SVD based Term Suggestion and Ranking System

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

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

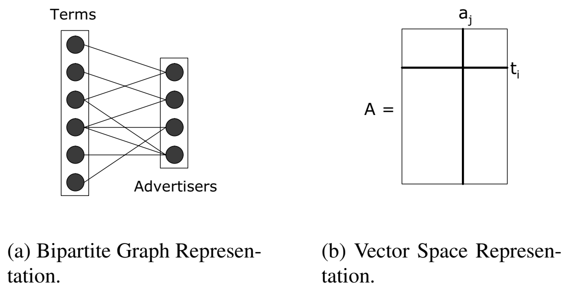
Abstract

In this paper, we consider 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 refinement method based on orthogonal subspace projections. We demonstrate that SVD subspace-based methods: 1) expand coverage by reordering the results, and 2) enhance the clustered structure of the data. The numerical experiments reported in this paper were performed on Overture's pay-per-performance search market data.

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 search term may have bids from many advertisers, and almost every advertiser bids on more than one search term. The practical motivation for our work was creating a "term suggestion" tool, which, for any given search term, provides a sorted list of relevant search term suggestions from the existing database. One of the desired features of this term suggestion tool is a smoothly controlled level of "generality" of suggested terms. To that end, we decided to use both a vector space model and a singular value decomposition (SVD) [5] based approach to term ranking and suggestion.

It is well known that Latent Semantic Indexing (LSI) [3, 4], an SVD based-method, can help to expose semantic information within a dataset. Most papers cite the use of LSI to enhance text information retrieval systems [1]. In this context, LSI is used to compute a document-query similarity score for each document in the collection. However, as noticed in [3], 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 SVD for term-term similarity.



(a) Bipartite Graph Representation. (b) Vector Space Representation.

Figure 1. Data representation.

The vector space approach allows retrieval of all terms directly correlated with the search term – we call this an exact match. Using SVD, we can also perform a conceptual match [1, 3], which might expand the number of suggested terms and also change their ranking. In other words, SVD enables us to match terms globally, or conceptually, without the need for explicit connections.

The main goal of this paper is to investigate the use of SVD as a suggestion tool for relational data, establish a better understanding of its behavior, and provide a new method for interactive refinement of search results.

2. Data

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

The "advertiser–search term" relationship can be represented by a bipartite graph with edges connecting advertisers to keywords. The advertisers are on one side of the graph and the keywords are on the other side as depicted in Fig. 1(a). The edges of the graph might also contain bid values. In this representation all correlated terms are connected through the same advertiser, i.e. are next nearest neighbors.

An alternative representation for the data can be given by an "advertiser–search term" matrix, A, Fig. 1(b) whose columns correspond to advertisers and rows to bidded search terms. The number of rows in this matrix, m, is equal to the number of unique bidded search terms, and the number of columns, n, is the number of unique advertisers active on the market. Thus, every column of this matrix represents an advertiser vector described in the bidded terms space and every row is a bidded term vector in the advertiser space.

The matrix A is strictly non-negative and sparse. It is normalized using the binary frequency variant of term-frequency, inverse document frequency normalization [1].

3. Method

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

$$sim(t_i, t_j) = \cos(t_i, t_j) = \frac{t_i^T \cdot t_j}{||t_i||\ ||t_j||}. \tag{1}$$

In a subspace defined by its orthogonal projection $\hat{P}_k$, the similarity (cosine of the angle) between vector projections is

$$sim(\hat{P}_k t_i, \hat{P}_k t_j) = \frac{(\hat{P}_k t_i)^T \cdot (\hat{P}_k t_i)}{||\hat{P}_k t_i||\ ||\hat{P}_k t_j||}. \tag{2}$$

Below we consider an orthogonal subspace constructed by SVD

$$A = USV^T. \tag{3}$$

The first k columns of the matrix V form the truncated orthogonal subspace, V_k. The number of columns in V_k is equal to the rank of the subspace we use, and every column of V_k is a basis vector. The Eckart and Young theorem [5] guarantees that the matrix formed by the top k singular vectors provides the best (closest in the L_2 norm sense) approximation of the data matrix A in any basis of order k.

An orthogonal projection on a subspace spanned by V_k is given by a projection operator $\hat{P}_k = V_k V_k^T$.

