# A two-stage citation recommendation system

Julian Kates-Harbeck
Stanford University
juliankh@stanford.edu

Mickey Haggblade
Stanford University
mickeyh@stanford.edu

December 6, 2013

**Note:** We counted this project towards both CS221 and CS224N.

## 1 Task Definition

**Functionality** Given a single manuscript query paper and a set of full candidate papers our algorithm should rank the candidates by relevance to the query. Such an automated "reference finder" could be very useful for scientists from all disciplines, who generally have to manually search in a multi-stage process in order to find relevant works.

**Input** (identified by an abstract, a list of authors, and a publication date) The manuscript query paper consist of an abstract, i.e. a short (100-300 word) sample of text with a high concentration of technical words, an author list and a publication date. The set of full candidate papers is simply a set of papers to be ranked by relevance to the manuscript query.

**Output** The program returns a ranking of candidate papers by relevance to the query.

## 2 Previous Work

The goal of our project is to improve upon the standard tools used by scientists to discover relevant works. The most widely used tools with the largest databases, such as Web of Science and JSTOR, offer only simple topic hierarchy and keyword-based search. This often yields poor results when variations in terminology and re-use of technical terms lead to poor ranking, many irrelevant results, and the omission of important references.

We will organize scientific publications in a way that leverages the substantial work that authors have *already* done in a) building their own lists of relevant citations, and b) writing with the goal of drawing interest from those in related fields. Furthermore, we use textual analysis to build a search tool capable of using substantially more information than simple phrases, i.e. an entire abstract, to better refine results.

Google Scholar [1] is the closest tool to what we plan to build, as it offers a "related articles" functionality to branch out from a particular paper. GoPubMed [2], a semantic search engine for medical literature, also provides similar functionality. However, both of these are closed, commercial systems, and the underlying algorithms are not known.

Some work has been performed recently [3, 4, 5] on devising automated citation recommendation systems. The common approach in these projects is to learn features that can discriminate between relevant and irrelevant candidate papers for a given query paper. In general, these features can involve both textual similarity between the two papers, as well as context-based information about the two papers – such as past citation behavior of the authors of the query paper, or the popularity of the candidate paper. In this project, we use a similar feature-based approach. In addition, we implement an original ranking algorithm to improve relevancy by considering the connectivity information of papers after scoring by the classifier. To the best of our knowledge, our algorithm had not been used in citation recommendation before.

