Similar Term Discovery using Web Search

Peter Anick, Vijay Murthi, Shaji Sebastian
Yahoo, Inc.
Santa Clara, CA
E-mail: panick@yahoo-inc.com, murthij@yahoo-inc.com , colleague@shaji.com

Abstract

We present an approach to the discovery of semantically similar terms that utilizes a web search engine as both a source for generating related terms and a tool for estimating the semantic similarity of terms. The system works by associating with each document in the search engine's index a weighted term vector comprising those phrases that best describe the document's subject matter. Related terms for a given seed phrase are generated by running the seed as a search query and mining the result vector produced by averaging the weights of terms associated with the top documents of the query result set. The degree of similarity between the seed term and each related term is then computed as the cosine of the angle between their respective result vectors. We test the effectiveness of this approach for building a term recommender system designed to help online advertisers discover additional phrases to describe their product offering. A comparison of its output with that of several alternative methods finds it to be competitive with the best known alternative.

1. Introduction

Sponsored search, the presentation of ads in response to online search queries, has grown into a multi-billion dollar industry. Most current systems are driven by "bidded terms", words or phrases bid on by advertisers which are intended to trigger relevant ads when they appear in user queries. Since web search queries exhibit a "long tail" – a significant number of low volume queries which in aggregate comprise a large percentage of overall search traffic – it is often difficult for advertisers to generate a set of terms with wide enough scope to cover the great variety of ways users may express their intents. As a result, the development of automatic and semi-automatic tools to assist advertisers in this effort has become an active area of terminological research (e.g., Jones, 2006; Bartz, 2006).

Previous approaches undertaken within the web search community have mined search click and session logs in pursuit of such related terms. It is, for example, reasonable to assume that different queries for which the same url is selected are likely to be semantically related. Similarly, frequently used query reformulations within user search sessions may constitute restatements or refinements of the same intent. One drawback of these approaches is spotty coverage in the search logs; given the "long tail" of low frequency queries issued to search engines, there may not be enough data logged to statistically capture many of the relationships that exist between terms.

In this paper, we present an approach that utilizes web search itself, rather than search logs, to implement an automated term recommender system. Our goal is a system capable of suggesting up to fifty "highly related" terms given a seed set of several short phrases that describe an advertiser's product or service. Suggested terms might include synonyms, hypo and hypernyms, lexical or phrasal variants, as well as other semantically related terms which, if entered as web search queries, would imply a strong interest in the product that is described by the seed set. For example, given the seed set {boots, western wear, cowboy hats}, terms such as *cowboy boots*, *western apparel*, *stetson hats*, and *wrangler jeans* would be considered good queries for matching the advertiser's product offerings.

Our system exploits two methods well known in information retrieval research - the use of the *vector space model* (Salton, 1975) to assess the similarity between queries and documents and the use of *pseudo relevance feedback* (Salton, 1990) to adjust query terms and weights. In the vector space model, both queries and documents are represented as weighted term vectors. Weights are typically computed using a combination of term frequency (the number of times a term appears in a document or query) and inverse document frequency (a measure of the specificity of the term derived from the number of documents in the corpus in which the term occurs). The degree of relatedness between a query and a document in the corpus is computed using the cosine of the angle between their term vectors. Pseudo relevance feedback is a technique for refining search expressions, in which the terms within the top ranked documents for an initial search are used to augment or reweight the initial query terms. As noted by Sahami & Heilman (2006) and Metzler et al (2007), variations on these techniques can be used to compare the similarity in meaning of words or phrases by running them as web queries and then comparing the term vectors produced by the application of pseudo-relevance feedback from web results.

2. Prisma similar terms

Our term recommender system is built using a set of facilities, collectively known as "Prisma", which are integrated into the Yahoo web search engine and which support the run-time generation of query refinements (Anick, 2003).[1] In this section, we describe the components of the Prisma system and follow an example through the sequence of steps that produce a ranked set of similar term suggestions.

2.1 Concept dictionary

Most vector space systems use a uniterm model in which the term vectors are composed of single words. Since we are interested in recommending *phrases*, not just words, we first construct a large "concept dictionary" intended to capture the broad range of topics appearing on the web. The majority of these concepts are either short noun phrases, such as "civil liberties" and "glossy photo paper", or proper names such as persons, places, and products. The dictionary is built in a semi-automatic fashion – drawing candidate terms from concept-rich sources such as query logs, web sites (e.g., wikipedia) and entity name feeds (e.g., movie titles, place names), then filtering out noise terms via automated rules and editorial review. It currently holds over 24 million terms, spanning 5 European languages.

