clusters from the data.

8. ACKNOWLEDGMENTS

We would like to thank Professor Al Barr for illuminating discussions on the properties of dimensionality reducing projections.

9. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] M. Berry, Z. Drmac, and E. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[3] M. W. Berry and M. Browne. *Understanding search engines: mathematical modeling and text retrieval.* Society for Industrial and Applied Mathematics, 1999.

[4] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, January 2003.

[5] J. J. M. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *International Conference on Data Mining, 2003, Data mining workshop*, 2003.

[6] C.-M. Chen, N. Stoffel, M. Post, C. Basu, D. Bassu, and C. Behrens. Telcordia lsi engine: Implementation and scalability issues. In *Proceedings of the Eleventh International Workshop on Research Issues in Data Engineering*, pages 51–58, April 2001.

[7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by

latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] S. Dumais. Improving the retireval of onformation from external sources. *Behaviour Research Methods, Instruments, and Computers*, 23(2), 1991.

[9] G. H. Golub and C. F. V. Loan. *Matrix Computations*. John Hopkins Univ. Press, 1989.

[10] F. R. Gomes and D. Sorensen. Arpack++. http://www.ime.unicamp.br/ chico/arpack++/, 1998.

[11] J. Kleinberg and A. Tomkins. Applications of linear algebra in information retrieval and hypertext analysis. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 185–193. ACM Press, 1999.

[12] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users' Guide: Solutions of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, 1997.

[13] L. N. Trefethen and I. D. Bau. *Numerical Linear Algebra*. SIAM, 1997.