Classifying and Ranking Search Engine Results as Potential Sources of Plagiarism

Kyle Williams[‡], Hung-Hsuan Chen[†], C. Lee Giles[†‡]
[‡]Information Sciences and Technology, [†]Computer Science and Engineering
Pennsylvania State University, University Park, PA 16802, USA
kwilliams@psu.edu, hhchen@psu.edu, giles@ist.psu.edu

ABSTRACT

Source retrieval for plagiarism detection involves using a search engine to retrieve candidate sources of plagiarism for a given suspicious document so that more accurate comparisons can be made. An important consideration is that only documents that are likely to be sources of plagiarism should be retrieved so as to minimize the number of unnecessary comparisons made. A supervised strategy for source retrieval is described whereby search results are classified and ranked as potential sources of plagiarism without retrieving the search result documents and using only the information available at search time. The performance of the supervised method is compared to a baseline method and shown to improve precision by up to 3.28%, recall by up to 2.6% and the F_1 score by up to 3.37%. Furthermore, features are analyzed to determine which of them are most important for search result classification with features based on document and search result similarity appearing to be the most important.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.7.5 [**Document and Text Processing**]: Document Capture—*Document Analysis*; H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval

General Terms

Experimentation, Measurement, Performance

Keywords

Source retrieval; plagiarism detection; search result ranking; query generation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DocEng'14, September 16–19, 2014, Fort Collins, Colorado, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2949-1/14/09 ...$15.00.
http://dx.doi.org/10.1145/2644866.2644879.

1. INTRODUCTION

The advent of the Web has led to an exponential increase in the amount of information that is publicly available and accessible. This increase in information has had a number of benefits in important domains, such as healthcare, education, disaster management and community involvement. Search engines have become an important tool in dealing with the exponentially increasing amount of information available on the Web by allowing people to construct queries that describe their information needs and effectively retrieving search results that may satisfy those information needs. However, in addition to the many positive effects search engines have had, they have also made it increasingly easy to plagiarize information from the Web. For instance, a study conducted over the years 2002-2005 found that 36% of undergraduate college students admitted to plagiarizing information from the Web without proper citation [18] and a study in 2010 found that 1 in 3 American high school students admitted to plagiarizing from the Internet[1].

Given the negative impact that plagiarism has on education and society, a number of techniques have been developed for identifying cases of plagiarism [17]. Generally, the plagiarism detection problem is framed as:

Problem 1: *Given a suspicious document and a potential source document for plagiarism, find all areas of overlapping text, which may have been subjected to obfuscation.*

A number of approaches have been developed for addressing Problem 1. For instance, a method based on citation pattern matching has been developed for detecting plagiarism among scholarly documents [7] and many software tools have been developed for identifying plagiarism [17]

There is an inherent assumption in the definition of Problem 1 that potential source documents for plagiarism have been identified. For small collections of documents one might just assume that all documents in the collection are potential sources of plagiarism and perform a comparison between the suspicious document and each document in the collection. However, this approach is infeasible for all but the smallest collections. Another task in plagiarism detection that does not make this assumption is known as source retrieval. In source retrieval, the goal is to use a search engine to retrieve a subset of documents in a collection that are likely to be sources of plagiarism by constructing queries from the suspicious document that can be used to query the search engine. The source retrieval problem can be described as:

[1]http://charactercounts.org/programs/reportcard/2010/installment02_report-card_honesty-integrity.html

Problem 2: *Given a suspicious document and a search engine, use the search engine to retrieve candidate documents that may be sources of plagiarism.*

To address *Problem 2*, a function $\Phi(\cdot)$ must be designed such that, for a suspicious document D, $\Phi(D) \rightarrow \mathbb{Q}$, where $\mathbb{Q}$ is a set of queries that can be submitted to the search engine. Once the set of queries is submitted to the search engine, the search result documents can then be retrieved and more accurate matching performed in order to check if they are sources of plagiarism. One way of doing this is to download the results in the order that they are ranked by the search engine; however, there is no guarantee that the search engine ranking reflects the probability of a result being a source of plagiarism. Thus, documents that are not sources of plagiarism may be downloaded and unnecessary attempts at solving *Problem 1* may take place.