## 3 Data

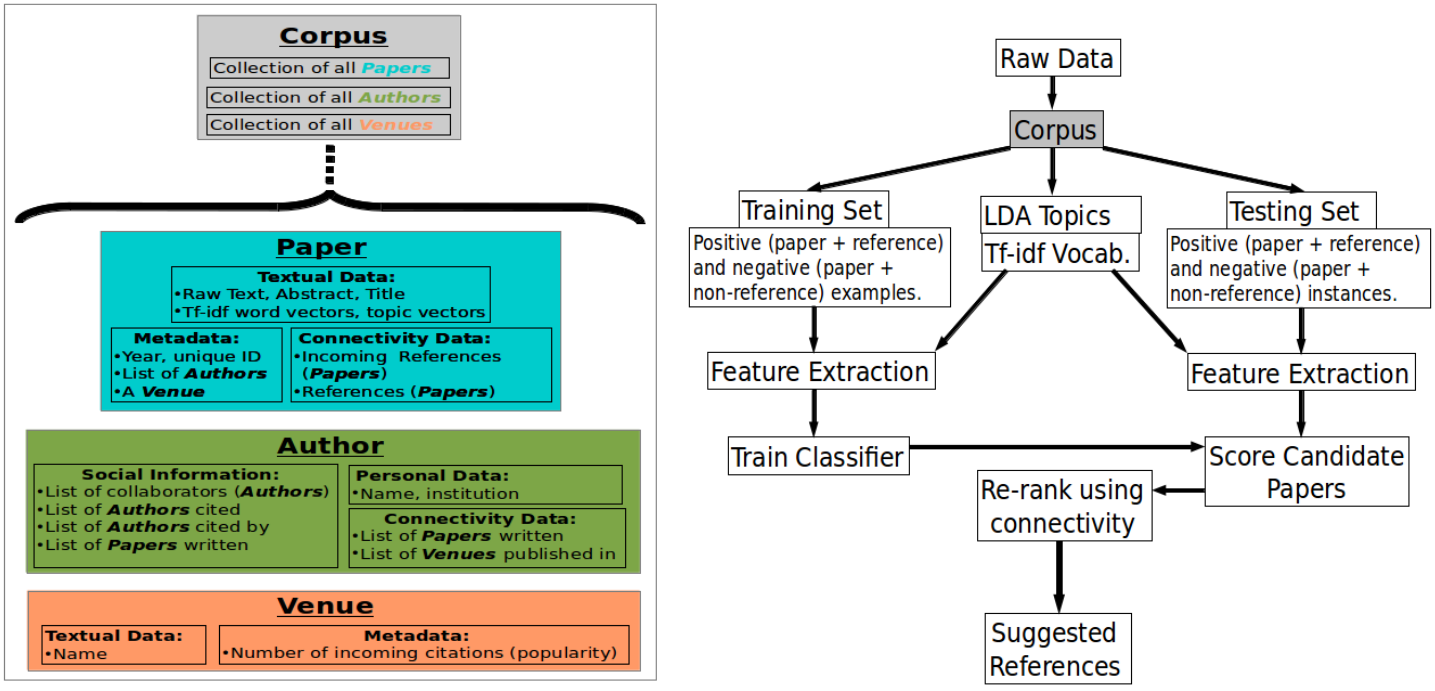
### 3.1 Data Set

We use the Association of Computational Linguistics (ACL) Anthology Network as our data set [6]. This data set consists of raw text data describing a large set of papers –including authors, venue, publication year, and raw text — and their citations — consisting of other papers[1]. In total, the data set contains $19,647$ paper, $16,152$ authors, and $94,973$ citations. A significant portion of our labor was concerned with reading the data, cleaning it, and transforming it into usable form. This was nontrivial in unexpected ways. For example, because different files in the dataset had authors names spelled differently (they coped with special characters differently), we had to use a variation of Levenshtein distance to tell if authors were the same or not. Another example was that we had write custom scripts to locate and extract the abstract from the raw paper text, since it was not provided as separate data.

### 3.2 Processed Data

Figure 2 summarizes the layout of our data set after reading, cleaning, processing, and transforming it. The data consists of a *corpus*, which in turn holds a list of *papers*, *authors*, and *venues*. Each of these is a seperate data structure, holding both internal information, as well as connectivity information. For instance, each paper holds its raw text, but also holds a list of papers that it

[1] When we found out about this dataset we were very excited because we thought that it would save us the significant overhead of scrape papers ourselves. Ultimately it did save us time, but not as much as we'd hoped. It was still a giant pain in the neck to clean and structure the data, actually




(a) Internal data structure

(b) Processing pipeline

Figure 1: Schematics of the internal data structures (a) and the processing pipeline (b) used in the project.

references. Each author holds a list of papers that she has written, and a list of past collaborating authors, but also her name and affiliation. In this way, our corpus represents a single connected data structure containing all relevant information about authors, papers, and venues, as well as their connectivity information — with respect to citation, co-authorship, collaboration, and so on.

# 4 Algorithm

## 4.1 Overview

We want to create an algorithm that can predict with high accuracy the papers most likely to be cited by a given query paper. For this purpose, we distinguish between the *query* paper — the paper for which we want to find relevant literature — and the *candidate* papers — the set of papers to rank by relevance to *query*.

To rank the *candidates* we use the measure of confidence of a stock machine learning classifier (we ended up using an SVM trained with Stochastic Gradient Descent provided by scikit-learn [7]). To do this, we extract a set of features from each (*query*, *candidate*) pair to feed to the classifier. We train a classifier on labeled positive and negative examples of query and candidate papers[2] for these features. The classifier will indicate whether or not we believe *candidate* is relevant to *query* (in practice this is whether or not *query* cites *candidate*).

Finally, we use a newly developed algorithm to re-rank the scored candidate papers using connectivity information. The intuition is similar to the PageRank [8] algorithm used in search engines. Given a query paper, the final score of a candidate paper is a combination of the local confidence of the classifier for that particular candidate paper *combined* with a measure of confidence of all the neighbors of the candidate paper — where connectivity is defined by incoming our outgoing citations.

Figure 1b gives an overview of the processing pipeline used in our project.

