Mining the Web to Determine Similarity Between Words, Objects, and Communities*

Mehran Sahami
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
sahami@google.com

Abstract

The World Wide Web provides a wealth of data that can be harnessed to help improve information retrieval and increase understanding of the relationships between different entities. In many cases, we are often interested in determining how similar two entities may be to each other, where the entities may be pieces of text, descriptions of some object, or even the preferences of a group of people. In this work, we examine several instances of this problem, and show how they can be addressed by harnessing data mining techniques applied to large web-based data sets. Specifically, we examine the problems of: (1) determining the similarity of short texts– even those that may not share any terms in common, (2) learning similarity functions for semi-structured data to address tasks such as record linkage between objects, and (3) measuring the similarity between on-line communities of users as part of a recommendation system. While we present rather different techniques for each problem, we show how measuring similarity between entities in all these domains has a direct application to the overarching goal of improving information access for users of web-based systems.

Introduction

Information finding is generally considered the dominant activity on the World Wide Web. In order to help users find information that is relevant to them, it is often necessary to understand the degree to which various entities on the web are similar or related to each other. For example, in the case of queries to a web search engine, it is often useful to determine how similar one query may be to another in order to potentially suggest related queries to a user or automatically "broaden" the user's query with related terms during retrieval to find more relevant documents. In other contexts, such as E-commerce, users may be looking to find the same item sold at different vendors on the web in order to do comparison shopping. Thus, having a means of automatically determining whether two descriptions of products sold by different vendors are in fact referring to the same true item becomes a critical element in helping the user to compare the right information. And in social networking applications, where users are seeking to find others with similar interests, there is a need to determine how similar different communities of users are to each other in order to make recommendations to users as to what other communities they may find of interest. In all three cases mentioned above, having an appropriate measure of similarity between the relevant domain entities is of paramount concern. Fortunately, the web also provides a rich source of data that may be utilized to address the problem of constructing appropriate similarity measures.

Here we provide an overview of previous research work aimed at the goal of constructing similarity measures for the domains described above. We begin by describing a robust method for measuring the "semantic" similarity between short texts (such as search engine queries) even when the texts being compared may not share any terms in common (Sahami & Heilman 2006). We describe how web search engines are directly harnessed to construct this similarity function.

We then examine the use of machine learning to produce similarity functions between semi-structured data elements (representing objects). We show how by learning such functions we can address the task of adaptive record linkage, so that sufficiently similar items may be linked as referring to the same true underlying object. We also describe an instantiation of this problem in the context of finding similar items for comparison shopping on the web (Bilenko, Basu, & Sahami 2005).

Finally, we describe a large-scale empirical comparison of several similarity measures used as part of a recommendation system in an on-line social network. Specifically, we consider the problem of how to recommend communities (user-created discussion groups) to members of the Orkut social network (www.orkut.com) by determining similarity between a user's current community memberships and other communities that the user is not yet a member of (Spertus, Sahami, & Buyukkokten 2005).

Measuring Similarity of Short Text Snippets

In analyzing text, there are many situations in which we wish to determine how similar two short text snippets are. For example, there may be different ways to describe some

*This paper includes joint work that was done previously in collaboration with Sugato Basu, Mikhail Bilenko, Orkut Buyukkokten, Timothy Heilman, and Ellen Spertus. References to the original works are incorporated in the relevant sections of this paper.

concept or individual, such as "United Nations Secretary-General" and "Kofi Annan", and we would like to determine that there is a high degree of *semantic* similarity between these two text snippets. Similarly, the snippets "AI" and "Artificial Intelligence" are very similar with regard to their meaning, even though they may not share any actual terms in common.

Directly applying traditional document similarity measures, such as the widely used cosine coefficient (Salton, Wong, & Yang 1975; Salton & McGill 1983), to such short text snippets often produces inadequate results, however. Indeed, in both the examples given previously, applying the cosine would yield a similarity of 0 since each given text pair contains no common terms. Even in cases where two snippets may share terms, they may be using the term in different contexts. Consider the snippets "graphical models" and "graphical interface". The first uses *graphical* in reference to graph structures whereas the second uses the term to refer to graphic displays. Thus, while the cosine score between these two snippets would be 0.5 due to the shared lexical term "graphical", at a semantic level the use of this shared term is not truly an indication of similarity between the snippets.

