Constructing Career Histories: A Case Study in Disentangling the Threads

Paul R. Cohen
USC Information Sciences Institute
Marina del Rey, California

Abstract

We present an algorithm for organizing partially-ordered observations into multiple "threads," some of which may be concurrent., The algorithm is applied to the problem of constructing career histories for individual scientists from the abstracts of published papers. Because abstracts generally do not provide rich information about the contents of papers, we developed a novel relational method for judging the similarity of papers. We report four experiments that demonstrate the advantage of this method over the traditional Dice and Tanimoto coefficients, and that evaluate the quality of induced multi-thread career histories.

1 Introduction

Like most researchers, I work on several problems at once, so my publications represent several research "threads." Some threads produce several papers, others lie dormant for months or years, and some produce infrequent, irregular publications. Some papers clearly represent new lines of research (for me) while others are continuations. Although my research has changed over the years it is difficult to identify when these changes occurred. Yet, in retrospect, it is usually clear to me that one paper is a direct descendent of another, while a third is unrelated to the other two.

How should we model the processes that produce research and publications? A single researcher can be viewed as a bundle of non-independent processes, each of which produces one kind of research. For example, some of my papers are about modeling cognitive development on robots and some are about finding structure in time series, and although these threads are distinct, they are not independent. Occasionally two or more threads go through one paper, or, to say it differently, one paper can descend from two or more parents. Within a thread, papers may be ordered roughly by their publication dates, although some appear months or years after they were written. Threads sometimes appear with no apparent connection to previous work.

My publications can be arranged in a graph like the one in Figure 1. Later publications are represented as children of earlier ones or, when they are unrelated to earlier publications, as children of the root (e.g., the node labeled d begins a new thread). To avoid false advertising, I must say immediately that the algorithm I present in this paper produces trees in which every publication has only one parent (e.g., it cannot produce nodes like the one labeled a, b). An algorithm for generating graphs like the one in Figure 1 is under development.

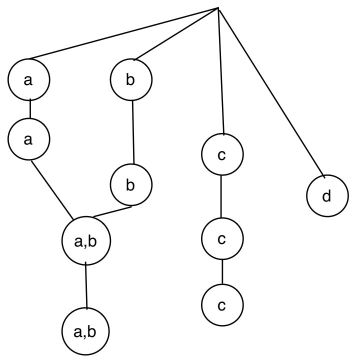

Figure 1: A publication tree. Each node represents a publication. Threads are identified by letters. The depth of a node represents its publication date.

Trees or graphs of publications are not models of the processes that generate publications, but nor are they entirely descriptive. Inference is required to generate them, in particular, research threads and parent-child relationships must be inferred. The inference problem is as follows: Given an ordered or partially-ordered sequence of observations, identify the processes that generated each observation, and assign each observation to at least one process. Some processes may not be active at the beginning of the sequence. I call this the problem of *disentangling the threads*.

2 The PUBTREE Algorithm

The PUBTREE algorithm greedily adds papers to a growing tree of papers:

1. Arrange the papers in chronological order, $p_1, p_2, ...p_n$, least to most recent.
2. Create a root node **r**. Make p_1 a child of **r**.

3. For each remaining paper p_i, find the most similar paper in the tree and make p_i its child. If p_i is not similar enough to any paper in the tree, make it the child of **r**.

The last step is managed with the help of a threshold. For two papers, A and B, if $\mathcal{S}(A, B) > T_{thread}$, treat them as similar enough for B to be A's child, otherwise make B the child of **r**.

2.1 Relational Similarity

Clearly, the PUBTREE algorithm relies on a good similarity metric. This section describes a novel approach to judging the similarity of the abstracts of papers. Two abstracts, A and B, contain sets of words W_A and W_B, of sizes $N_A = \|W_A\|$ and $N_B = \|W_B\|$ respectively. For example, we might have W_A = (*tree search optimize constraint restart*) and W_B = (*graph search vertex constraint*). Note that the abstracts have two common or shared words, *search* and *constraint*, and five words that are not common, or unshared. Shared and unshared words each contribute a component to our measure of similarity of abstracts.

A prevalent way to compare abstracts is to count the number of shared words and divide by the total number of words in both abstracts:

$$D(A, B) = \frac{2 \cdot \|W_A \cap W_B\|}{N_A + N_B} \qquad (1)$$

This is called the Dice coefficient. Its value will be one when $W_A = W_B$ and zero when the abstracts share no words. Several other coefficients have been proposed (e.g., Jaccard, Cosine, see [Grossman and Frieder, 1998]). The differences between them were slight in the experiments that follow.