## 4.2 Formal Description

Abstractly, we can phrase our algorithm as follows. We are given a query paper $q$ and a set of $n$ candidate papers $P = \{p_i | i \in \{1,...,n\}\}$. We assume there is a gold standard probability

$$P_{cite}(q, p_i) = P(\text{Paper } q \text{ cites paper } p_i) \ .$$

We want to rank $P$ by $P_{cite}(q, p_i)$. For this purpose, we want to learn an approximation

$$\bar{P}_{cite}(q, p_i) \approx P_{cite}(q, p_i) = P(\text{Paper } q \text{ cites paper } p_i)$$

from the data. We choose a model of the form

$$\bar{P}_{cite}(q, p_i) \equiv g(z) \ . \tag{1}$$

In this case, $z$ is a function of the query paper and the candidate paper, as well as of a set of weights $\mathbf{w}$:

$$z = z(q, p_i | \mathbf{w}) \equiv z(\phi(q, p_i) | \mathbf{w}) \ , \tag{2}$$

[2] A positive example is when the query paper actually cites the candidate paper. A negative example is when the query paper does not.

where we used boldface to denote vector quantities. Now, our data consists of papers with lists of references, which are again papers. Let $R(p) \subset P$ denote the set of references of paper $p$, where we recall that $P$ denotes the data set of all papers. For every paper $p$ in our data set, we can thus produce $n$ training data of the form $(x_i, y_i)\ \forall i \in \{1, ..., n\}$, where

$$\begin{aligned} \mathbf{x}_i &= \phi(p, p_i) \\ y_i &= \begin{cases} 1 & \text{if } p_i \in R(p) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

In words, we produce a data tuple for every combination of papers from our raw data set and set the label of that tuple to 1 iff the first paper cites the second. We can now apply our classifier to this data (the pairs $\{(\mathbf{x}, y)\}$). In practice we limit the number of examples with $y = 0$ because otherwise we could get a strong imbalance between the number of positive and negative examples, which hurts classifier performance.

Given a query paper $q$ and a candidate paper $p_i$, we use $\bar{P}_{cite}(q, p_i) \equiv g(z) = g(\phi(q, p_i)|\bar{\mathbf{w}})$ to quantify the relevance of $p_i$ to $q$.

## 4.3 Outline

With the above definitions, we can describe our project pipeline concisely.

1. Obtain raw training data in form of papers and their citations.
2. Define a function $\phi(q, p_i)$ that returns a feature vector given a query paper and a candidate paper.
3. Use the raw data to produce a set of labeled training data of the form $\{(\mathbf{x}_i, y_i)\}$, as described in equation 3.
4. Run a machine learning algorithm using the feature function $\phi$ on the training data to obtain an estimate for the weights $\bar{\mathbf{w}}$.
5. For a given query paper $q$, we can obtain the estimate $\bar{P}_{cite}(q, p_i) \equiv g(z) = g(\phi(q, p_i)|\bar{\mathbf{w}})\ \forall p_i \in P$.
6. Rerank each candidate paper using the KRank measure, giving a combination of the score $\bar{P}_{cite}(q, p_i)$ of the paper itself and the scores of its neighbors.
7. For a query $q$, we rank $k$ papers $p_i$ by the estimate $\bar{P}_{cite}(q, p_i)$.

## 4.4 Feature Extraction and Modeling

### 4.4.1 Feature description

We seek to define features that build a good *model* of when papers are likely to cite each other. The ML takes care of tuning the model. Performance tuning using training and development sets, evaluated as outlined in section 6, will indicate better and better features to use.

In general, we can distinguish *context-based* features and *text-based* features. Context-based features capture information related to the context of a given paper, i.e. the authorship, venue, year of publication, etc. This type of feature captures social citation patterns of the authors, identifies popular papers, etc. On the other hand, text-based features encode the similarity of the actual content of the query and candidate papers.

As of now, we propose the following **context-based** features given a query $q$ and a candidate $p$:

**Paper Popularity** : A feature giving the citation number of the candidate paper, reflecting its popularity.

**Author Popularity** : A feature giving the maximum citation number of the candidate paper's authors, reflecting their popularity.

**Venue Popularity** : A feature giving the citation number of the candidate paper's venue, reflecting its popularity.

**Recency** : Difference between Date($q$) and Date($p$) in days. The intuition is that recent papers might be cited more often than old papers.

**Recency Indicator** : Binary feature indicating whether or not Date($q$) > Date($p$). The intuition is that papers newer than the query are never cited by the query.

**Same Venue** : Indicator feature denoting whether both papers come from similar venues.

**Authorship Overlap** : Overlap between Authors($q$) and Authors($p$). The intuition is that authors often cite papers by themselves or their collaborators.