To address this problem, we would like to have a method for measuring the similarity between such short text snippets that captures more of the semantic context of the snippets rather than simply measuring their term-wise similarity. To help us achieve this goal, we can leverage the large volume of documents on the web to determine greater context for a short text snippet. By examining documents that contain the text snippet terms we can discover other contextual terms that help to provide a greater context for the original snippet and potentially resolve ambiguity in the use of terms with multiple meanings.

Our approach to this problem is relatively simple, but surprisingly quite powerful. We simply treat each snippet as a query to a web search engine in order to find a number of documents that contain the terms in the original snippets. We then use these returned documents to create a *context vector* for the original snippet, where such a context vector contains many words that tend to occur in context with the original snippet (i.e., query) terms. Such context vectors can now be much more robustly compared with a measure such as the cosine to determine the similarity between the original text snippets.

Formalizing the Approach

Presently, we formalize our kernel function for semantic similarity. Let x represent a short text snippet[1]. We compute the *query expansion* of x, denoted $QE(x)$, as follows:

1. Issue x as a query to a search engine S.
2. Let $R(x)$ be the set of (at most) n retrieved documents $d_1, d_2, \ldots, d_n$
3. Compute the TFIDF term vector v_i for each document $d_i \in R(x)$
4. Truncate each vector v_i to its m highest weighted terms
5. Let $C(q)$ be the centroid of the L_2 normalized vectors v_i:
$$C(q) = \frac{1}{n}\sum_{i=1}^{n}\frac{v_i}{\|v_i\|_2}$$
6. Let $QE(x)$ be the L_2 normalized centroid of $C(q)$:
$$QE(x) = \frac{C(q)}{\|C(q)\|_2}$$

Note that we consider a TFIDF vector weighting scheme (Salton & Buckley 1988), where the weight $w_{i,j}$ associated with with term t_i in document d_j is defined to be $w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i})$, where $tf_{i,j}$ is the frequency of t_i in d_j, N is the total number of documents in the corpus, and df_i is the total number of documents that contain t_i. Clearly, other weighting schemes are possible, but we choose TFIDF here since it is commonly used in the IR community and we have found it to empirically give good results in building representative query expansions.

Given that we have a means for computing the query expansion for a short text, it is a simple matter to define the semantic kernel function K as the inner product of the query expansions for two text snippets. More formally, given two short text snippets x and y, we define the semantic similarity kernel between them as:

$$K(x,y) = QE(x) \cdot QE(y).$$

We note that $K(x,y)$ is a valid kernel function, since it is defined as an inner product with a bounded norm (given that each query expansion vector has norm 1.0), thus making this similarity function applicable in any kernel-based machine learning algorithm (Cristianini & Shawe-Taylor 2000) where (short) text data is being processed.

Empirical Results

We have tested this similarity measure in a broad range of anecdotal tests and found very good results. We give a few such examples below:

K("UN Secretary-General", "Kofi Annan")	= 0.825
K("Google CEO", "Eric Schmidt")	= 0.845
K("Google Founder", "Larry Page")	= 0.770
K("space exploration", "NASA")	= 0.691
K("Google Founder", "Bill Gates")	= 0.096
K("Web Page", "Larry Page")	= 0.123

Here we see that our kernel function is successful at determining the similarity between pairs of texts, capturing more of their semantic relationship. Also, the function is effective at giving low scores to pairs which semantically differ, such as "Web Page" and "Larry Page".

We have also applied this similarity measure as a key component of a query suggestion system for a web search engine and again seen positive empirical results. We refer the interested reader to the original paper on this approach (Sahami & Heilman 2006) for more information.

Learning Similarity Functions for Record Linkage

Turning our attention to another important problem in measuring similarities, we consider the record linkage task.

[1] We note that while the focus of our work here is short text snippets, there is no technical reason why x must have limited length.

Record linkage is the problem of identifying when two (or more) *references* to an object are describing the same true entity. For example, an instance of record linkage would be identifying if two paper citations (which may be in different styles and formats) refer to the same actual paper. Addressing this problem is important in a number of domains where multiple users, organizations, or authors may describe the same item using varied textual descriptions.

Historically, one of the most examined instances of record linkage is determining if two database records for a person are referring to the same individual, which is an important data cleaning step in applications from direct marketing to survey response (e.g., the US Census). More recently, record linkage has found a number of uses in the context of several web applications, e.g., the above-mentioned task of identifying paper citations that refer to the same publication is an important problem in on-line systems for scholarly paper searches, such as *CiteSeer* and *Google Scholar*.

