Who Should I Cite?
Learning Literature Search Models from Citation Behavior

Steven Bethard
Stanford University
353 Serra Mall
Stanford, CA, USA
bethard@stanford.edu

Dan Jurafsky
Stanford University
450 Serra Mall
Stanford, CA, USA
jurafsky@stanford.edu

ABSTRACT

Scientists depend on literature search to find prior work that is relevant to their research ideas. We introduce a retrieval model for literature search that incorporates a wide variety of factors important to researchers, and learns the weights of each of these factors by observing citation patterns. We introduce features like topical similarity and author behavioral patterns, and combine these with features from related work like citation count and recency of publication. We present an iterative process for learning weights for these features that alternates between retrieving articles with the current retrieval model, and updating model weights by training a supervised classifier on these articles. We propose a new task for evaluating the resulting retrieval models, where the retrieval system takes only an abstract as its input and must produce as output the list of references at the end of the abstract's article. We evaluate our model on a collection of journal, conference and workshop articles from the ACL Anthology Reference Corpus. Our model achieves a mean average precision of 28.7, a 12.8 point improvement over a term similarity baseline, and a significant improvement both over models using only features from related work and over models without our iterative learning.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.5.4 [**Pattern Recognition**]: Applications—*Text Processing*

General Terms

Algorithms

Keywords

literature search, retrieval models, citation patterns, author behavior, topic models

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.
Copyright 2010 ACM 978-1-4503-0099-5/10/10 ...$10.00.

1. INTRODUCTION

Effective scientific research requires keeping up with a large, ever-growing body of literature. A researcher that comes up with a seemingly novel project idea that is outside their area of expertise will often find it difficult to determine whether that research has already been done, and what prior research in the field is most relevant. The advent of search engines has made this task somewhat easier, but scientific article search is often still a complex process that alternates between guessing "good" keywords for a search engine, and looking at articles citing and cited by the retrieved articles.

One of the reasons for this iterative process is that traditional information retrieval relies primarily on keywords for matching queries to documents. Search engines may include some analysis of links between webpages, but this is less effective for scientific articles which are often in link-less formats like PDF. Moreover, the relevance of a scientific article to a particular project idea often cannot be characterized by keyword matching alone, and must consider other factors like citation patterns, co-authorship networks and subject area matching.

We believe that the complexity of scientific literature search can be reduced by building a retrieval model that integrates the features of classic keyword search with features important to scientific literature retrieval, such as the citation network, the recency of publication, the topical similarity of articles, and some understanding of the role social networks play. We introduce a number of such features, and show how they can be integrated in a simple retrieval model whose weights can be learned effectively using an iterative training algorithm.

Ideally, such a system would be trained on a set of short project descriptions and lists of articles relevant to each. Because no such data is readily available, in this article we use the following proxy: we treat the abstract of an article as the project idea, and the list of references at the end as the relevant articles. We then build a retrieval system that takes abstracts as inputs and produces reference lists as output. Using this proxy makes the task somewhat harder – because researchers generally read more than just what is in the references list, and thus some relevant articles will be missing – but we believe it is a good approximation, and it is easy to collect such data from existing citation databases.

The rest of this article explains our approach to citation retrieval. We first describe related work, and explain our formulation of the task. Then we introduce the new features we have developed, and our algorithm for training a retrieval model. The model is then evaluated on the ACL Anthology, and we analyze the effects of classifier choices, the effectiveness of multiple training iterations, the importance of different features, and some sociological implications. Finally, we discuss some avenues for future research.

2. RELATED WORK

Early research found that augmenting keyword search with some knowledge of citations had potential for improving literature search. Pao [21] ran a case study where medical professionals gave a description of a topic of interest and an example article, and librarians searched using both keywords and citations. Pao found that searching by citations added an extra 24% relevant articles not found by keyword search. Larsen [9] later tried to automate something like this process, retrieving articles by keywords and then following their citations to retrieve additional articles. However, on queries over IEEE Computer Society journals, an implementation of this approach by Larsen and Ingwersen performed worse than a traditional term based retrieval system [10].

One of the difficulties with the Larsen and Ingwersen approach was that it treated keyword and citation search as independent processes. Models that integrate the two have seen some gains in retrieval quality. For example, Meij and de Rijke [15] worked with queries generated by biologists against the MEDLINE database, and found that boosting documents based on the number of times they were cited provided absolute gains of up to 2.5 percentage points in mean average precision. Similarly, in a patent retrieval task where a patent application was the query and the expected results were the reason for that application's rejection, Fujii [6] found an absolute gain of 0.4 percentage points in mean average precision by boosting documents based on their PageRank score [20].

Another way researchers have integrated citation and keyword information is through *citing snippets*, the bits of text near where an article is cited. Bradshaw [3] found that on Citeseer articles that were cited at least once, a model that indexed only the citing snippets for each article outperformed a model that indexed the article texts. Similarly, using author generated research questions as queries against the ACL Anthology, Ritchie et al. [24] found that concatenating the citing snippets with the article text led to absolute gains of up to 6.8 percentage points in mean average precision. Ritchie also showed that exact snippet boundaries were unimportant – even just the 100 words around the citation was effective.