2.2 Document term vectors

The Yahoo search engine constructs an index from documents discovered by crawling the world wide web. The primary function of this index is to map individual words to the set of documents which contain them, but the index also includes metadata for each document, such as its title, url and body text, which are used at query time to create the snippets that summarize the results of a query. For the purposes of Prisma-based applications, we compute one additional metadata field for each document at indexing time – a weighted term vector of up to twenty phrases intended to capture the key concepts represented in the document. This term vector is created by identifying all dictionary terms that occur within the document using a left-to-right greedy phrase matcher and then scoring the terms using a formula that takes into account features such as their position in the document, number of appearances in title, body, or anchor text, and occurrences of sub-phrases (components of longer phrases which also appear as maximal phrases in the same text[2]). Scores for known lexical or inflectional variants are folded into a single entry. The resulting *term importance scores* are used to sort the terms and the top twenty scoring terms are retained. Term weights are then computed by multiplying these scores by each term's inverse document frequency (log(N/df) where N is the number of documents in the corpus, and df is the number of documents which contain the term). Vectors are then normalized to unit vectors.

We refer to the metadata field constructed by this process as the *prisma document vector*. Once a document and its metadata have been added to the index, we can retrieve its prisma document vector via an API call to the index.

2.3 Result vectors

For any natural language query, the Yahoo search engine produces a relevance-ranked list of documents which match the terms of the query. The addition of the prisma document vector as a metadata field within the Yahoo index allows us to utilize the Yahoo search engine to fetch the vectors associated with the top *n* results for any query. We compute a *prisma result vector* for a query by (1) averaging the term weights for each term that appears within the top ranked document vectors, (2) selecting the *m* (e.g., 80) terms with highest average weights, and (3) applying unit vector normalization to the resulting composite vector. For example, for the query "party supplies", the most highly weighted *related terms* in the result vector are:

0.387	party supplies
0.080	birthday party supplies
0.075	party favors
0.055	party decorations
0.043	decorations
0.041	pinatas
0.040	birthday
0.039	birthday party
0.038	invitations
0.038	birthdays
0.030	theme party
0.027	baby shower
0.024	tableware
0.022	party planning
0.021	party games
0.021	favors
0.020	costumes
0.020	discount party supplies
0.019	kids birthday party supplies
0.019	theme party supplies
0.019	prom supplies
0.019	balloons
…	

Among these terms are *instances* of party supplies (piñatas, tableware), *specializations* (birthday party supplies), and *related concepts* (birthday party). For an advertiser interested in matching queries related to "party supplies", many of these terms would be highly relevant.

However, the result vector also includes a number of terms that when taken out of the original query context would be less likely to suggest an interest in "party supplies":

[1] Note that deep integration with an underlying search engine is not essential to our approach but it can substantially reduce the run-time cost of key steps.

[2] For example, the phrase "John Lennon" would receive a boost in its term weight for each occurrence of the sub-phrase "Lennon" that appears elsewhere in the same document as an independent term.

favors
costumes
invitations
flowers
bargain prices
suppliers
crafts

By themselves, these terms are not specific enough. A typical user searching for “crafts” is not likely to be interested in an ad for “party supplies”. Some of these terms would, however, make good bidded terms if they were combined with parts of the original query. For example, “*party* crafts” and “*party* invitations” would be very appropriate queries to match an ad for “party supplies”

2.4 Phrase generation

Since some potentially relevant phrases such as “party crafts” may not be in the dictionary or may not appear in the result vector, it is useful to extend the initial set of candidate phrases with new phrases built by concatenating components. There is a cost to validate these phrases (see discussion section below), so we apply a set of heuristics designed to generate those phrases with the highest likelihood of relevance. Each generated phrase combines one or more components of the seed term with an entire related term.

We must first determine whether the seed term is decomposable and, if so, decide which component(s) are most relevant to the sense of the term as a whole. Those components of the seed term which also appear as components of related terms are taken as candidates. In our example, the component “party” appears (without “supplies”) in modifier (prefix) position in four of the top twenty related terms. This is strong evidence that “party” is a meaningful component suitable for use as a modifier to construct new related phrases.

To choose which of the related terms to combine with “party”, we utilize an offline process that produces a “standalone score” for each term in the dictionary. The standalone score measures the likelihood that when the term appears in a query it comprises the full query rather than a component of a longer query. The score is computed as the ratio of standalone appearances to all appearances in a very large query log derived from over a year of Yahoo’s search traffic. A low ratio (e.g., < .1) is strong evidence that the term typically requires further context to serve as a useful query.

Among the related terms for “party supplies” that have scores < .1 are:

decorations
tableware
candy
invitations
balloons
streamers

For each of these terms we construct a new candidate phrase by concatenating the modifier “party”. The viability of these concatenations as naturally occurring phrases is then tested empirically by running them as quoted (i.e., phrasal) queries on the search engine. Phrases which fail to reach a threshold of web hits are eliminated.