Record linkage is also a key component of on-line comparison shopping systems. When many different web sites sell the same product, they provide different textual descriptions of the product (which we refer to as "offers"). Thus, a comparison shopping engine is faced with the task of determining which offers are referring to the same true underlying product. Solving this *product normalization* problem allows the comparison shopping engine to display multiple offers for the same product to a user who is trying to determine from which vendor to purchase the product.

In such a context, the number of vendors and sheer number of products (with potentially very different characteristics) make it very difficult to manually craft a single similarity function that can adequately determine if two arbitrary offers are for the same product. Moreover, for different categories of products, different similarity functions may be needed that capture the potentially diverse notions of equivalence for each category. Hence, an approach that allows learning similarity functions between offers from training data becomes necessary.

Furthermore, in many record linkage tasks including product normalization, records to be linked contain multiple fields (e.g., product name, manufacturer, price, etc.). Such records may either come in pre-structured form (e.g., XML or relational database records), or the fields may have been extracted from an underlying textual description (Doorenbos, Etzioni, & Weld 1997). While it may be difficult for a domain expert to specify a complete similarity function between two records, they are often quite capable of defining similarity functions between individual record *fields*. For example, it is relatively simple to define the similarity between two prices as a function related to the inverse of the difference of the prices, or the difference between two textual product descriptions as the (well-known) cosine between their vector-space representations. Thus, an appropriate learnable similarity function for comparing records must be able to leverage multiple *basis* similarity functions that capture individual field similarities.

Fortunately, in the comparison shopping domain a small proportion of the incoming product offers include the Universal Product Code (UPC) attribute which uniquely identifies products. This provides a source of supervision, therefore a classification approach to the linkage problem in such settings becomes feasible.

Formalizing the Approach

Our proposed approach to product normalization is a modular framework that consists of several components: an initial set of basis functions to compute similarity between fields of records to be linked, a learning algorithm for training the parameters of a composite similarity function, and, finally, a clustering-based linkage step.

In formulating our approach, we begin with a set of k basis functions $f_1(R_1,R_2),...,f_k(R_1,R_2)$, defined as similarity functions between fields of records R_1 and R_2. We then learn a linear combination of these basis functions with k corresponding weights α_i and an additional threshold parameter α_0 to create a composite similarity function, f^*:

$$f^*(R_1,R_2) = \alpha_0 + \sum_{i=1}^{k} \alpha_i f_i(R_1,R_2)$$

Once trained, f^* can be used to produce a similarity matrix S over all pairs of records. In turn, S can be used with any similarity-based clustering algorithm to identify clusters, each of which contains a set of records which presumably should be linked. Then, we can interpret each cluster as a set of records referring to the same true underlying item.

Identifying co-referent records requires classifying every candidate pair of records as belonging to the class of co-referent pairs $\mathcal{M}$ or non-equivalent pairs $\mathcal{U}$. Given some domain Δ_R from which each record is sampled, and k similarity functions $f_i : \Delta_R \times \Delta_R \rightarrow \mathbb{R}$ that operate on pairs of records, we can produce a *pair-space vector* $\boldsymbol{x}_i \in \mathbb{R}^{k+1}$ for every pair of records (R_{i_1},R_{i_2}): $\boldsymbol{x}_i = [1, f_1(R_{i_1},R_{i_2}),...,f_k(R_{i_1},R_{i_2})]^T$. The vector includes the k values obtained from basis similarity functions concatenated with a default attribute that always has value 1, which corresponds to the threshold parameter α_0.

Any binary classifier that produces confidence scores can be employed to estimate the overall similarity of a record pair (R_{i_1},R_{i_2}) by classifying the corresponding feature vector $\boldsymbol{x}_i$ and treating classification confidence as similarity. The classifier is typically trained using a corpus of labeled data in the form of pairs of records that are known to be either co-referent ($(R_{i_1},R_{i_2}) \in \mathcal{M}$) or non-equivalent ($(R_{i_1},R_{i_2}) \in \mathcal{U}$). To efficiently train such a classifier (and also to allow for incremental updates as more data becomes available), we employ the *averaged perceptron* algorithm (Collins 2002), a space-efficient variation of the *voted perceptron* algorithm proposed and analyzed by Freund and Schapire (Freund & Schapire 1999).