Recent work has looked at feature-based models where keyword, citation, and other kinds of information can be assigned different weights and integrated into a single scoring function. Tbahriti et al. [28] developed a classifier for segmenting MEDLINE abstracts into purpose, methods, results and conclusion sections, and built a model with a feature for each section. In a task where the title, abstract and MeSH terms of an article were used as the query and the reference list was the expected result, manually boosting the purpose and conclusion features led to an absolute improvement of 0.8 percentage points in mean average precision.

Strohman et al. [26] explored additional features in a task where an entire document was the query and the document reference lists were the expected results. They first employed one iteration of a Larsen and Ingwersen-like system, retrieving 100 results by terms alone, and then adding in all papers they cited. Then they re-ranked all of the retrieved articles using features like the year in which they were published, the citation count, authors in common with the query and a measure of distance through the citation graph. Weights for these features were learned using coordinate ascent, and their model achieved an absolute gain of 9.4 percentage points in mean average precision over a simple term-based baseline.

Our work improves upon this prior work in several ways. We first propose a more realistic task to represent a researcher investigating a new project idea, where only the title and abstract of the paper are given, rather than assuming the author has already written the entire article or assigned terms from a controlled vocabulary. Next, we introduce a wide variety of new features for citation retrieval, drawing from research in both bibliometrics and statistical topic modeling, to construct useful features that were previously unexplored for this task. Finally, we construct a new model for combining the various features, and demonstrate that an iterative paradigm for learning model weights outperforms the single-iteration training typical of prior work.

3. FEATURES

At its core, our retrieval model scores each document (article) d against the query (project idea) q using a weighted sum of feature scores:

$$\text{score}(q, d) = \sum_i w_i \times f_i(q, d)$$

The features represent facts about the article that might make it worth citing. For example, in addition to finding articles with similar terms, it would also be helpful to know if the article was frequently cited, if it is old or new, or if it pertains to similar topics or subject areas. It would also be good to understand the social reasons that people cite – articles from more prestigious authors or venues might be preferred over others. The following sections describe the scoring functions we introduced to characterize such features.

3.1 Similar Terms

Despite our complaints about the inadequacy of current search engines for literature search, there is no doubt that looking for similar terms is a great place to start. Thus, we include the classic TF-IDF term scoring from information retrieval, which represents both the query and the document as a vector of word counts [13, pages 107-116]. We use the Lucene 3.0.0 default implementation of TF-IDF, whose full calculation covers several pages of Java documentation[1], but at its core looks basically like:

$$\text{tf}(t, d) = |\{t' \in \text{terms}(d) : t' = t\}|$$

$$\text{idf}(d) = 1 + \log \frac{|D|}{|\{d \in D : t \in \text{terms}(d)\}| + 1}$$

$$\text{score}_{\text{terms}}(q, d) = \sum_{t \in q} \text{tf}(t, d)^{.5} \cdot \text{idf}(d)^2 \ldots$$

This score increases when terms (t) are shared between the query (q) and the document (d), but terms that appear in many documents in the collection (D), such as *the*, are heavily discounted. The Lucene additions to this calculation do other useful things like normalizing for query and document length, making query scores comparable, etc.

[1] `http://lucene.apache.org/java/3_0_0/api/all/org/apache/lucene/search/Similarity.html`

3.2 Cited by Others

Most search engines incorporate site popularity by boosting webpages that are linked to by many other pages. We introduce features that encode scientific article popularity by inspecting the network of article citations. The first feature boosts articles by their *citation count*, i.e. the number of times the article was cited:

$$\text{citing}(d) = \{d' \in D : d' \text{cites } d\}$$

$$\text{score}_{\text{citation-count}}(q, d) = |\text{citing}(d)|$$

The second feature boosts articles by their PageRank [20] score, but calculated over the citation network instead of the hyperlink network (similar in spirit to what was suggested by Pinski and Narin [22]):

$$\text{score}_{\text{pagerank}}(q, d) = \frac{1-x}{|D|} + x \sum_{d' \in \text{citing}(d)} \frac{\text{score}_{\text{pagerank}}(q, d')}{|\{d'' \in D : d' \text{cites } d''\}|}$$

The intuition behind PageRank is that articles cited by many articles who are in turn cited by many articles, are good candidates to propose to a user. Thus, PageRank is a recursive scoring function that calculates the popularity of an article by summing the PageRank scores of all articles that cite the original one, and normalizing by the number of items each such article cites. The damping factor (x in the formula above) is a parameter that determines how much of the PageRank of an article can be modified by the articles that cite it. We initialize all article scores uniformly, set the damping factor to 0.85 and run 100 iterations of PageRank to determine the final article scores. These parameter settings were taken from [4] and [20].

It is also possible to characterize an article based on how often its authors or venue have been cited. Such statistics are common in bibliometric analyses of author and venue impact, but have not to our knowledge been used for information retrieval. We add a feature to boost articles by the citation count of their venue, similar to the *impact factor* of [7]:

$$\text{score}_{\text{venue-citation-count}}(q, d) = \sum_{d' \in D:\ \text{venue}(d') = \text{venue}(d)} |\text{citing}(d')|$$

We also add a feature to boost articles by the maximum citation count of their authors:

$$\text{score}_{\text{author-citation-count}}(q, d) = \max_{a \in \text{authors}(d)} \sum_{d' \in D:\ a \in \text{authors}(d')} |\text{citing}(d')|$$

As a variant of the latter, we also add a feature to boost articles by the maximum h-Index [8] of their authors. An author with an h-Index of h has published h papers each of which has been cited by others at least h times.

$$\text{score}_{\text{author-h-index}}(q, d) = \max_{a \in \text{authors}(d)} \text{h-Index}(a)$$

All of these features in some way measure the past popularity of an article, something we expect to be a strong predictor of future citation.