This paper describes a strategy for solving *Problem 2* that attempts to minimize the number of unnecessary comparisons made as described above. This is done by classifying each result returned by the queries in $\mathbb{Q}$ as either being a candidate source of plagiarism or not using only the information that is available at search time, i.e., without retrieving the documents themselves. Furthermore, attempts are made at ranking those results that are classified as being candidate sources of plagiarism in order to improve the order in which they are retrieved. In order to do this, various features are extracted and various methods for search engine result classification and ranking are analyzed and compared to a baseline method that achieved the highest F_1 score in the 2013 PAN Source Retrieval Task [23]. In summary, this work makes two main contributions:

- It presents a novel supervised source retrieval strategy for finding potential sources of plagiarism on the Web.
- It compares various methods for search engine result classification for source retrieval and evaluates the features used for this classification.

In making these contribution, the rest of this paper is structured as follows. We first begin by discussing work related to source retrieval in Section 2. Section 3 describes the supervised source retrieval strategy and algorithm and Section 4 describes the creation of a data set for source retrieval evaluation as well as set of features that can be used for supervised search result classification. Section 5 describes a set of supervised methods that were used for classifying and ranking search results. Section 6 then presents a set of experiments comparing these methods to a baseline and also provides an evaluation of the features used for classification. Lastly, conclusions and future work are discussed in Section 7.

2. RELATED WORK

To our knowledge, there has been no previous work on supervised classification of search results as potential sources of plagiarism; however, there has been previous research on classifying search results in other ways. For instance, it has been noted that the order of results returned by search engines is based on relevance scores alone and may not take the topic of documents into consideration [35]. As a result, some efforts have been made to group search engine results into categories or clusters. For instance, [33] describe a probabilistic ranking model that includes document categories and [26] describe an approach to presenting search results in user defined hierarchies by classifying documents into concepts from an ontology.

The *Learning to rank* machine learning framework has been used to learn a function for re-ranking the search results returned by a search engine for a specific query [16]. For instance, the ranking may be based on user clickthrough data [13] which can be used to infer which results are relevant for a specific queries. Learning to rank has been applied in a number of domains, such as learning a ranking of sponsored search results [34] and for ranking answers in Q&A systems [27]. [32] argue that many approaches for learning to rank in information retrieval attempt to optimize scores such as accuracy and ROCArea, rather than the mean average precision (MAP) score often used to evaluate information retrieval systems. They thus propose a method for optimizing MAP and show how it performs better than other methods when it comes to maximizing MAP. Supervised ensemble ranking methods have been used to combine the ranking outputs from multiple ranking algorithms; however, it has been noted that the drawback of this approach is that the learned weights are query independent and thus semi-supervised solutions for ensemble ranking have been proposed [12]. In [6], a set of candidate answers in a bibliographic Q&A system are re-ranked after being retrieved. This is similar to this work where search results are retrieved, classified and then re-ranked.

The Query By Document (QBD) search methodology is similar to the source retrieval problem. In QBD, a user submits a whole document as a query and the QBD system returns a list of documents ranked using some pre-determined similarity function [29]. Several systems have been designed for similar document retrieval. For instance, systems have been developed that make use of standard search engines and different query formulation strategies [21, 9, 3]. In each case, queries are formulated from the input document and submitted to a search engine and the results are ranked based on some similarity metric. QBD systems have been designed for use with full documents, passages of text [14], books [20] and blog posts [31]. The main difference between QBD and the source retrieval strategy described in this paper is that QBD systems return documents that are similar to a query document whereas we specifically focus on classifying search results as being potential sources of plagiarism for a given suspicious document. Furthermore, QBD systems usually perform ranking based on the full text similarity of documents whereas we instead focus on classifying search results based only on information that is available at search time without retrieving the actual documents.

Commercial systems for plagiarism detection generally focus on finding the overlap among texts; however, some systems do consult external sources. For instance, the Essay Verification Engine[2] automatically constructs queries from an input document and conducts a Web-based search to retrieve similar documents.

The work most similar to that described in this paper would be the approaches used in the Source Retrieval task at PAN 2013 [8]. In that task, the approach that achieved the highest F_1 score used a naive method for determining whether or not a search engine result is a source of plagia-

[2]http://www.canexus.com/eve/abouteve.shtml

rism based on a measure of similarity between the snippet of text associated with the result and the original suspicious document. Other approaches were similar in that they often compared the similarity of some part of the search result snippet with the original suspicious document and used that to determine whether to download a document or not [23].

This work differs from existing work in that it specifically attempts to classify and rank the search results returned by a search engine as being sources of plagiarism using a supervised approach and using only features that are available at search time.

3. SOURCE RETRIEVAL STRATEGY

Having introduced the problem and described related work, we now describe an online source retrieval strategy that can be used for real interaction with a search engine for retrieving potential sources of plagiarism. This strategy is similar to existing source retrieval strategies [23]; however, the main difference is that it makes use of a supervised method for determining which results are potential candidate sources of retrieval. Algorithm 1 shows the source retrieval strategy employed in this study.