After we have trained a composite similarity function f^* to correctly predict whether a pair of records is co-referent using labeled data, we can now use it to determine linkages within a set of m unlabeled records. To this end, we use f^* to compute an $m \times m$ similarity matrix $S = \{s_{i,j}\}$ between all pairs of records considered for linkage, where each $s_{i,j} = f^*(R_i,R_j), 1 \leq i,j \leq m$. Armed with this similarity matrix, we face the subsequent problem of identifying groups of co-referent records via clustering.

Empirical Results

In our work, we compare the linkage performance of three variants of the Hierarchical Agglomerative Clustering (HAC) algorithm: single-link, group-average and complete-link (Jain, Murty, & Flynn 1999) on a variety of product offer datasets from the *Froogle* comparison shopping website (`froogle.google.com`). For brevity, here we present an evaluation of our approach on a single representative dataset of 4823 product offers for digital cameras.

The dataset was created by collecting product offers that contain a UPC (Universal Product Code) field, which uniquely identifies commercial products. While less than 10% of all data includes the UPC values, they provide the "golden standard" labels for evaluating linkage accuracy. In our experiments, the UPCs were only used for evaluation purposes, while in an actual fielded product normalization system they can be used as a highly useful attribute for computing overall similarity (although, as discussed below, our results indicate that UPC values alone should not be used as the sole linkage criterion due to the presence of noise).

To evaluate our system, precision and recall defined over pairs of co-referent records were computed after each merge step in the clustering process. *Precision* is the fraction of identified co-referent pairs that are correct, while *recall* is the fraction of co-referent pairs that were identified. While traditional use of precision-recall curves involves interpolating several precision points at standard recall levels to the highest value seen for any lower recall, we have found that such interpolation may grossly misrepresent the dynamics of the clustering process. Therefore, we report non-interpolated (observed) precision values, averaged over 10 trials of 2 folds each.

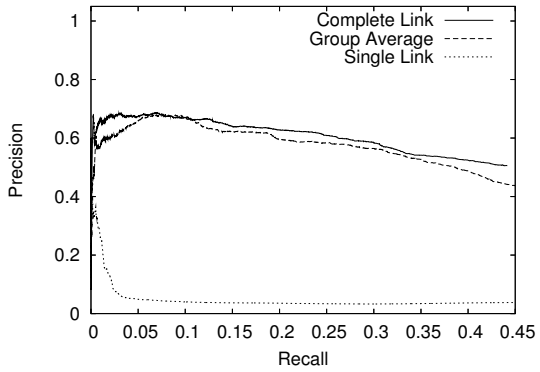

Figure 1: Precision-recall curve for Digital Camera data.

Our empirical results, seen in Figure 1, show that the group-average and complete-link variants of HAC tend to produce much better linkage results than the often used single-link method. Moreover, additional experiments (which have been omitted for brevity) show that great variability exists between the similarity functions that are learned for different product categories. Also, by examining the errors reported by our approach we have found the rather surprising result that (human) errors exist in the labeling of product offers from vendors, as highly similar product offers (which were in fact for the same actual product) were mistakenly given different UPCs by vendors. Our approach to learning the similarity between offers allows us to more readily identify these errors for data cleaning. For more details on this approach, as well as a more comprehensive empirical evaluation, we refer the reader to the original paper on this work (Bilenko, Basu, & Sahami 2005).

Comparing Similarity Functions for Making Recommendations in On-line Communities

In addition to web search and comparison shopping, we have also examined the use of similarity measures in online social networks. Social networking sites such as Orkut (`www.orkut.com`), Friendster (`www.friendster.com`), and others have quickly gained in popularity as a means for letting users with common interests find and communicate with each other. Unfortunately, such social networks, and their corresponding communities (discussion groups), have grown so quickly that new users of the system find it difficult to find the communities that are truly relevant to them. For example, the Orkut social network alone already contains over 1.5 million communities. A new user would find it very difficult to navigate this space of communities manually, and thus automated methods to recommend potentially relevant communities to users become necessary.

While there are many forms of recommender systems (Deshpande & Karypis 2004), we choose a method akin to collaborative filtering (Sarwar *et al.* 2001), where similarity between communities is measured as a function of the communities' overlap in membership (i.e., the number of common members in two communities). How this similarity function is defined has a direct bearing on the quality of the recommendations made to users.

Formalizing the Approach

To define similarity, we start with the notion of a "base" community B, and seek to find its potentially "related" communities R_i. In other words, we seek to measure the (possibly asymmetric) similarity of B to some other community R. Treating a community as simply the set of users in that community, we can use standard set operations (or probabilities defined over sets) to define our similarity measures.