**Institution Overlap** : Overlap between the institutions of Authors($q$) and Authors($p$).

**Have Collaborated** : The amount of collaboration between query and candidate authors that has happened in the past.

**Previously Cited Author** : The amount by which authors of the query paper have cited authors in the candidate paper in the past.

**Previously Cited Paper** : The amount by which authors of the query paper have cited the candidate paper in the past.

**Previously Cited Venue** : The amount by which authors of the query paper have cited the venue of the candidate paper in the past.

Moreover, we implemented the following **text-based features**:

**TFIDF Textual similarity** : Bag-of words (TFIDF) similarity between abstract($q$) and abstract($p$).

**Topic similarity** : Topic similarity between abstract($q$) and text($p$), measured by cosine distance between the respective topic vectors. The reasoning behind the Tf-idf word vectors is to capture a combination of Keyword similarity and overall bag-of-words similarity. The word vectors weight each of the words by their inverse frequency and thus act as to add more weight to rarer, more important words. In this sense it behaves like a generalized keyword extractor. On the other hand, larger sets of words are captured in the Tf-idf vectors, since no cutoff is made.

In addition, we use an LDA topic model [9] trained in an unsupervised preprocessing stage to extract a topic-vector representation for each text. This feature attempts to capture the overall topic overlap of the two documents. We use the *gensim* implementation [10] of LDA topic modeling.

#### 4.4.2 Classifier tuning

**Feature scaling** Since we are using a linear classifier, we use a scaling algorithm to remove the mean and enforce unit variance for all our features. This scaling improves the MAP results by a few percent. We also tried scaling all values to the interval $[0,1]$, as was done in [3], which also improved the overall performance, but not as much as the standard scaling.

**Training examples** One hyperparameter worth mentioning was the number of positive and negative training examples. Each paper cites only a limited number of other papers, but nearly every paper in the corpus can be paired with a query paper to produce a negative training example. We thus had to limit the number of negative training examples to not negatively affect the classifier. We found that a ratio of $\sim 10$ gave best performance.

**Feature nonlinearities** For several features, the simple linear version of the feature often overwhelmed the classifier, negatively affecting performance. For example, initially, we had paper popularity as a linear feature. However, since the classifier gave it significant weight, there were some papers[3] (those with a very large number of citations) that were almost always recommended at the top due to their large weight, negatively affecting performance. After projecting features with such an uneven distribution of feature values into log space, our performance increased by a few percent (and the top recommendations became more individualized).

# 5 Novel Reranking Scheme: KRank

## 5.1 Motivation

After scoring all candidate papers for a given query paper using the classifier, we use a newly developed algorithm to re-rank the scored candidate papers using connectivity information. Intuitively, the classifier itself only uses "local" information about any given candidate paper — the authors of the paper, the similarity between the candidate and the query, etc — to produce a score. A more holistic scoring approach should also take advantage of the more global connectivity information of papers. Intuitively, a paper that is cited by (or itself cites) many high-scoring papers is more likely to be relevant. This is exemplified by the commonly-used "human citation recommendation algorithm". When actual people look for references (in a field that they are not intimately familiar with), they often use a simple similarity metric (such as keyword search on google scholar, important references from the last paper they read, etc.) and then use the respective references of those "starting papers" to find more relevant candidate papers.

Our algorithm — we call it "KRank" — attempts to capture this approach, albeit in a more systematic and general manner. The intuition is similar to the PageRank [8] algorithm used in search engines. Given a query paper, the final score of a candidate paper should be a combination of the local confidence of the classifier for that particular candidate paper *combined* with a measure of confidence of all the neighbors of the candidate paper — where neighborhood and connectivity is defined in terms of citations. If we look at a candidate paper, it will increase our confidence if it is cited by (or cites) many other high-scoring, relevant papers. On the other hand, if it cites only irrelevant papers, and is cited by only irrelevant papers, this will decrease our confidence. In order to enforce self-consistency, we define the confidence scores of the neighbors as *their* KRank score. In order to measure the aggregate score of a given paper's neighbors, we simply use the mean score of the neighbors.