2.5 Similar term ranking

The result set term vector and phrase generation rules together yield a set of candidate similar terms. The final step in the term recommender system is ordering these candidates according to their similarity with the original seed term. This is accomplished by running each candidate term as a query and computing its own result set term vector. Each candidate’s result vector is then compared to that for the seed term using the *cosine similarity measure*:

$$SIM(d_i, d_q) = \frac{\sum_{j=1}^{s} w_{ij} * w_{qj}}{\sqrt{\sum_{j=1}^{s} w_{ij}^2}\sqrt{\sum_{j=1}^{s} w_{qj}^2}}$$

where d_q corresponds to the seed term’s result vector and d_i is the result vector of a candidate related term Weights (w) are the vector term weights as described in section 2.3.

Once the similarities between seeds and candidates are computed, the candidates are reordered to reflect the degree of similarity of their web results.

The results of this step are shown, in part, below.

Cosine	**Candidate term**
0.859850	discount party supplies
0.819140	birthday party supplies
0.759797	party decorations
0.747896	kids birthday party supplies
0.697549	patriotic party supplies
0.604303	party supply store
0.603412	birthday party supply
0.544990	party birthday
0.525055	party novelties
0.480525	party theme
0.438847	birthday themes
0.421452	party items
0.409184	luau supplies
…	

3. Evaluation

Yahoo employs editors to generate and review keyphrase suggestions for advertisers using a standardized set of guidelines for relevance. To evaluate our term recommender output, we randomly selected 100 web "landing pages" from a set of typical advertisers in the U.S. market. An editor then created three independently relevant seed terms to describe the product or service offered on each page. Some examples of seed sets follow:

- african art, african imports, african gifts
- bass fishing, fishing guide, fishing charter
- boots, western wear, cowboy hats
- broker comparison, investment information, stock research

For each seed set, the terms were independently input to our recommender system to produce three sets of similar terms. Scores for similar terms were then averaged over the three seeds and the 50 highest scoring terms for each seed set were presented to a separate set of editors for evaluation of their relevance as descriptors for the original landing page. Editors made a binary relevance decision for each term, providing a measure of precision at ranks 10, 20, 30, 40, and 50. Performance was compared to the output of three other recommender systems (described in Bartz, 2006):

(1) Ninepin
(2) SearchCLickCF
(3) SearchCLickLogistic

Ninepin is a collaborative filtering algorithm that proposes related terms for a seed set using a bipartite graph which relates known advertiser URLs with their bidded terms.

SearchClickCF uses the same collaborative filtering algorithm as Ninepin except that the bipartite graph relates logged user queries (instead of bidded terms) and search click URLs (instead of advertiser bidded URLs).

SearchClickLogistic is a logistic model using lexical features and features from search logs. The model is trained using maximum likelihood to compute the probability of relatedness between seed terms and terms found in search logs.

For this evaluation, our system was configured to use the top 50 web results to produce its result vectors and to submit the top ranked 80 terms from each vector (along with any phrases composed using concatenation heuristics) to the similar term ranking stage described above.

4. Results

Figure 1 shows the precision for the top 10-50 results for each of the systems tested. PrismaSimilarTerms had the highest precision at 50, with 61.8% of the terms rated as acceptable. Statistical significance tests showed no difference between PrismaSimilarTerms and Ninepin but both were significantly better than the SearchClick models.[3]

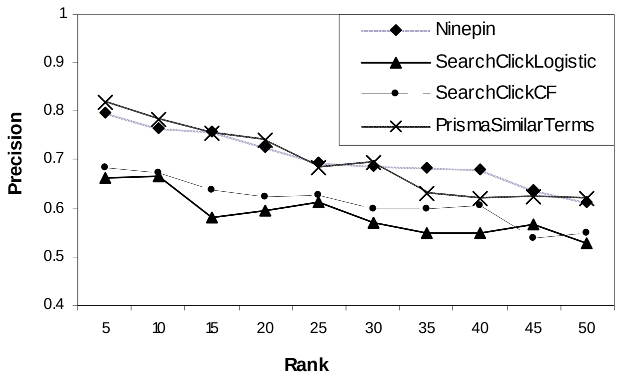

Figure 1: Precision at ranks 10 to 50 for term recommendations produced by four methods.

Table 1 shows the average degree of overlap between terms generated by each model. The high degrees of overlap indicate that the term sets produced by each model are similar on the average but that there is some opportunity for improvement by blending results from multiple models.

	Ninepin	**Prisma**	**SC Log**	**SC CF**
Ninepin	100	89.6	89.2	90.7
Prisma	90.3	100	88.2	89.2
SC Log	89.3	87.9	100	94.6
SC CF	90.8	88.7	94.6	100

Table 1: Overlap matrix, showing percent overlap of terms produced by a given model in the left-hand column with each model tested.