Given a community S, we can define the probability of drawing a user s from the social network that happens to be a member of S as $P(s \in S) = \frac{|S|}{N}$, where N is the total number of users in the social network[2]. Note that we simply assume a uniform distribution with regard to drawing users from the social network.

We consider six standard measures of similarity by which to compare communities. These measures are defined in Table 1. If we represent communities as vectors of length N, where the i^{th} element is 1 if user i is a member of the community and 0 otherwise, we can then consider geometric notions of similarity between these vectors, such as the L_1 and L_2 normed vector overlap. These measures are presented using equivalent set theoretic notation in Table 1. We also con-

[2]For notational convenience, we refer to probabilities such as $P(s \in S)$ as simply $P(s)$ in the rest of the paper.

sider information theoretic measures, such as variations of Pointwise Mutual Information and Log Odds ratios (Cover & Thomas 1991). Finally we consider a measure based on Inverse Document Frequency (IDF) popular in the information retrieval literature (Salton, Wong, & Yang 1975).

We designed an experiment to determine the relative value of the recommendations produced by each similarity measure by interleaving the recommendations from different pairs of similarity measures and tracking user clicks. Specifically, we measured the efficacy of different similarity measures using pair-wise binomial sign tests on click-through data rather than using traditional supervised learning measures such as precision/recall or accuracy since there is no "true" labeled data for this task (i.e., we do not know what are the correct communities that should be recommended to a user). Rather, we focused on the task of determining which of the similarity measures performs best on a relative performance scale with regard to acceptance by users.

When a user viewed a community page, we selected an ordered pair of similarity measures to compare. Let S and T be the ordered lists of recommendations for the two measures, where $S = (s_1, s_2, \ldots, s_{|S|})$ and $T = (t_1, t_2, \ldots, t_{|T|})$ and $|S| = |T|$. The recommendations generated using each measure are combined by Joachims' "Combined Ranking" algorithm (Joachims 2002), which interleaves the top related communities suggested by the two similarity measures (resolving duplicates suggested by both measures), and provides a method for credit assignment to each similarity measure based on which of its results are actually clicked.

Whenever a user visited a community, two measures were chosen and their recommendations interleaved, as discussed above. This was done in a deterministic manner so that a given user always saw the same recommendations for a given community. To minimize feedback effects, we did not regenerate recommendations after the experiment began.

A user who views a base community is either a member (denoted by "m") or non-member (denoted by "n"). In either case, recommendations are shown. When a user clicks on a recommendation, there are three possibilities: (1) the user is already a member of the recommended community ("m"), (2) the user joins the recommended community ("j"), or (3) the user visits but does not join the recommended community ("n"). The combination of base and related community memberships can be combined in six different ways. For example "m→j" denotes a click where a member of the base community clicks on a recommendation to another community to which she does not belong and joins that community. Traditionally, analyses of recommender systems focus on "m→j", or identifying other items a user will like given that they have a stated interest in some initial item. In a similar vein, we judged each similarity measure based on its "m→j" performance.

Empirical Results

We analyzed all accesses from July 1, 2004 to July 18, 2004 of users who joined Orkut during that period. The system served 4,106,050 community pages with recommendations, which provides a lower bound on the number of page views. (Unfortunately, we could not determine the total number of page views due to browser caching.) There were 901,466 clicks on recommendations, 48% by members of the base community, 52% by non-members.

We compared each measure pairwise against every other measure by analyzing clicks of their merged recommendations. If the click was on a recommendation ranked higher by measure L_2 than measure L_1, for example, we considered it a "win" for L_2 and a loss for L_1. If both measures ranked it equally, the result was considered to be a tie. We say that a measure *dominates* another if, in their pairwise comparison, the former has more "wins". In our initial analysis, we found that this definition applied to the click data yielded a total order (to our surprise) among the six measures as follows (listed from best to worst performer): L_2, $MI1$, $MI2$, IDF, L_1, $LogOdds$. Indeed, the L_2 measure outperformed all other measures with statistical significance ($p < 0.01$) using a binomial sign test (Lehmann 1986). The detailed click results comparing the L_2 measure to the other five measures are presented in Table 2, which shows the outcomes of all clicks, with conversions by members ("m→j") and non-members of the base community ("n→j") broken out separately.