For unshared words, we define a simple *relational* measure of similarity. Let $occ(w_i, w_j)$ be the number of abstracts in the corpus in which words w_i and w_j co-occur. Two words are said to be *related* if they co-occur in a super-threshold number of abstracts:

$$rel(w_i, w_j) = \begin{cases} 1 & occ(w_i, w_j) > T_{occ} \\ 0 & Otherwise \end{cases} \qquad (2)$$

For two abstracts, A and B, with words W_A and W_B, a relatedness coefficient is

$$R(A, B) = \frac{\sum_{w_i \in W_A} \sum_{w_j \in W_B, w_j \neq w_i} rel(w_i, w_j)}{(N_A N_B) - \|W_A \cup W_B\|} \qquad (3)$$

R is the average value of rel for all pairs of unshared words. Its value will be one when every pair of unshared words is related, that is, occurs in a super-threshold number of abstracts. Note that R does not require two abstracts to share *any* words. That is, the measure works even if $W_A \cap W_B$ is empty.

Putting D and R together we get:

$$\mathcal{S}(A, B) = \alpha D(A, B) + (1 - \alpha) R(A, B) \qquad (4)$$

This measure, a weighted sum of similarity measures on shared and unshared words, also will range from zero to one. If $\alpha = 1$ then $\mathcal{S}(A, B)$ is the Dice coefficient and depends entirely on shared words. If $\alpha = 0$ then $\mathcal{S}(A, B)$ is the relational similarity measure, R, and depends entirely on unshared words.

To illustrate the value of the relational similarity measure, R, consider two abstracts of papers about the same piece of research, a system for automating exploratory data analysis. After non-content words are removed, the first abstract is represented by: *automate, capture, complexities, contribute, exploratory, EDA, IGOR, knowledge-based, manage, Phoenix, relies* and *script-based*. The second abstract, by the same authors, represents another paper about a different aspect of the system, with these words: *Aide, assistant, ARPAROME, begins, captured, contracts, data-driven, descriptive, extracted, examined, enormous, explored, EDA, exploratory, informal, presentation, plays, prelude, researcher, subtle, understanding, word.* The Dice coefficient for these abstracts is just $D = 0.11$ because, although the abstracts are from very similar papers, they share only the words. *exploratory* and *EDA*. The relational coefficient is $R = 0.49$, a much more satisfactory representation of the similarity of these papers.

3 Related work

PUBTREE is neither a clustering algorithm nor a sequence analysis algorithm but has some attributes of both. Like clustering, PUBTREE forms groups of objects, that is, it groups papers together into threads. Yet clusters are generally unstructured, or hierarchically structured, while PUBTREE's threads are organized sequentially by publication dates. Another difference is that, generally, an item may be added to a cluster if it is similar to *all* items in the cluster (usually via comparison to the cluster centroid) whereas an item may be added to a thread if it is similar to the last item on the thread.

Sometimes, adding an item to a thread causes the thread to split into two or more threads (i.e., threads are organized into trees). A thread $A, B, C...$ splits under node B when a paper D is considered more similar to B than to C. Recall, a thread is supposed to represent a process that produces research papers, and a researcher is thought of as a bundle of such processes, so splitting a thread is analogous to changepoint analysis (e.g., [Lai, 1995]), or finding a point in a sequence where a process changes. One can imagine a kind of PUBTREE algorithm that repeatedly samples sequences from the author's list of papers, builds Markov chain models of some kind, and identifies threads as sequences that *do not* feature changepoints. The challenge, of course, is to model the production of papers as a stochastic process. We do not know of any way to do this, which is why we built the similarity-based PUBTREE algorithm. It would be nice to model the author as a Hidden Markov Model (HMM), where the hidden state is the thread identifier, but, again, we do not have a stochastic framework for the production of research papers. Nor do we know a priori how many threads (i.e., hidden states) an author will produce, as fitting HMMs requires.

Genter and her colleagues introduced the term *relational similarity* to emphasize that situations may be similar by merit of the relationships that hold among their elements, even though the elements themselves are different [Gentner and Forbus, 1995; Gentner and Markman, 1997; Goldstone *et* *al.*, 1997]. This notion of similarity requires situations to be structured; there are no relationships between elements in a bag. However, as this paper shows, elements w_i and w_j from unstructured bags of words W_A and W_B, respectively, can be related by their appearances in other bags $W_C, W_D, \ldots$. Clearly, this is a weak kind of relational similarity, but the term seems apt, nonetheless.