5. Discussion and Conclusions

We have described the construction of a term recommender system that exploits a web search engine to generate and test candidate similar terms. The fact that the system performed on a par with Ninepin is significant, since the Ninepin model draws from a very large pool of existing advertiser related term sets accumulated over many years for the US market. Prisma Similar Terms offers the potential for bootstrapping similar term databases for developing markets in which the pool of existing advertiser term sets is small. However, our model does depend on having a very large concept dictionary of words and phrases for each language.

[3] Since each method was tested on the same seed sets, a pairwise comparison similar to a pairwise t-test was used. We performed the bootstrap (Davison et al, 1997), a non-parametric estimation process, on the seed sets, drawing them with replacement to find 95% confidence intervals for the difference in precisions. No correction was made for multiple comparisons.

Experiments with smaller dictionaries on the order of 1-2 million terms have so far yielded precision scores which are 20-30% lower than that achieved for English, which benefits from a dictionary with over 12 million phrases. A second drawback of our web-based approach is its run-time cost. Web searches, particularly those involving fetches of document vector metadata, are expensive; and the similar term ranking phase of our model requires one web search per comparison made. A possible solution is to pre-compute and cache result vectors for the entire concept dictionary. Then only those generated phrases not in the dictionary would require run-time web searches.

Inspecting the phrases produced by each of the four models tested did not reveal any obvious differences in the nature of the phrases, but given that the outputs tend to differ by about 10%, blending results from different models may be a way to improve overall results. There are several ways to approach this. One approach would be to compute the cosine similarity scores between all pairs of seed set items and the term suggestions made by the other systems. Another approach would be to train a logistic model using features such as the model score and which models contributed to each suggestion, thereby taking into account three sources of information – advertiser bidded data, search click data and web search result set relationships.

Our approach to discovering similar terms via web search has many potential applications beyond the term recommender system described here. Its ability to pull out synonyms and hypo/hypernyms for arbitrary phrases could be exploited for a number of terminological tasks including thesaurus building, paraphrasing, lexical cohesion analysis and web search query expansion. Piggybacking on the search engine's power to find the "most relevant documents" for a given query allows us to hone in on a tiny subset of a huge corpus but its results must be used judiciously. For short or ambiguous queries, web and search engine biases may distort or miss relationships. Depending on the application, this can be a positive or negative feature. Unlike many automated systems which utilize syntactic context (e.g., Grefenstette, 1994; Lin, 1998), lexico-syntactic patterns (Hearst, 1992) or automatic pattern mining (e.g., Nenadić et al, 2002), our method is not sensitive to the immediate sentential context of the terms it compares. As a result, it is less suited for some tasks, such as capturing siblings within a category. On the other hand, it provides a practical way to test for similarity between phrases for which such contextual data is sparse or undiscriminating. Thus, while there are some applications for which it may be usable on its own, there are also opportunities to combine its strengths with the complementary strengths of pattern-based systems. Future work includes further tuning of the algorithm, investigating run-time performance enhancements, and exploring other such applications.

6. References

Anick, P. (2003) Using terminological feedback for web search refinement: a log-based study. In Proceedings of SIGIR 2003, Toronto, Canada

Bartz, K., Murthi, V. and Sebastian, S. (2006) Logistic Regression and Collaborative Filtering for Sponsored Search Term Recommendation, Proceedings of EC'06, Ann Arbor, Michigan.

Davison, A. C. & Hinkley, D. (1997) Bootstrap Methods and their Applications. Cambridge: Cambridge Series in Statistical and Probabilistic Mathematics.

Grefenstette, G. (1994): Explorations in automatic thesaurus discovery. Kluwer Academic Publishers.

Hearst, M. (1992) Automatic acquisition of hyponyms from large text corpora. In Proceedings of 14th Conf. on Computational Linguistics, Vol. 2, Morristown, NJ: Association for Computational Linguistics, pp. 539-545.

Jones, R., Rey, B., Madani, O., and Greiner W. (2006) Generating query substitutions. In Proceedings of WWW 2006, pages 387-396.

Lin, D. (1998) Automatic Retrieval and Clustering of Similar Words. In Proceedings of COLING'98, pp 768-774, Montreal, Canada.

Metzler, D., Dumais, S., and Meek, C. (2007) Similarity Measures for Short Segments of Text. In Proceedings of ECIR 2007, 16-27.

Nenadić, G., Spasić, I. and Ananiadou, S. (2002) Automatic discovery of term similarities using pattern mining. In COMPUTERM 2002: second international workshop on computational terminology, p.1-7.

Sahami, M. and Heilman, T. (2006) A web-based kernel function for measuring the similarity of short text snippets. In Proceedings of WWW 2006, pages 377-386.

Salton, G.. and Buckley, C. (1990) Improving retrieval performance by relevance feedback Journal of the American Society for Information Science, 41:288-297, 1990.

Salton, G., Wong, A. and Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. Communications of the ACM, vol. 18, nr. 11, pages 613–620.