## 5.2 Definition

With this prescription, we only need to define two parameters. First, we need to define how much of the final KRank score comes from the initial classifier estimate and how much comes from the neighbors' KRank score. Second, we need to define whether we want to weight directed citations differently, i.e. whether we want to weight incoming and outgoing citations differently. Define the number of neighbors of a given paper $p_i$ as $n_i$. We can then define the KRank score $R(q,p_i)$ for a given candidate paper $p_i$, given a query paper $q$ and an existing scoring function $\bar{S}(q,p_j)\forall j \in \{1,...,n\}$ (we use the overbar to emphasize that the function $S$ is an *estimate* for a score):

$$\mathrm{R}(q,p_i,\gamma,\alpha) \equiv (1-\gamma)\cdot\bar{S}(q,p_i) + \gamma\frac{1}{n_i}\left(\alpha \sum_{\{j|\,\mathrm{cites}(p_i,p_j)\}} R(q,p_j,\gamma,\alpha) + (1-\alpha)\sum_{\{j|\,\mathrm{cites}(p_j,p_i)\}} R(q,p_j,\gamma,\alpha)\right) \tag{4}$$

In the case that a given paper has no neighbors, we use $p_i$ itself as its own neighbor in the second term (of course, this simply amounts to setting the KRank value of $p_i$ equal to $\bar{S}(q,p_i)$). The above formula is easy to interpret. The most important parameter, $\gamma$, denotes the "damping" of KRank: it measures how much of the KRank score comes from the paper's initial scoring estimate, and how much comes from the information from its neighbors. If $\gamma = 0$, $K(q,p_i) = \bar{S}(q,p_i)$. As we increase $\gamma$, we place more and more importance on the neighbors. The parameter $\alpha$ measures how we weight outgoing citations with respect to incoming citations. A default value would be $\alpha = \frac{1}{2}$, in which case they are weighted equally ($\alpha > \frac{1}{2}$ weights outgoing citations stronger, and vice versa). For $\alpha = \frac{1}{2}$, the formula simplifies to a perhaps even more intuitive form:

$$\mathrm{R}(q,p_i,\gamma,\alpha) \equiv (1-\gamma)\cdot\bar{S}(q,p_i) + \gamma\cdot \underset{\{j|\,\mathrm{neighbors}(p_i,p_j)\}}{\mathrm{mean}} R(q,p_j,\gamma,\alpha) \tag{5}$$

Note that this is a universal approach that does not depend on the nature of the scoring function. In our case, $\bar{S}(q,p_i)$ is simply given by the classifier output, but any roughly linear scoring estimate will work[4]. Note also that the recursive prescription of the KRank score makes it a truly global measure: albeit damped exponentially by the factor $\gamma$, the neighbors of the neighbors of our candidate paper

[3] Our system's top candidate papers included the *Building a Large Annotated Corpus of English: The Penn Treebank*, *Structure-Sharing in Lexical Representation* and *The Mathematics of Statistical Machine Translation: Parameter Estimation*, all seminal papers in their fields.

[4] Assuming some underlying ground truth score that increases linearly with confidence, we take the linear mean of all neighbors. This algorithm will not work if one, say, exponentiates the score. Then we would have to take the geometric mean, etc.

(and in fact every paper part of the connected component of our candidate paper) also contribute to the KRank score of a candidate's KRank score.

### 5.3 Relationship to PageRank

After deriving our scheme 4, we found that it is closely related to topic-sensitive PageRank, a variant of PageRank that includes a prior belief on the importance of nodes [11]. There are two main differences between topic sensitive PageRank and our algorithm. First, we allow for distinction between incoming citations and outgoing citations, and treat them both as bidirectional edges, but potentially with different respective weights. Second, the intuition for our algorithm is not given by the random surfer model, as in PageRank [8]. Rather, we simply want to weight each candidate by the average relevancy of its neighbors. In topic-sensitive PageRank, the second term in equation 4 would not be a mean, but rather a sum of the scores of the neighbors, inversely weighted by their degrees. One reason to motivate our difference (using the mean) is the undirected nature of the edges in the citation graph. We consider both incoming and outgoing citations to transfer relevancy. However, it is known [12] that PageRank on undirected graphs — due to the inverse weighting by the number of outgoing edges of the score of a given node in the recursion — simply closely approximates the degree of a node. Thus, had we used the PageRank recursion formula in equation 4, we would have simply distorted our results towards favoring nodes with a higher number of total citations. However, the number of outgoing citations is not necessarily relevant, and the number of ingoing citations is already captured as a feature in the original score. Thus, this would have only added spurious noise to our ranking. We thus do not adopt the PageRank formulation and rather keep the mean of the neighbors. In order to confirm this intuition, we implemented PageRank in addition to our KRank algorithm, and ran our pipeline algorithm with both the PageRank and the KRank implementation on a development set and with the optimal value of $\gamma$. While both methods outperformed the case without reranking by about 1.5%, KRank performed even better than PageRank by about 0.5%, confirming our choice.