Our kind of relational similarity is a bit like query expansion in information retrieval. The idea of query expansion is that words in queries are rarely sufficient to find the right documents, so the query is "expanded" or augmented with additional words. Local query expansion [Xu and Croft, 1996] uses the query to retrieve some documents and the words in those documents to expand the query. So, in query expansion, $w_i \in W_A$ causes $W_C, W_D, \ldots$ to be retrieved, and $w_j \in W_C, w_k \in W_D, \ldots$ expand the query, which might cause W_B to be retrieved. Whereas, in our technique W_A and W_B are judged to be similar if $w_i \in W_A$ and $w_j \in W_B$ and $w_i, w_j \in W_C; w_i, w_j \in W_D, \ldots$. Clearly these aren't identical inferences, but they are related.

4 Experiments

The experiments reported here were performed on a corpus of 11672 papers from the Citeseer database. The papers were selected by starting with my own Citeseer entries, adding all those of my coauthors, and all those of their coauthors. Each paper's abstract was processed to remove common and duplicate words so each paper was represented by a set of words. (Unlike in most information retrieval applications, no special effort was taken to identify discriminating words, such as ranking words by their tf/idf scores.) The mean, median and standard deviations of the number of words in an abstract were 21.25, 21, and 10.6, respectively.

4.1 Experiment 1: Same/different author

Our first experiment tested the relative contributions of the D and R coefficients to performance on a classification task. The task was to decide whether two papers were written by the same author given their abstracts. The procedure is: Select abstract A at random from the corpus; with probability p select another abstract, B, from the papers of the same author as abstract A, and with probability $1-p$ select B at random from the corpus; record whether A and B have the same author; record $D(A,B)$ and $R(A,B)$. (For this experiment, $T_{occ} = 5$.) Now, for a given value of α, compute the similarity score $\mathcal{S}(A,B)$ and compare it to a threshold, $T_{\mathcal{S}}$, and if $\mathcal{S}(A,B) > T_{\mathcal{S}}$ decide that A and B have the same author. Because we recorded whether the papers were truly written by one author or two, we know whether a "single-author" decision is a correct (a hit) or a incorrect (a false positive). Repeating this process for many pairs of papers (1000 in these experiments) yields a hit rate and a false positive rate for each value of $T_{\mathcal{S}}$, and by varying $T_{\mathcal{S}}$ we get a ROC curve.

The ROC curve represents how good a classifier is. Ideally, the curve should have a vertical line going from the origin to the top-left corner. Such a curve would indicate that the false-positive rate is zero for all levels of hit rate up to 1.0. More commonly, as $T_{\mathcal{S}}$ decreases, one gets more hits but also more false positives. Figure 2 shows curves for three levels of α, that is, for three degrees of mixing of the Dice and relational similarity measurements. The curve labeled R is for relational similarity alone ($\alpha = 0$) and the curve labeled D is for the Dice coefficient alone ($\alpha = 1$). Between them lies a curve for $\alpha = .5$.

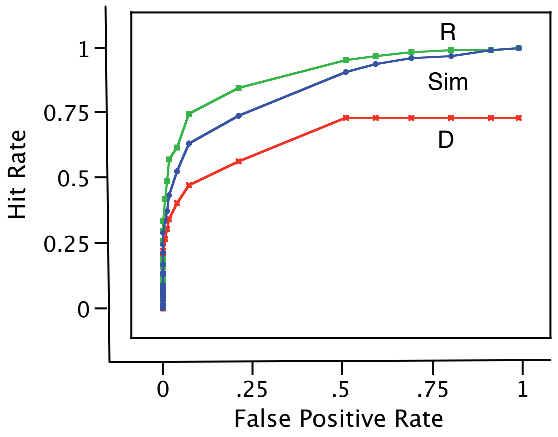

Figure 2: ROC curve for classifying whether two papers are written by one author or two, based on the decision criterion $\mathcal{S}(A,B) > T_{\mathcal{S}}$. Points are for different values of $T_{\mathcal{S}}$.

The relational similarity measure R has a slightly better ROC curve than the others because it is a more nuanced metric than the Dice coefficient, D. In fact, D has a fairly high accuracy; for example, if one says two abstracts are written by different authors when $D = 0$ and by the same author when $D > 0$, then one's overall accuracy (true negatives plus true positives) is 79%. However, roughly 25% of the pairs of abstracts that are written by one author have $D = 0$, and it is here that the R coefficient provides additional resolution.