3.3 Recency

Researchers often want the most recent articles on a topic, thus a measure of the age of an article is important. We include as a feature the number of years since the article was published:

$$\text{score}_{\text{age}}(q, d) = \text{year}(q) - \text{year}(d)$$

We expect a negative weight on this feature – the older an article, the less likely it will be cited.

3.4 Cited Using Similar Terms

Search engines often include with the text of an indexed article the anchor text of all the links into that article [14]. The prior work on citing snippets is an extension of this idea to scientific articles, and has been used for both retrieval [3, 24] and summarization [17]. However, instead of concatenating the text and the citing snippets together as has been done in previous work, we introduce separate features for the text and the citing text so that our model can learn separate weights for each.

In our corpus, citing snippets are not immediately available – we only know that article A cites article B, not where in the text of article A the citation occurs. It might be possible to design patterns or a classifier to identify these citation locations, but we opt for a simpler approach that relies only on the text. We add a feature that compares the query terms to all the terms from all the citing articles, using the TF-IDF scoring described previously:

$$\text{score}_{\text{terms-citing}}(q, d) = \text{score}_{\text{terms}}\left(q, \underset{d' \in \text{citing}(d)}{\text{concat}} \left(\text{terms}(d')\right)\right)$$

We also introduce a feature that tries to select important citing terms using pointwise mutual information (PMI). We look for words (and bigrams) that are used to cite an article more often than we would expect by chance:

$$\text{pmi}(t, d) = \log \frac{p_{\text{termciting}}(t, d)}{p_{\text{term}}(t) p_{\text{citing}}(d)}$$

We calculate the probability of a term t, of citing an article d, and of a term t being used when citing an article d by counting occurrences of terms and citations in our training corpus (bigrams are counted similarly):

$$p_{\text{term}}(t) = \frac{\sum_{d' \in D} |\{t' \in \text{terms}(d') : t' = t\}|}{\sum_{d' \in D} |\text{terms}(d')|}$$

$$p_{\text{citing}}(d) = \frac{|\text{citing}(d)|}{\sum_{d' \in D} |\text{citing}(d')|}$$

$$p_{\text{termciting}}(t, d) = \frac{\sum_{d' \in \text{citing}(d)} |\{t' \in \text{terms}(d') : t' = t\}|}{\sum_{d'' \in D} \sum_{d' \in \text{citing}(d'')} |\text{terms}(d')|}$$

The intuition here is that words and bigrams with high PMI scores should be the terms people use when citing that article. For efficiency reasons – pairing all terms with all articles produces a very large number of pairs – we only generate PMI scores for articles that are cited at least five times and terms that occur at least twice in some citing article.

To create a feature from this information, we select the 1000 terms and bigrams that have the highest PMI with each article[2], and concatenate these into a pseudo-document we

[2] The number 1000 was selected on the development data, though the model did not seem to be very sensitive to the exact value chosen.

Topic 05	prosodic speech syllable phonological stress
Topic 11	alignment translation statistical alignments
Topic 12	resolution antecedent pronoun anaphora
Topic 38	frame semantic argument role arguments
Topic 48	event temporal time events tense
Topic 63	parsing parser grammar parse chart

Table 1: Top words from selected LDA topics.

call text_{pmi}. We then use TF-IDF scoring to compare the query against this pseudo-document:

$$\text{score}_{\text{terms-pmi}}(q, d) = \text{score}_{\text{terms}}(q, \text{text}_{\text{pmi}}(d))$$

Note that the text_{pmi} pseudo-document is just a subset of the terms used by the $\text{score}_{\text{terms-citing}}$ feature, but it offers the potential advantage that the terms are more targeted to the citation retrieval task.

3.5 Similar Topics

Finding relevant articles also means constraining the search to a relatively small number of subject areas. The terms in an article may hint at such topic distinctions, but it may also be useful to have a more explicit representation of these topics. Our corpus does not provide manually annotated subject headings, so we instead rely on statistical topic models to infer a reasonable set of topics. We selected latent Dirichlet allocation [2] to infer topics for our corpus as it been applied successfully to many tasks, and implementations are freely available[3]. We trained a 100 topic model on our corpus – the top words for a few of the topics are shown in Table 1.

These topics will form the basis for a number of different features introduced below. While various work in topic modeling has considered the interaction between topics and citation, e.g. [16, 18, 27], to our knowledge, this is the first time features based on topic models have been integrated into a literature retrieval system.

All of our topic features work by inspecting the probability distribution over topics assigned to a text. That is, for each of our $N = 100$ topics, our topic model predicts $p_{\text{topics}}(i, d)$, the inferred probability of topic i in document d. The distribution over all topics in document d is then:

$$\text{topics}(d) = \{p_{\text{topics}}(1, d), p_{\text{topics}}(2, d), \ldots, p_{\text{topics}}(N, d)\}$$

Note that latent Dirichlet allocation infers document topics in such a way that the above truly is a distribution – i.e. it will sum to 1. We use these topic distributions in some of the same ways we used the TF-IDF term vectors.