### 5.4 Implementation

In practice, the recursive nature of the expression for $R(q, p_i)$ means that we need to use an iterative algorithm (or a global linear algebra solver) to find the KRank scores of all the papers. We use an iterative approach, in which we simply apply the recursion 4 to all papers in the data set, and then repeat the procedure until convergence. The algorith stops once either the solution converges (measured by the relative $L_\infty$ (max-difference) norm over all candidate paper scores between successive iterations decreasing below a given threshold — in our case $10^{-4}$), or once a maximum number of iterations is reached — 100 in our case[5].

The reason we have to append the KRank scoring to the original algorithm — instead of attempting to make it its own feature — is that the fundamental assumption of KRank is that we already have some more or less accurate estimate of the score of a paper. Only with this "source" term does it make sense to refine a given paper's scores using its neighbors. Moreover, KRank is a fundamentally global operation: it relies on simultaneous knowledge of the initial scores for all papers, and thus cannot be computed locally for each individual paper.

### 5.5 Parameters

The only requirement for implementing KRank is finding optimal parameters for $\gamma$ and $\alpha$. Let us first consider $\gamma$. With the expression 4, one can easily see the tradeoff. If we make $\gamma$ very small, we use almost no information about the neighbors, and KRank will not improve the original scores. If we make $\gamma$ too close to 1, we will use almost only the information about the neighbors of our candidate paper, but not the information about the candidate itself. We expect there to be some nontrivial optimal value for $\gamma \in [0,1]$. Similarly for $\alpha$, we use information primarily from outgoing citations if $\alpha$ is close to 1, and vice versa if $\alpha$ is close to 0. Intuitively, whether a paper cites a relevant paper, or whether it is cited by a relevant paper gives approximately equal confidence about the paper itself.

## 6 Evaluation

To evaluate our algorithm we use it to rank all known papers by relevancy to each query $q$ in a set of test queries. We then caculate the Mean Average Precision (MAP) using just the top 100 relevant papers for each $q$. We limit calculation to the top 100 to save time. Since the MAP of the top 100 papers gives a lower bound on the true MAP, and also provides an accurate estimate this is justified.

### 6.1 Mean Average Precision

We'll use mean average precision (MAP) to measure the success of our algorithm. To motivate MAP consider that our algorithm ranks a set of papers by relevance. This means our metric should capture how many actually relevant articles appear near the top of our ranking. MAP works by averaging the precision in across a set of windows growing down from the top of the results ranking, so it fits our bill. An additional benefit of using MAP is that Bethard and Jurafsky[3] use it on the same dataset we're using. This allows direct comparisons between ours and their results.

To rigorously define MAP we'll first define *precision*. For a query paper $q$ let $\bar{R}(q)$ denote a set papers ostensibly relevant to $q$ and $R(q)$ denote the set papers actually relevant to $q$. Then the *precision* of $\bar{R}(q)$ with respect to $R(q)$ is:

$$\text{Precision}(R(q), \bar{R}(q)) = \frac{|\bar{R}(q) \bigcap R(q)|}{|\bar{R}(q)|}$$

To extend *precision* to apply to a ranking rather than a fixed set of relevant papers consider *average precision*. Let $R(q)$ be as before and

[5] we played with this number and found that convergence is almost always reached well before this point and that both convergence parameters do not affect results measurably

let $\bar{K}(q)$ be list of papers in increasing order of ostensible relevance to $q$. Then *average precision* $\bar{K}(q)$ with respect to $R(q)$ is:

$$\text{AveragePrecision}(R(q), \bar{K}(q)) = \frac{\sum_{i:\bar{K}[i]\in R(q)} \text{Precision}(R(q), \bar{K}[1:i])}{|R(q)|}$$

Now since *average precision* only applies to the suggestions from a single query it makes sense to extend it this into a metric for an entire test set. Denoting $Q$ as a test set of queries, *Mean average precision* (MAP) is:

$$\text{MeanAveragePrecision} = \frac{\sum_{q\in Q} \text{AveragePrecision}(R(q), \bar{K}(q))}{|Q|}$$

# 7 Results

## 7.1 LDA Topics

We were at the topics that Latent Dirichlet Allocation was able to extract. The table below illustrates some examples of LDA's success.

| Qualitative Topic | Top Words |
|---|---|
| Pronoun Resolution | pronoun resolution anaphora antecedent pronouns anaphoric definite anaphor zero subject |
| Sentiment Analysis | sentiment positive negative polarity reviews review product classification al., citation |
| French | de la le des les : en un du une |
| Parts of Speech | noun nouns phrase phrases compound head adjectives adjective proper modifier |
| Social | you social tweets al., twitter people users my your conversation |
| Speech Recognition | prosodic pitch phrase boundary speech accent prosody tone (np boundaries |

## 7.2 Without KRank

To measure the importance of features in our classifier stage (ie our algorithm without KRank) we left them out one at a time and observed the impact on performance. Table 2a characterizes performance on a $4,000$ paper subset of our data.