A followup experiment compared not individual papers but the entire oeuvres of authors. The protocol was as follows: Select two authors A and B at random from the corpus. Select random pairs a, b in which a and b are papers of A's and B's, respectively. Calculate $\mathcal{S}(a,b)$ for each such pair and add the mean of these scores (i.e., the mean pairwise similarity of papers by authors A and B) to the distribution of *between-author* scores. Select random pairs of papers a_i, a_j and b_i, b_j, calculate their similarity scores and add the mean similarity scores to the distribution of *within-author* scores. Finally, repeatedly select a mean score from either the within-author or between-author distribution and, if the score exceeds a threshold T assert that it is a within-author mean score (as these are expected to be higher than between-author scores). Calculate the classification accuracy of these assertions for different values of T to get an ROC curve. The result is that the area under the ROC curve is 0.955, which is nearly perfect classification. In other words, the mean similarity scores for pairs of papers by the same author are very much higher, in general, than the mean similarity scores of pairs of papers by different authors.

4.2 Experiment 2: Author trees and intrusions

The second experiment tests how the similarity measure in Equation 4 performs in the PUBTREE algorithm. It is difficult to get a gold standard for performance for PUBTREE. The algorithm is intended to organize the work of a research into "threads," but we lack an objective way to assess the threads that characterize researchers' work (see the following section for a subjective assessment). However, if we build a graph from *two* researchers' abstracts, then we would hope the PUBTREE algorithm would not mix up the abstracts. Each thread it discovers should belong to one author and should include no abstracts by the other. This is our proxy for PUBTREE's ability to disentangle research threads, and it can be evaluated objectively.

The procedure for Experiment 2 is: Select ten abstracts at random from the work of a random author. Select another ten abstracts from a second author. Run PUBTREE to produce a tree for the first author. Now, have the algorithm add the papers of the second author to this tree. To score performance, recall that every abstract can be added as the child of one already in the tree, extending a thread; or it can be added as a child of the root, starting a new thread. Good performance has two aspects: there should be no "intrusions" of the second author into threads of the first author, and the number of new threads should be small. If the number of new threads is large, then PUBTREE is saying, essentially, "every abstract looks different to me, I can't find any threads." Thus, each replicate of the experiment returns two numbers: The proportion of all the abstracts, from both authors, that join extant threads; and the proportion of the second author's abstracts that intrude on the threads of the first author. Fifty replicates were run at each of ten levels of the threshold T_{thread}, which is the threshold for allowing an abstract to extend a thread. The entire experiment was repeated three times for different mixing proportions of D and R, namely, $\alpha = [1.0, 0.5, 0]$

The results of this experiment are shown in Figure 3. The axes are the levels of the two dependent measures described in the previous paragraph. On the horizontal axis is the proportion of all abstracts that join extant threads, and on the vertical axis is the proportion of abstracts of the second author that intrude into threads of the first author. Each point on the graph represents the mean of fifty replicates associated with one level of α and one level of T_{thread}. In general, higher values of T_{thread} cause new threads to be formed because if an abstract is not similar enough to an abstract in an extant thread to exceed T_{thread}, then it starts a new thread. If an abstract begins a new thread then it cannot intrude in an extant one, so there is a positive relationship between the tendency to join an extant thread and the number of intrusions.

Although the relationship between these variables is not linear, we can interpret the slopes of the lines in Figure 3 very roughly as the error rate, in terms of intrusions, of joining extant threads. This error rate is highest for the relational similarity measure, R, and appears to be lowest for the Dice coefficient D. In fact, almost all of the data for $\alpha = 1$ — which is equivalent to the Dice coefficient — is grouped near the origin because $D = 0$ so often, as we noted in the previous section. Said differently, when the similarity metric is D, intrusions are avoided by starting a lot of new threads. In contrast, when the similarity metric is R, the relational similarity coefficient, the error rate associated with joining extant threads is highest. An intermediate error rate is evident for $\alpha = .5$.