Our first topic-based feature uses cosine similarity[4] to compare the topic distribution assigned to the project idea with the topic distribution assigned to the article:

$$\text{score}_{\text{topics}}(q, d) = \cos(\text{topics}(q), \text{topics}(d))$$

Like the term similarity feature, we are converting text into vectors and comparing them with a cosine. The difference is that the vectors here represent topic distributions instead of TF-IDF word counts.

As an analog to the $\text{score}_{\text{terms-citing}}$ feature, we also introduce a feature comparing the query to the citing articles. First, we calculate the mean of the topic distributions of all the citing articles:

$$\text{topics}_{\text{citing}}(d) = \frac{\sum\limits_{d' \in \text{citing}(d)} \text{topics}(d')}{|\text{citing}(d)|}$$

Then we compare the query topic distribution with this mean-citing-topic distribution:

$$\text{score}_{\text{topics-citing}}(q, d) = \cos(\text{topics}(q), \text{topics}_{\text{citing}}(d))$$

A high $\textit{score}_{\textit{topics-citing}}$ means that others with topically similar projects have also cited this article.

Sometimes it may make sense to focus on the single most prominent topic. For example, if most semantic role labeling papers cite the Penn Treebank, then we should recommend that article for any semantic role labeling abstract. Thus we introduce the idea of *topic citation count*, where we represent each article by its most prominent topic, consider the articles it cites to be cited by this topic, and calculate for all articles the number of times they were cited by each topic. This results in 100 citation counts for each article, one for each topic. Our feature then boosts all articles by their citation count for the query's most prominent topic:

$$\text{toptopic}(d) = \arg\max_{i=1}^{N} p_{\text{topics}}(i, d)$$

$$\text{score}_{\text{topic-citation-count}}(q, d) = \\ |\{d' \in \textit{citing}(d) : \text{toptopic}(d') = \text{toptopic}(q)\}|$$

We can also get a notion of the breadth or depth of an article from its topics. This is important for identifying methodology papers, which are often cited by a wider topical range of articles. To measure the topical breadth of an article, we calculate the entropy of the article's topic distribution:

$$\text{score}_{\text{topic-entropy}}(q, d) = \sum_{i=1}^{N} -p_{\text{topics}}(i, d) \log p_{\text{topics}}(i, d)$$

To measure how broad an audience the article has reached, we calculate the entropy of the article's mean citing topic distribution[5]:

$$\text{score}_{\text{topic-entropy-citing}}(q, d) = \\ \sum_{i=1}^{N} -p_{\text{topics}_{\text{citing}}}(i, d) \log p_{\text{topics}_{\text{citing}}}(i, d)$$

Overall, we expect these topic-based features to better characterize the influences of subject areas on citation patterns. Such influences should be easier to identify with topics than they would have been using purely term based features.

3.6 Social Habits

In addition to the primarily content-based features discussed above, there are social factors that influence what articles an author considers relevant. We include the self-citation feature from prior work which indicates when an author prefers papers they have written themselves:

$$\text{score}_{\text{authors}}(q, d) = \text{score}_{\text{terms}}(\text{authors}(q), \text{authors}(d))$$

[3] We use Stanford TMT (`http://nlp.stanford.edu/software/tmt/`), with default settings for all model parameters.

[4] We also tried Kullback-Leibler divergence and Jensen-Shannon divergence, but cosine outperformed both of these on the development set.

[5] This feature is calculated in roughly the same way as *topic diversity*, introduced in Mann et al. [12].

Note that authors(x) here just consults the article metadata, and returns the authors as a list of terms. We can thus apply the default TF-IDF scoring of $\text{score}_{\text{terms}}$ over these author name-term vectors in the same way we did for regular term vectors.

We also introduce new author-based features that consider the past experience of an author and capture their preferences for articles, authors and venues. The first such feature boosts articles that have been previously cited by the query authors:

$$\text{score}_{\text{authors-cited-article}}(q, d) = \\ \text{score}_{\text{terms}}(\text{authors}(q), \underset{d' \in \text{citing}(d)}{\text{concat}} (\text{authors}(d')))$$

The next feature boosts articles written by people that the query authors have previously cited:

$$\text{score}_{\text{authors-cited-author}}(q, d) = \\ \text{score}_{\text{terms}}(\text{authors}(q), \underset{\substack{d'' \in \text{citing}(d'): \\ d' \in D \wedge \text{authors}(d') \cap \text{authors}(d) \neq \emptyset}}{\text{concat}} (\text{authors}(d'')))$$

We also boost articles published at venues that the query authors have previously cited:

$$\text{score}_{\text{authors-cited-venue}}(q, d) = \\ \text{score}_{\text{terms}}(\text{authors}(q), \underset{\substack{d'' \in \text{citing}(d'): \\ d' \in D \wedge \text{venue}(d') = \text{venue}(d)}}{\text{concat}} (\text{authors}(d')))$$

Finally, we boost articles written by people with whom the query authors have co-authored:

$$\text{score}_{\text{authors-coauthored}}(q, d) = \\ \text{score}_{\text{terms}}(\text{authors}(q), \\ \underset{d' \in D : \text{authors}(d') \cap \text{authors}(d) \neq \emptyset}{\text{concat}} (\text{authors}(d') - \text{authors}(d)))$$

All these features try to tailor the retrieval to match the past behavior of an author. Of course, for authors that have never been seen before, these features will not add any information.