Search Terms 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. 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

$$sim(t_i, q_j) = \cos(t_i, q_j) = \frac{(AA^T)_{ij}}{||a_i||\ ||a_j||}, \tag{4}$$

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

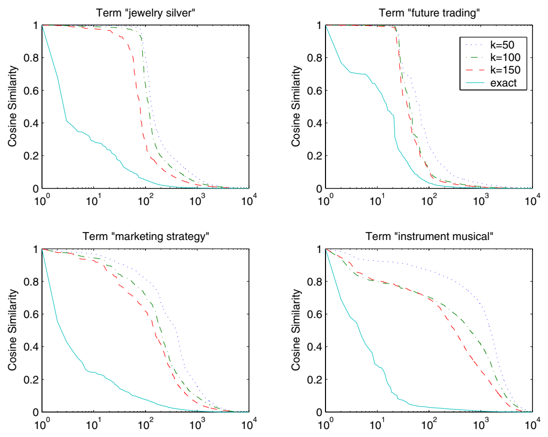


Figure 2. The exact and SVD subspace cosine similarity scores between four search terms and the data.

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

$$sim(\hat{P}_k^T t_i, \hat{P}_k^T q_j) = \frac{(SU_k^T)_i^T (SU_k^T)_j}{||(SU_k^T)_i^T||\ ||(SU_k^T)_j||}. \tag{5}$$

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

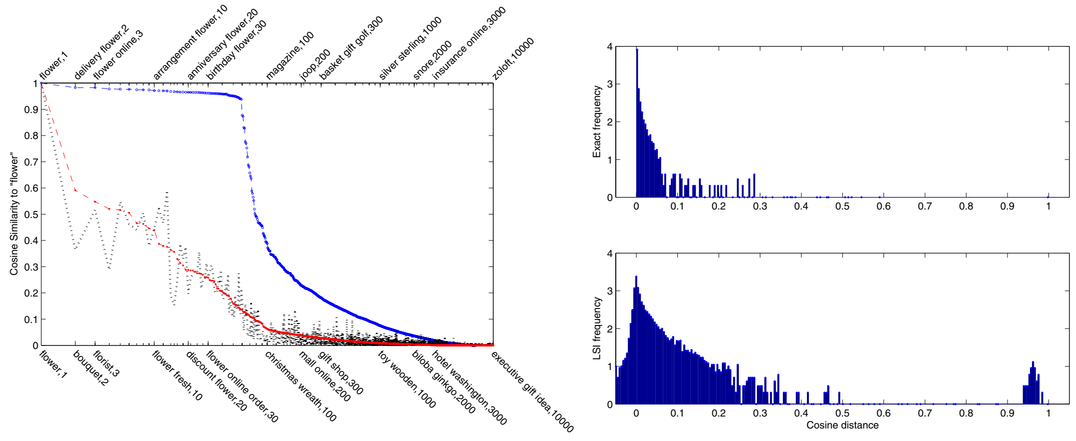
By positive refinement, we mean the user selects positive, reinforcing examples to add to the query from the provided term list. Then, instead of using q_0 as a query we can construct a new, extended query that spans the space $\{q_0, a_{j1}, a_{j2}, ..a_{jp}\}$, where a_j is the j-th column of A^T corresponding to the term that the user chooses to reinforce the query. Notice, that we are not computing the centroid for a new query as in [1], 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 [5].

$$\hat{P}_Q = Q(Q^T Q)^{-1} Q^T. \tag{6}$$

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,

$$sim(t_i, Q) = \frac{t_i^T (\hat{P}_Q t_i)}{||t_i||\ ||\hat{P}_Q t_i||}. \tag{7}$$

This feedback mechanism works in both the entire space and the 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 $a_{ik} = V_k V_k^T a_i$.



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

Figure 3. The suggested results for the term "flower" from the exact cosine method and the SVD method with $k = 100$.

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. In this case, we need to build a complementary projector to the negative examples space, $\hat{P}_{Qn} = I - \hat{P}_Q$.

Then the new similarity score becomes

$$sim(t_i, Q) = \frac{t_i^T (I - \hat{P}_{Qn}^T) t_i}{||t_i||\ ||(I - \hat{P}_{Qn}^T) t_i||}. \tag{8}$$

and search results are ranked according to this similarity.

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 and ARPACK++ libraries [6]. The second is a Java program to query a dataset and retrieve results between general and specific associations.

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, the implementation is solely based on sparse and dense matrix vector multiplications.

5. Results

Figure 2 demonstrates the cosine similarity scores for all terms in the dataset to four different search terms using four methods: exact match and SVD projections into rank $k = 50, 100, 150$ subspaces.