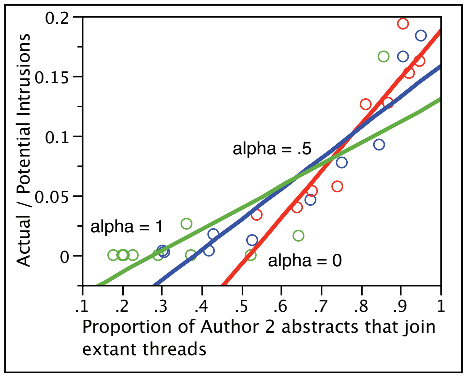

Figure 3: The relationship between the number of threads and the proportion of opportunities to include intrusions that were taken, for three levels of α and ten levels of T_{thread}

Experiment 2 shows that the PUBTREE algorithm does a good job in most conditions of keeping the abstracts of one author from intruding on the threads of another. This result provides some evidence that PUBTREE can separate abstracts into threads and keep the threads relatively "pure." Of course, this experiment actually tested whether PUBTREE would confuse the abstracts of two different authors, it did not test whether PUBTREE correctly finds the threads in the papers of one author.

5 Experiment 3: One author with coauthor indicator

The third experiment uses coauthor information as a gold standard for the performance of PUBTREE. The idea is that papers within a thread often have the same coauthors, whereas papers in different threads are less likely to. We would have reason to doubt whether PUBTREE finds real threads if the degree of overlap between coauthors within a thread is no different than the overlap between coauthors across threads.

The procedure for Experiment 3 is replicated many times, once for each author of ten or more abstracts. For each such author, build a publication tree with PUBTREE. Then for every pair of abstracts in the tree, calculate the overlap between coauthor lists with our old friend, the Dice coefficient (Eq. 1), substituting lists of authors for W_A and W_B. Next, the overlap between papers *within* each thread is calculated in the same way. Thus we generate two lists of overlap scores, one for all pairs of papers, the other for papers in threads. Finally, a two-sample t test is run on these lists and the *t* statistic is reported. (Note we are not using the t test for any inferential purpose, merely using the t statistic as a descriptive measure of the difference between two samples.)

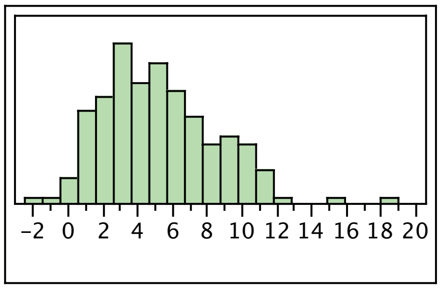

Figure 4: The distribution of t statistics for 165 authors of at least 10 papers. A publication tree is built for each author and the degree of overlap between coauthors of pairs of papers within and between threads is calculated. The between-thread and within-thread samples are compared with a two-sample t test and this is the distribution of t statistics.

Virtually all the authors showed the same pattern: Coauthors of papers on a thread overlapped more than coauthors of papers on different threads. The distribution of t statistics is shown in Figure 4. Values greater than zero indicate more within-thread coauthor overlap, values less than zero indicate more between-thread overlap. Values greater than 2 would be significant at the .01 level if we were testing the hypothesis that within-thread overlap is no different than between-thread overlap. Roughly 2% of the authors had values less than zero and 81% had values greater than 2.0. On average, the overlap coefficients for papers within and between threads were .73 and .48, respectively. Said differently, papers within threads had 64% more overlap among coauthors (calculated per primary author and then averaged) than papers between threads. We may safely conclude that PUBTREE's threads, *which it constructs with no knowledge of coauthors*, satisfy one intuitive feature of threads, namely, that papers on threads often share coauthors.

6 Experiment 4: Subjective judgments

The first three experiments report proxies for the subjective notion of a research thread. Experiment 1 showed that the similarity measure S(A,B) does pretty well as a classifier of whether abstracts A and B are written by the same author. Experiment 2 showed that PUBTREE tends to segregate papers written by different authors into different threads of a publication tree. Experiment 3 used exogenous information about coauthors to show that abstracts in PUBTREE's threads tend to share coauthors. None of these is a direct test of whether PUBTREE builds trees that represent threads in authors' research. Research threads are subjective entities and the accuracy with which they are recovered by PUBTREE cannot be assessed objectively with the information available in Citeseer. The best we can do is ask authors to assess subjectively the publication trees constructed for them by PUBTREE. Three authors volunteered for this arduous job. Each was instructed to examine every thread of his or her publication tree and count the number of inappropriate parent/child relationships. For example, consider a thread of three papers, A, B and C. Two papers, B and C might be misplaced: B might not be a child of A and C might not be a child of B. The latter question is settled independently of the former, so even if B is not appropriately placed under A, C might be appropriately placed under B. One ought to ask whether C is appropriately placed under A, but this question made the task of scoring publication trees considerably harder for the volunteers and was not asked.