4. RETRIEVAL MODEL

We combine all these features into a scoring function that ranks the retrieved articles. Our model is essentially a weighted sum of feature scores:

$$\text{score}(q, d) = \sum_i w_i \times f_i(q, d)$$

where q is the query abstract, d is the potentially relevant article, w_i are the feature weights and f_i are the feature scores.

Both to make it easier to interpret the feature weights, and because in experiments on the development set we found it increased performance, we perform some normalization of the feature scores. First, we put all features into log-space to better cope with features like citation count where the difference between 1 and 5 citations is more important than the difference between 201 and 205:

$$f_i(d, q) = \text{scale}_i(\log(1 + \text{score}_i(d, q)))$$

Then, we scale these log-space feature scores to the interval $[0, 1]$ based on the maximum and minimum values of f_i observed in the training data:

$$\text{scale}_i(s) = \frac{s - min_i}{max_i - min_i}$$

If the value in the test data is below 0 or above 1, we clip it to the endpoint. This simple zero-one scaling often performs as well as more elaborate schemes [29].

Algorithm 1 Iterative weight learning

```
procedure LEARN(train articles $T$, dev articles $D$, hits $N$)
    $w_{\text{terms}} = 1.0$
    for $i : i \neq$ terms do
        $w_i = 0.0$
    $data$ = []
    repeat
        $model \leftarrow$ BUILDRETRIEVALMODEL($w$)
        $map \leftarrow$ MEANAVGPREC($model$, $D$, $N$)
        $data \leftarrow data$ + TRAININGDATA($model$, $T$, $N$)
        $classifier \leftarrow$ LEARNCLASSIFIER($data$)
        $w \leftarrow$ WEIGHTS($classifier$)
    until $map$ converges

procedure TRAININGDATA(model $m$, articles $T$, hits $N$)
    $data$ = []
    for $d \in T$ do
        $q \leftarrow$ TERMS(TITLE($d$)) + TERMS(ABSTRACT($d$))
        for $d' \in$ Retrieve($m$, $q$, $N$) do
            if $d' \in$ CitedArticles($d$) then
                $label \leftarrow +1$
            else
                $label \leftarrow -1$
            $data \leftarrow data$ + ($label$, FEATURES($q$, $d$))
    return $data$
```

To learn the feature weights of our model, we train classifiers based on the iterative process shown in Figure 1. First, we run the baseline (terms-only) retrieval model, collecting N articles for each abstract. Retrieved articles that are also found in the abstract's reference list are labeled +1, and the other retrieved articles are labeled −1. A linear classifier is then trained on this data, using the same features as the retrieval model. Since the model is linear, when training is complete, we can read the feature weights off the classifier and insert them directly into the retrieval model. The same process can be repeated with the updated retrieval model to collect another set of retrieval lists, add these to the training data and train a new classifier. The process continues until the model mean average precision on the development set converges or a set number of iterations is reached.

The key point of this process is that since the model weights are being updated on each iteration, the retrieval model is typically adding some new examples to the training set on each iteration[6]. We do not remove duplicates, so at iteration k, there are kN items in the training data. Of course, if the number of items in the training data were all that mattered, we could simply set N to kN from the beginning (e.g. retrieve 2000 results per article once instead of retrieving 100 results per article 20 times). This is the normal approach that has been taken in prior work. However, intuitively the iterative

[6] This iterative selection of training examples is similar in spirit to the Minimum Error Rate Training (MERT) method used in machine translation [19].

Venue	Papers	Years
CL (Journal)	535	1979-2005
COLING	2181	1965-2004
ACL	2140	1979-2007
HLT	940	1986-2005
EACL	588	1983-2006
NAACL	387	2001-2007
ANLP	335	1983-2000
MUC	182	1991-1998
IJCNLP	143	2005-2005
TINLAP	119	1975-1987
TIPSTER	114	1993-1998
Workshops	3270	1990-2007

Table 2: Venues in the ACL Anthology

	Train	Dev	Test
Years	2000-2003	2004	2005-2006
Articles	613	319	794
References	5027	2600	7020
Refs/Article	8.2	8.2	8.8

Table 3: Training, development and test data from the ACL Anthology Reference Corpus.

	$N = 100$		$N = 2000$
Classifier	1 iter	20 iter	1 iter
Logistic	7.9	16.8	10.7
Logistic 50/50	19.9	24.9	25.7
SVM-MAP	25.3	**27.9**	25.5

Table 4: Mean average precision on the development data using different classifiers and all features. The Logistic 50/50 model downsamples the number of negative examples to match the number of positive examples.

approach is adding articles that the model wrongly thinks are relevant and needs to learn are not, while the non-iterative approach is adding more irrelevant results from the tail of the retrieval. Thus we expect that the iterative approach will learn better retrieval models.

We considered two types of classifiers for the weight learning step: a logistic classifier trained with a quadratic prior ($\sigma = 1.0$) using L-BFGS [11], and an SVM-MAP [30] classifier which trains a support vector machine (SVM) to optimize mean average precision (MAP). Note that the logistic classifier is optimizing only to the +1/-1 labels, while the SVM-MAP classifier considers the order in which articles are retrieved and optimizes directly to the MAP. Thus, we expect the SVM-MAP classifier to learn more appropriate weights to our task. We also observed in early experiments that the logistic classifier had difficulty with the skew in the retrieved data (many more non-relevant than relevant articles), so we also considered a variant of the logistic classifier training in which we downsampled the negative examples for each article to match the number of positive examples.