## 7.3 With KRank

We evaluated KRank by adding the KRank reranking scheme as described in section 5 to the recommendation pipeline. We measure performance as a function of $\gamma$ and compare to the baseline without KRank. The results are given in figure 2b.

## 7.4 Error Analysis

In our evaluation of the recommendation system, we noted that the standard deviation of the MAP score was on the same order as the MAP score itself ($\sim 0.3$). Upon looking at the actual score distribution, we found that our system performs exceedingly well on some examples (average precision $\sim 0.7-1.0$) and rather poorly on other examples (average precision $\sim 0.05$). We therefore give some examples of good and poor performance to attempt to explain where and why our system didn't perform well.

### 7.4.1 Base System (Without KRank)

Without KRank our system seemed to do either do well or badly depending on how well textual features could rank candidates.

If the query abstract (the query's source of textual similarity) was too broad then our base system performed badly. For example the abstract of *Lexical Semantic Techniques for Corpus Analysis* was very broad talking about linguistics, corpora, statistical analyses, semantic relationships, etc. . Comparing the top suggested papers to the query's abstract they are all resonable suggestions based on textual similarity. We observe that the context-based features pushed more popular candidates to the top of the recommendations, leaving the first correctly identified reference in 55th place presumably because of its lack of popularity. We ultimately got an average precision of 0.052.

On the other hand if the query abstract was narrowly define, having clear differentiating themes, then our algorithm performed well. We see this in the candidate ranking for *Using Syntax to Improve Word Alignment Precision for Syntax-Based Machine Translation*. It's abstract described improving syntax-based machine translation by improving word-alignments. This clear, narrow abstract greatly improved the signal from text-based features. Our system's top suggestions were *The Alignment Template Approach to Statistical Machine Translation*, *A Systematic Comparison of Various Statistical Alignment Models*, and *The Mathematics of Statistical Machine Translation: Parameter Estimation*. These are exactly the true references for our query paper, giving us a perfect average precision of 1!

These results suggest that textual similarity is our 'true' source of relevance, and context-based features merely refine the signal from the text-based features. If the text-based features can't discriminate well enough, then the context-based features can only do so much. This explains why our average precision scores tend to either be good or not.

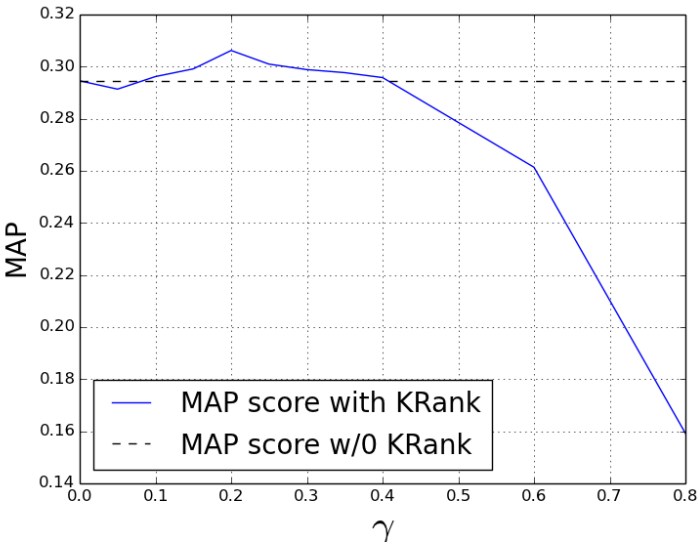
### 7.4.2 With KRank

KRank acts as another context-based feature, accounting for a measure of more "global relevance" (i.e. measuring the relevance of the paper locally, but also the relevance of the citation graph vicinity of the paper). We observe that this acts as other context-based features. If text-based features give little signal KRank has virtually no impact. However, when text-based features give good signal, KRank leverages this signal to rerank suggestions based on global relevance, resulting in significant improvements in average precision.