A more detailed example is given in Fig. 3, where we present the suggested results for the term "flower" from the exact cosine method and the SVD method with $k = 100$. In this example, we directly compare the two similarity curves based on their values and the ordering each method generates. The corresponding term suggestions for the query "flower" are shown in Table 1. There is little difference between the top set of results (1-5). The second set of results (74-78) demonstrates the benefits of LSI.

The system is designed for suggestion of search terms to an advertiser which sells flowers. Hence, the suggestion of "cooking" and "cosmetic" are not relevant. However, the suggestion of "stuffed bear" and "gourmet basket" are relevant in that many companies which sell flowers also sell these items. Many other terms suggested for LSI are holidays or events where flowers are frequently given as gifts, i.e. "valentine's day" and "birthday" – good terms for an advertiser selling flowers.

Precision and Recall With relevance judgments, we evaluated precision and recall for the top 10 to 10000 results from the various methods. Fig. 4(a) presents the precision and recall curves from the exact and the SVD method with $k = 100$ for the term "flower." The results show when precision is high, the methods are all roughly equivalent. As precision decreases, the results from SVD are better. Fi-

“flower” (full)			“flower” (SVD $k = 100$)		
1	flower	1	1	flower	1
2	deliver flower	0.591	2	bouquet	0.983
3	flower online	0.548	3	florist	0.983
4	flower send	0.520	4	floral	0.977
5	florist	0.516	5	flower online	0.977
74	sunflower	0.106	74	gourmet basket	0.579
75	birthday	0.106	75	valentine day	0.565
76	cooking	0.106	76	wreath	0.552
77	baby	0.100	77	fruit basket gift	0.519
78	cosmetic	0.098	78	birthday	0.501

Table 1. Comparison of exact and SVD $k = 100$ similarity scores

nally, when the precision is low ($\approx< 0.3$), the methods all show bad results, although SVD appears moderately worse.

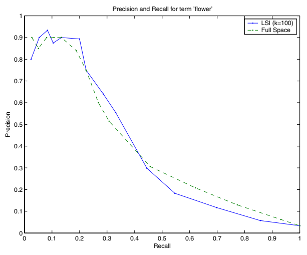
Next, Fig. 4(b) demonstrates the results for the query “internet provider” with positive relevance feedback on “cheap isp” obtained by the subspace projection method. The results after refinement are more relevant to the concept behind the query, i.e. inexpensive internet access

Expanded Coverage By expanded coverage, we refer to the increase in recall at moderate precision (0.3-0.7). The change in coverage happens due to the change in ranking caused by SVD. When projected onto SVD subspace, the angles between term vectors change non-uniformly. This change results in differences in ranking (ordering) of search results for a given search term from the exact match method to the SVD approximate match method. This, in turn, leads to the increase in coverage through SVD subspace projections.

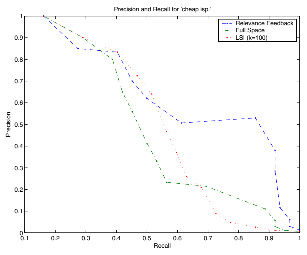
The reordering (change in ranking) caused by SVD is only local. The dotted line in the middle of Fig. 3(a) shows the similarity values from the exact cosine sorting results using the ordering from the SVD results. Of note in this figure is that there is very little reordering of the results beyond the plateau, thus SVD only performs a *local reordering* of the results.

Clustering Behavior Figure 3 also demonstrates the *clustering behavior* that occurs with SVD. The plateau at the top of the SVD curve represents a set of results whose similarity scores are high. If we take the histogram of the distances, as in Fig. 3(b) this behavior becomes even more apparent. The cluster at the right of the SVD histogram represents the “flower” cluster in the data; the steep decline of the SVD curve corresponds to the end of related terms.

The clustering behavior is also due to a non-uniform change in angles in subspace projection. Vectors that are close in the original data get closer when projected into the SVD subspace. Likewise, vectors that were distant in the original data become more distant in the SVD subspace [2].



(a) Precision and Recall for “flower”.



(b) Precision and Recall for ”cheap isp.”

Figure 4. Precision and recall curves

This behavior, however, does not occur for all terms in the dataset. In Fig. 2, the terms “marketing strategy” and “instrument musical” do not display any clear clustering behavior.

Since the steep decline in the SVD 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. For the terms, “marketing strategy” and “instrument musical,” then, there is no good cluster in the data for these terms.

6. Conclusions

We investigated the effect of SVD subspace projections on data from Overture’s advertising market. We developed a tool to suggest related terms at varying levels of generality and a novel relevance feedback system for positive and negative examples using subspaces.

References

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

[2] 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.

[3] 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.

[4] S. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, and Computers*, 23(2), 1991.

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

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