5. EXPERIMENTS

We evaluated our literature search models on the ACL Anthology Reference Corpus (ACL-ARC) [1], a set of 10,921 papers from computational linguistics workshops, conferences and journals, summarized in Table 2. For each article, the full text is available, as is metadata containing normalized author names, venues, titles and citation information.

For our retrieval experiments, we constructed a query by concatenating an article's title and abstract, had our retrieval model find relevant articles for this query, and compared the results against the references list of the query article. Abstracts were not annotated in the data, so we extracted them using a set of manually constructed patterns. For the reference lists, we discarded references to articles outside of the corpus, and only considered articles that had at least five references remaining. Table 3 shows the sizes of our training, development and test sets.

Note that for all training and evaluation, we only used features calculated over previous years. For example, when retrieving articles for the abstracts published in 2004, all the articles up through 2003 were indexed, and only the titles, abstracts, and authors of the articles from the year 2004 were available (as the queries). Thus, time dependent features like citation count would only include citations from papers published in 2003 or earlier. Structuring the evaluation in this way is more realistic – when presented with new project ideas, a literature search system can only predict appropriate references based on the patterns it has previously observed.

Given this setup, we applied our iterative training algorithm to build a retrieval model. During learning, we retrieved 100 articles for each query, allowed the algorithm to run for at most 20 iterations, and selected the model from the iteration that achieved the highest MAP on the development set. The SVM C parameter was set by learning a model for each C value of 1, 10, 100, 1000 and 10000, and selecting the C value with the highest MAP on the development set. Typically either $C = 1000$ or $C = 10000$ was selected.

We evaluate the models using mean average precision (MAP), which gives the highest score when all relevant articles precede all non-relevant articles in the retrieval results. MAP is defined in terms of *precision* and *average precision*. Precision determines what percent of the list of retrieved results (D) were also in the list of relevant articles (R):

$$\text{precision}(R, D) = \frac{|R \cap D|}{|D|}$$

Average precision calculates precision at each point where a relevant article was found:

$$\text{aveprec}(R, D) = \sum_{i \in [1, |D|]:\, D_i \in R} \frac{\text{precision}(R, D_{[1:i]})}{|R|}$$

Mean average precision is then the average of all average precisions across a set of test queries:

$$\text{map}(Q) = \sum_{q \in Q} \frac{\text{aveprec}(\text{relevant}(q), \text{retrieved}(q))}{|Q|}$$

5.1 Classifier Analysis

Using these metrics, we first compared the effects of using different classifiers to learn the model weights. As Table 4 shows, the SVM-MAP model outperformed both logistic classifiers ($p < .001$)[7], achieving a mean average precision

[7] We run a Student's t-test and a Wilcoxon signed-rank test over the paired per-document average precisions, and report the most conservative p value.

Feature	Dev MAP	Test MAP
Only *terms* feature	15.4	15.9
Related work features	24.0	24.8
All features	**27.9**	**28.7**
All features (only 1 iteration)	25.3	26.9

Table 5: Mean average precision on the development and test data using SVM-MAP with different feature sets.

of 27.9, compared to the 16.8 and 24.9 of the two logistic classifier variants. Still, all classifiers improved ($p < .001$) with our iterative learning procedure.

We also found that the iterative training procedure was generally better than working with a larger number of examples from the start. We compared retrieving 100 training examples on each of 20 iterations to retrieving the full 2000 training examples on a single iteration. As shown in the last two columns of Table 4, for both the simple logistic classifier and the SVM-MAP model, iterative learning outperformed a larger retrieval size ($p < .001$). For the downsampling logistic classifier, the larger retrieval size was slightly better than iteration ($p = .03$), though still worse ($p < 0.001$) than the iterative SVM-MAP model.

5.2 Feature Analysis

Thus, we selected the iterative SVM-MAP model to evaluate on our test set. Table 5 compares our model against two baselines. The first baseline uses only the *terms* feature, and is equivalent to downloading Lucene and using it out of the box. The second baseline is an approximation of prior work and uses all of the features we collected from related research: *terms*, *citation-count*, *pagerank*, *age*, *terms-citing* and *authors*. Most related work was on different tasks, so we were unable to compare directly, but these features cover most of what previous research has tried, so we believe this model is a reasonable approximation. Our model outperforms ($p < 0.001$) both of these baselines, achieving a MAP of 28.7 – a 12.8 point increase over the term similarity baseline, and a 3.9 point increase over the related work baseline.

To get a better understanding of the influence of each feature, we performed three analyses. First, we inspected the feature weights themselves, since they were normalized to the same range so as to be roughly comparable. Second, we trained models with just the *terms* feature and a single other feature to see how much of the MAP each feature could produce on its own. Finally, we trained models with all features except for one, and observed by the drop in MAP how much the model depended on the missing feature.