As part of the overall performance of the recommendation system, we also defined *conversion rate* as the percentage of non-members who clicked through to a community who then joined it. The conversion rate was three times as high (54%) when the member belonged to the base community (from which the recommendation came) than not (17%). While this relative difference is not surprising, the large absolute magnitude of the conversion rate (i.e., the percentage of people who clicked on a recommendation and ended up joining the recommended community) shows the efficacy of recommendations as a means for providing users with relevant information in a social network setting. More details on this research, as well as the results of several additional experiments, are available in the original paper on this work (Spertus, Sahami, & Buyukkokten 2005).

Conclusions

In this paper we have presented several web-based applications where measuring the similarity between different entities is an important element for success. We have seen that the large quantity of available on-line information can be harnessed to help determine similarities between entities as disparate as words (short texts), objects (such as products offers), and communities (on-line discussion groups). By leveraging this data in conjunction with the development of novel machine learning technologies, we can make significant improvements in a variety of domains, including helping web users find more relevant information while searching the web, engaging in on-line comparison shopping, or participating in a social network. In future work, we seek to find additional applications for the similarity measures defined here and apply the methodology developed for learning similarity functions in other contexts.

References

Bilenko, M.; Basu, S.; and Sahami, M. 2005. Adaptive product normalization: Using online learning for record

Measure Name	**Definition**
L_1 Norm	$L_1(B,R) = \frac{\lvert B \cap R \rvert}{\lvert B \rvert \cdot \lvert R \rvert}$
L_2 Norm	$L_2(B,R) = \frac{\lvert B \cap R \rvert}{\sqrt{\lvert B \rvert \cdot \lvert R \rvert}}$
Pointwise Mutual Information (positive correlations only)	$MI1(B,R) = P(b,r) \cdot \log \frac{P(b,r)}{P(b) \cdot P(r)}$
Pointwise Mutual Information (positive and negative correlations)	$MI2(B,R) = P(b,r) \cdot \log \frac{P(b,r)}{P(b) \cdot P(r)} + P(\overline{b},\overline{r}) \cdot \log \frac{P(\overline{b},\overline{r})}{P(\overline{b}) \cdot P(\overline{r})}$
IDF	$IDF(B,R) = \frac{\lvert B \cap R \rvert}{\lvert B \rvert} \cdot \log \frac{N}{\lvert R \rvert}$
Log Odds	$LogOdds(B,R) = \log \frac{P(r\lvert b)}{P(\overline{r}\lvert b)}$

Table 1: Definition of the six measures used to determine similarity between communities in the Orkut social network.

Measures		m → j			n → j			all clicks		
		Win	Equal	Loss	Win	Equal	Loss	Win	Equal	Loss
L_2	$MI1$	6899	2977	4993	2600	1073	1853	30664	12277	20332
L_2	$MI2$	6940	2743	5008	2636	1078	1872	31134	11260	19832
L_2	IDF	6929	2697	5064	2610	1064	1865	30710	11271	20107
L_2	L_1	7039	2539	4834	2547	941	1983	28506	13081	23998
L_2	$LogOdds$	8186	1638	4442	2852	564	1655	34954	6664	18631

Table 2: The relative performance of L_2 versus the other five measures in pairwise combination on clicks leading to joins, divided by base community membership status, and on all clicks.

linkage in comparison shopping. In *Proc. of the 5th IEEE Int'l Conference on Data Mining*, 58–65.

Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora(EMNLP-2002)*, 1–8.

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory*. New York, NY: Wiley.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Deshpande, M., and Karypis, G. 2004. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems* 22(1):143–177.

Doorenbos, R. B.; Etzioni, O.; and Weld, D. S. 1997. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*, 39–48.

Freund, Y., and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37:277–296.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: A review. *ACM Computing Surveys* 31(3):264–323.

Joachims, T. 2002. Evaluating retrieval performance using clickthrough data. In *Proc. of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*.

Lehmann, E. L. 1986. *Testing Statistical Hypotheses (2nd Edition)*. Springer-Verlag.

Sahami, M., and Heilman, T. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of the 15th Int'l World Wide Web Conference*.

Salton, G., and Buckley, C. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management* 24(5):513–523.

Salton, G., and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.

Salton, G.; Wong, A.; and Yang, C. S. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18:613–620.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th Int'l World Wide Web Conference*, 285–295.

Spertus, E.; Sahami, M.; and Buyukkokten, O. 2005. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proc. of the 11th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, 678–684.