An example of this marked improvment occured in the rankings for the query paper *Unsupervised phonemic Chinese word segmentation using Adaptor Grammars*. Here the abstract describes exactly the purview of the paper, namely exploring application

| Features Held Out | MAP | MAP decrease |
|---|---|---|
| Paper Popularity | 0.211 | 0.079 |
| TFIDF | 0.237 | 0.053 |
| Previously Cited Paper | 0.242 | 0.048 |
| Previously Cited | 0.261 | 0.029 |
| LDA Topics | 0.261 | 0.029 |
| Previously Cited Venue | 0.263 | 0.027 |
| Authorship Overlap | 0.264 | 0.026 |
| Institution Overlap | 0.266 | 0.024 |
| Recency | 0.269 | 0.021 |
| Recency Indicator | 0.274 | 0.016 |
| Author Popularity | 0.275 | 0.015 |
| Previously Collaborated | 0.279 | 0.011 |
| Previously Cited | 0.285 | 0.005 |
| Venue Popularity | 0.290 | 0 |
| None | 0.290 | 0 |

(a) Importance of Features




(b) MAP vs. $\gamma$

Figure 2: Performance Evaluation. In (a) we show the relative importance of the features of our classifier-based model by training models and holding out each feature one at a time. We see that the important features come from all feature categories. Recency and Paper Popularity are important context-based features. LDA Topics are an important textual feature. Previously Cited Paper is an imporant behaviour-based feature. This makes sense, since each category of feature only covers one facet of the relationship between citer and citee. In (b) we show the MAP score achieved using the classifier with all features and KRank reranking. It is plotted vs. the $\gamma$ parameter from equation 4. The default value of $\alpha = 0.5$ is used. Note that the scores improve significantly above the baseline when using KRank (indicated by the value for $\gamma = 0$ and by the dashed line). The best value occurs for $\gamma = 0.2$ at 31.0%, as compared to a baseline of 29.5%. We ran this analysis several times for different dev sets, and obtained similar results consistently, indicating that our improvement is significant.

of state-of-art adaptor grammars to unsupervised segmentation of Mandarin. Without KRank the true reference *Unsupervised word segmentation for Sesotho using Adaptor Grammars* is ranked second behind *PCFG Models of Linguistic Tree Representations*, a locally quite popular paper with some textual similarities to the query. KRank, howevever, reflects the fact that *Unsupervised word segmentation for Sesotho using Adaptor Grammars* occurs in a cluster of other highly ranked papers whereas *PCFG Models of Linguistic Tree Representations* is not. KRank permutes the top two suggestions, which improves the average precision from 0.5 to 1.

### 7.5 Conclusion

It seems that text-based features were central source of signal in rankings, with other features refining suggestions. When our system's text features performed poorly, we experienced low average precision. However if papers has clear textual themes our system did a good job of ranking candidates.

Our novel addition, KRank, refined rankings by accounting for citation graph context. It very rarely significantly decreased score, and occasionally significantly increased it, overall improving our performance. The significant increase usually occured when local relevance of recommendations differed significantly from global relevance (i.e. relevance including citation neighborhood information), and KRank corrected for the too-strong impact of local relevance. It outperformed PageRank in this capacity, likely because PageRank ignores incoming references or because it penalizes articles for having many references.

## References

[1] Matthew E Falagas, Eleni I Pitsouni, George A Malietzis, and Georgios Pappas. Comparison of pubmed, scopus, web of science, and google scholar: strengths and weaknesses. *The FASEB Journal*, 22(2):338–342, 2008.

[2] Andreas Doms and Michael Schroeder. Semantic search with gopubmed. In *Semantic techniques for the web*, pages 309–342. Springer, 2009.

[3] Steven Bethard and Dan Jurafsky. Who should i cite: learning literature search models from citation behavior. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 609–618. ACM, 2010.

[4] Fattane Zarrinkalam and Mohsen Kahani. A new metric for measuring relatedness of scientific papers based on non-textual features. *Intelligent Information Management*, 4(4):99–107, 2012.

[5] Fattane Zarrinkalam and Mohsen Kahani. A multi-criteria hybrid citation recommendation system based on linked data. In *Computer and Knowledge Engineering (ICCKE), 2012 2nd International eConference on*, pages 283–288. IEEE, 2012.

[6] DragomirR. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. The acl anthology network corpus. *Language Resources and Evaluation*, pages 1–26, 2013.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Radim **R**ehů**r**ek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[11] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[12] Vince Grolmusz. A note on the pagerank of undirected graphs. *arXiv preprint arXiv:1205.1960*, 2012.