The trees for the three authors contained 44, 71, and 74 papers, respectively. The authors all have written considerably more papers, so PUBTREE was working with incomplete data. The first author reported that 32 of her papers, or 73% were placed correctly. Of the remaining 12 papers, three should have started entirely new threads (i.e., should have been children of the root node), and two had incorrect Citeseer dates and might have been placed wrongly for this reason. The second author reports a slightly higher proportion of misplaced papers (24, or 34%) but notes that for more than half of these, the publication date was unknown and so was not used by the algorithm. The third author reported that 14 of the 74 papers had corrupted titles and were duplicates of others in the set. Of the remaining 60 papers, 10, or 17% were placed wrongly. Three started new threads but should have been added to extant threads, four had valid dates and were placed under the wrong parents, and three had no dates and were placed under the wrong parents.

These results are not dramatically good, but nor are they dramatically bad, especially when one considers that every paper might have been placed below any of two dozen (for the first author) or three dozen (for the second and third author) papers, on average. One of the authors said, "[The] tree did surprisingly well, given the subtle differences between papers. ...The tree made most but not all of the right connections. I noticed that some papers could be thought of as having more than one parent in reality, which would complicate threads, but of course that can't be addressed in a single tree."

7 Discussion and Future Work

As the author noted correctly in the previous section, PUBTREE builds trees, not graphs, and so it cannot give a paper more than one parent. This is unfortunate because some papers truly are the children of two or more research threads. A version of the algorithm that produces graphs is under development. Another likely way to improve the algorithm is to include additional information about papers, such as coauthor lists.

PUBTREE and its successors might be applied to several current problems. One is to track the development of ideas in the Blogosphere. Blogs are like abstracts of research papers in many ways and it will be interesting to look for blog threads in the writings of individual authors, and perhaps even more enlightening to find threads in amalgamations of several authors' blogs. Email threading is a related problem.

The relational similarity measure that underlies PUBTREE also has numerous applications, notably to problems that require some degree of semantic matching between documents that do not necessarily share terms. One example is a kind of entity resolution for authors. The Citeseer corpus we worked with contains abstracts for Paul R. Cohen and a nearly disjoint set of abstracts for Paul Cohen. We might wonder whether these authors are the same person. One approach is to calculate the average between-author similarity for these authors and compare it to a null hypothesis distribution of between-author similarities for pairs of authors who are known to be different. The between-author similarity for the two authors is 0.257, a number which is expected by chance fewer than six times in a thousand, so it is very likely that the two are the same person.

In conclusion, the contributions of this paper are three: We identified a class of problems that we call "disentangling the threads," in which several processes generate a single ordered or partially-ordered sequence of observations and the challenge is to assign each observation to its generating process. We developed a simple, greedy algorithm for solving the problem. And we introduced a new relational similarity metric that seems to work better than the Dice coefficient and other common metrics that rely on overlapping sets of items. We presented four experiments that suggest the metric and the PUBTREE algorithm do a credible job of disentangling the threads in the publication records of researchers. Many improvements to the algorithm are possible, and yet, even the simple version presented here performs very well.

8 Acknowledgments

Thanks to Wesley Kerr for help with the Citeseer corpus and to Wesley Kerr, Sinjini Mitra and Aram Galstyan for preliminary work on the problem of constructing career histories. Thanks to Padhraic Smyth for providing access to his topic model software. Special thanks to the three authors who painstakingly analyzed their own publication trees: Prof. Adele Howe, Prof. Tim Oates and Prof. Robert St. Amant. This work was supported by contract #2003*S518700*00 from IBM.

References

[Gentner and Forbus, 1995] Dedre Gentner and Kenneth D. Forbus. Mac/fac: A model of similarity-based retrieval, 1995.

[Gentner and Markman, 1997] D. Gentner and A. M. Markman. Structure mapping in analogy and similarity. *American Psychologist*, 52(45–56), 1997.

[Goldstone *et al.*, 1997] R. Goldstone, D. Medin, and J. Halberstadt. Similarity in context, 1997.

[Grossman and Frieder, 1998] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics*. Kluwer, 1998.

[Lai, 1995] Tze Leung Lai. Sequential changepoint detection in quality control and dynamical systems. *AI Journal of the Royal Statistical Society. Series B (Methodological)*, 57(5):613–658, 1995.

[Xu and Croft, 1996] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.