Feature	Weight	Add	Drop
terms (baseline)	1.67	0.0	–
citation-count	**0.58**	3.2^{***}	1.9^{***}
pagerank	0.06	0.2^{***}	0.1
venue-citation-count	0.07	1.0^{***}	0.1
author-citation-count	0.11	1.0^{***}	-0.2
author-h-index	-0.06	0.8^{***}	-0.1
age	**-0.58**	0.2	2.4^{***}
terms-citing	**0.59**	5.0^{***}	0.3
terms-pmi	0.13	3.1^{***}	0.2
topics	0.22	1.7^{***}	0.2
topics-citing	0.16	3.1^{***}	0.4
topic-citation-count	0.02	0.7^{***}	-0.1
topic-entropy	-0.17	0.0	0.0
topic-entropy-citing	-0.28	0.0	1.5^{***}
authors	**0.68**	1.1^{**}	0.3
authors-cited-article	**0.68**	1.7^{***}	0.2
authors-cited-author	0.21	1.0^{***}	0.5
authors-cited-venue	0.39	0.1	0.2
authors-coauthored	0.02	0.1	0.1
All features	–	12.5	12.5

Table 6: Feature analysis: learned feature weight (Weight), increase in MAP when added to the *terms* model (Add), and decrease in MAP when dropped from the *all-features* model (Drop). The top 5 weights are bold, and significant changes in MAP are marked with ** ($p < 0.01$) or *** ($p < 0.001$).

Table 6 shows these analyses. 16 of our 18 features made significant contributions to the model, either when added to the *terms* model or when removed from the *all-features* model. The best single features were *terms-citing*, *terms-pmi*, *citation-count* and *topics-citing*, suggesting that not only is it crucial to include the number of times an article was cited, but that combining this with the terms or topics from the citing articles was particularly effective. Features whose removal led to a significant drop in mean average precision included *citation-count*, *age* and *topic-entropy-citing*. The latter two are interesting because neither of these produced significant increases when added to the baseline *terms* model. The fact that their removal results in a significant decrease suggests that these important features are inaccessible to the many existing retrieval models that do not consider multiple features at the same time.

Our feature analysis also gives some insight into sociological phenomena. For example, the negative weight on the *topic-entropy-citing* feature indicates that authors prefer to cite papers that are of interest to a small number of fields, not papers with broad impact across many fields. We also see a high weight assigned to the self citation feature (*authors*), suggesting that even after taking all our other features into account, there is some benefit in assuming that authors will favor their own papers. Note that this is not necessarily a bad thing - self-citation is common across disciplines [25], and actually has a positive correlation with citation by others [5]. We also saw a trend for "favorite papers" in the highly weighted *authors-cited-article* feature – authors like to cite articles they have cited before.

5.3 Example Article

To give an idea of some of the issues our model still had trouble with[8], Table 7 shows the top eight results retrieved for the article "Parsing Arguments Of Nominalizations In English And Chinese", on which our model had a number of errors. The three articles marked with + were present in the reference list of the article, while the other five were not.

We gave the output of our system to one of the authors of the article and asked for help in understanding why these papers weren't cited. Of the five errors, W03-1006 and W03-1008 are both correctly semantic role labeling papers, but neither considers nominal predicates and are thus not relevant. Fixing this error would probably require some understanding that the primary contribution of the query ar-

[8] For a more hands-on analysis, a demo is also available at `http://nlp.stanford.edu:8080/citation-retrieval/`

+	J02-3001	Automatic Labeling Of Semantic Roles
+	P00-1065	Automatic Labeling Of Semantic Roles
-	W03-1007	Maximum Entropy Models For FrameNet Classification
+	P03-1002	Using Predicate-Argument Structures For Information Extraction
-	W03-1006	Use Of Deep Linguistic Features For The Recognition And Labeling Of Semantic Arguments
-	N03-2008	A Maximum Entropy Approach To FrameNet Tagging
-	W03-1008	Identifying Semantic Roles Using Combinatory Categorial Grammar
-	N03-2022	Semantic Extraction With Wide-Coverage Lexical Resources

Table 7: Retrieved articles for "Parsing Arguments Of Nominalizations In English And Chinese", by Sameer S. Pradhan, Honglin Sun, Wayne H. Ward, James H. Martin and Daniel Jurafsky (N04-4036). A + indicates that the article was found in the references list, a - indicates it was absent.

ticle is in parsing the arguments of *nominalizations*. Another error, N03-2008, was a brief 4-page short paper that was expanded later as W03-1007, a paper the system also suggested. Suggesting both papers is an error that to fix would require joint or global inference to reason that a workshop paper is less relevant if the longer conference paper is cited. The final two errors, W03-1007 and N03-2022, don't deal with nominalizations either, but since they both concern semantic role labeling of FrameNet, the author told us that they were considered for inclusion in the paper and omitted mainly for space and also partly because they were less directly relevant to nominalizations. These two out of our five errors were thus at least appropriate suggestions.

6. CONCLUSION

We have presented a model for scientific article retrieval that introduces new features and a new learning algorithm that outperform previous approaches. Our model includes many features previously absent from literature retrieval systems, such as author impact, author citation habits, topic similarity and citing topic entropy, as well as useful features gathered from related work, such as citation count, publication age and citing terms. We show that the weights for these features can be more effectively learned by an iterative training procedure which alternates between retrieving articles with the current retrieval model, and updating the retrieval model weights by training a classifier these articles.

We evaluated our model on the ACL Anthology Reference Corpus, giving our system abstracts as input, and evaluating it by how well it produced the reference lists of the corresponding articles. Our model achieved a mean average precision of 28.7, significantly outperforming a term-similarity baseline, a model using only features from prior work, and a model with all features but trained in the non-iterative fashion typical of prior work. Analyses of the model showed that augmenting word-based features with topic and author-based features allowed us to take advantage of interesting trends in scientific citation, for example: authors cite themselves, authors cite their favorite papers, and authors cite articles with narrow impact to only a few related topic areas.

Our work offers a number of interesting directions for future research. First, in this work we make the simplifying assumption that the articles actually cited by a paper are the same as the articles that should be cited for that paper. In practice however, we observe that some unnecessary citations are present and some appropriate citations are absent. It would be interesting to re-evaluate our models using a collection like that of Ritchie et al. [23] which includes relevance judgments for each article citation, where we would be able to distinguish between important and unimportant articles that our model failed to find. We would also like to run a more extensive manual evaluation of our system to determine how often relevant articles are being proposed but happen not to be in the references list.

Work is also needed to determine whether the feature weights and scientific trends we observed are specific to our computational linguistics corpus or are indicators of more general trends. Larger citation databases like Citeseer[9] or ISI's Web of Knowledge[10] include articles from many more domains, and the citation networks in such databases may exhibit different structures than those observed in our corpus. We could train our models on such corpora to see whether the learned feature weights and interactions between features follow the patterns observed here or reveal new trends in scientific citation.

Perhaps the most interesting future direction is to investigate how the disciplines of science differ from one another. Thus, we are currently looking at methods for analyzing how the importance of features changes across disciplines, e.g. by learning a separate retrieval model for each subject area. Can we characterize, for example, which disciplines rely more heavily upon their personal social networks? Answering such questions would advance our understanding of the scientific process, and at the same time enable more personalized, useful literature retrieval systems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0835614. Thanks also to the Office of the President, Stanford University, for partial funding.

References

[1] S. Bird, R. Dale, B. J. Dorr, B. Gibson, M. Joseph, M.-Y. Kan, D. Lee, B. Powley, D. R. Radev, and Y. F. Tan. The ACL Anthology Reference Corpus: A reference dataset for bibliographic research in computational linguistics. In *Language Resources and Evaluation Conference*, 2008.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3: 993–1022, Jan. 2003.

[3] S. Bradshaw. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Re-*

[9] `http://citeseerx.ist.psu.edu`

[10] `http://www.isiwebofknowledge.com/`

search and Advanced Technology for Digital Libraries, pages 499–510. Springer Berlin / Heidelberg, 2003.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[5] J. Fowler and D. Aksnes. Does self-citation pay? *Scientometrics*, 72(3):427–437, 2007.

[6] A. Fujii. Enhancing patent retrieval by citation analysis. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 793–794, 2007.

[7] E. Garfield. Citation indexes to science: a new dimension in documentation through association of ideas. *Science*, 122:108–111, 1955.

[8] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569–16572, Nov. 2005.

[9] B. Larsen. Exploiting citation overlaps for information retrieval: Generating a boomerang effect from the network of scientific papers. *Scientometrics*, 54(2):155–178, June 2002.

[10] B. Larsen and P. Ingwersen. Using citations for ranking in digital libraries. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 370–370, 2006.

[11] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[12] G. S. Mann, D. Mimno, and A. McCallum. Bibliometric impact measures leveraging topic analysis. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 65–74, 2006.

[13] C. D. Manning, P. Raghavan, and H. Sch utze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.

[14] O. A. McBryan. Genvl and wwww: Tools for taming the web. In *First International Conference on the World-Wide Web*, 1994.

[15] E. Meij and M. de Rijke. Using prior information derived from citations in literature search. In *Recherche d'Information AssistÃl'e par Ordinateur (RIAO)*, 2007.

[16] D. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, 2008.

[17] S. Mohammad, B. Dorr, M. Egan, A. Hassan, P. Muthukrishan, V. Qazvinian, D. Radev, and D. Zajic. Using citations to generate surveys of scientific paradigms. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 584–592, 2009.

[18] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550, 2008.

[19] F. J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[21] M. L. Pao. Term and citation retrieval: A field study. *Information Processing & Management*, 29(1):95–112, 1993.

[22] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12(5):297–312, 1976.

[23] A. Ritchie, S. Teufel, and S. Robertson. Creating a test collection for citation-based ir experiments. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 391–398, 2006.

[24] A. Ritchie, S. Robertson, and S. Teufel. Comparing citation contexts for information retrieval. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 213–222, 2008.

[25] H. Snyder and S. Bonzi. Patterns of self-citation across disciplines (1980-1989). *Journal of Information Science*, 24(6):431–435, Dec. 1998.

[26] T. Strohman, W. B. Croft, and D. Jensen. Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–706, 2007.

[27] J. Tang and J. Zhang. A discriminative approach to Topic-Based citation recommendation. In *Advances in Knowledge Discovery and Data Mining*, pages 572–579. Springer Berlin / Heidelberg, 2009.

[28] I. Tbahriti, C. Chichester, F. Lisacek, and P. Ruch. Using argumentation to retrieve articles with similar citations: an inquiry into improving related articles search in the MEDLINE digital library. *International Journal of Medical Informatics*, 75(6):488–495, June 2006. ISSN 1386-5056.

[29] S. Wu, F. Crestani, and Y. Bi. Evaluating score normalization methods in data fusion. In *Asia Information Retrieval Symposium (AIRS)*, pages 642–648, 2006.

[30] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278, 2007.