Algorithm 1 General overview of source retrieval strategy

```
 1: procedure SourceRetrieval(doc)
 2:     paragraphs ← SplitIntoParagraphs(doc)
 3:     for all p ∈ paragraphs do
 4:         p ← Preprocess(p)
 5:         queries ← ExtractQueries(p)
 6:         for i = 0 → n do            ▷ n is the top n queries
 7:             results ← SubmitQueries(queries[i])
 8:         end for
 9:         results ← ClassifyAndRank(results)
10:         for all result ∈ results do
11:             if result is True then
12:                 if PreviousSource(result) = false then
13:                     source ← Download(result)
14:                     if IsSource(source) then
15:                         print source
16:                         PreviousSource ← source
17:                         break
18:                     end if
19:                 end if
20:             end if
21:         end for
22:     end for
23: end procedure
```

First the input document is split into paragraphs made up of sentences (line 2). Then the text of each paragraph is preprocessed to remove stop words, etc., (line 4) and queries are extracted from the paragraph using the process described in Section 4.2.1 (line 5). The top n queries are submitted to the search engine and results are returned for each query resulting in a set of search results per paragraph, which are then classified as potential sources of plagiarism and possibly ranked (line 9; described later). The intuition behind submitting multiple queries and combining their search results before classifying them as sources of plagiarism is that the probability of the union of the search results of all queries containing a source of plagiarism is at least as high as the probability of the results for a single query containing a source of plagiarism and likely higher.

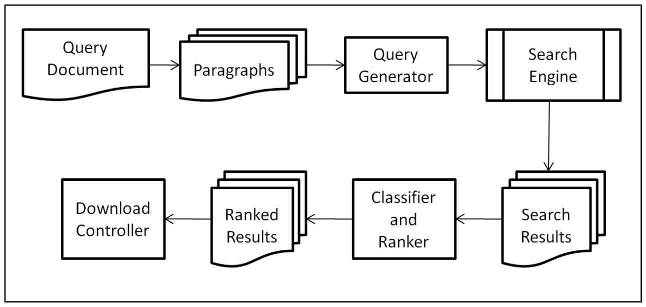

Figure 1: Flow chart representation of source retrieval strategy

If a result is classified as a source of plagiarism (line 11) and if the result has previously not been downloaded and identified as a source of plagiarism (based on the URL; line 12) then it is retrieved (line 13). The reason for checking whether or not a result has previously been downloaded and identified as a potential source of plagiarism is to prevent redundant retrieval. When a document is retrieved, an Oracle (described in the next section) is consulted to determine if it is a source of plagiarism (line 14). If it is, then the result is added to the set of previous successful results (line 16) and processing for the paragraph stops (line 17), otherwise the loop continues and the next result is processed. The reason for stopping the loop after a positive result has been retrieved is based on the previous intuition that a paragraph is likely to be plagiarized from a single source and thus retrieving additional results will not improve performance.

Figure 1 provides a high level visualization of Algorithm 1 showing the steps involving query generation and result submission, result classification and ranking, and the controlling of downloads.

4. DATA AND FEATURES

4.1 Original Dataset

The data used in this study is based on the training data provided as part of the Source Retrieval task at PAN 2013 [8]. The way in which this data was originally collected is described in detail in [25] and involved crowdsourcing whereby people were asked to write plagiarized documents on specific topics. Two batches of plagiarized documents were created in this way: for *Batch 1* people were allowed to freely search the ChatNoir Search Engine[3] [24] to find sources for plagiarism whereas for *Batch 2* people were provided with 20 search results on a specific topic and asked to use those results for plagiarism. The data used in this study is based on the *Batch 2*, which was the only data available at the time this study was conducted.

A total of 40 documents from *Batch 2* were provided as training data for the Source Retrieval task at PAN 2013 and those 40 documents were used in this study. Table 1 shows the descriptive statistics for the number of words in these documents.

As can be seen from these statistics, the documents are relatively long (i.e. around 5-10 pages). Twenty documents

[3]http://chatnoir.webis.de/

Table 1: Descriptive statistics for number of words

Min	Max	Mode	Median	Mean
1873	7335	5383	5104	5152

were randomly sampled from this collection for classification and model training, 10 were used for validation and the remaining 10 documents were retained for testing. From these documents, a search result dataset was created.

4.2 Search Result Dataset

A search result dataset was constructed by extracting a set of queries from each document, submitting them to the ChatNoir search engine, downloading the results, and labelling each result as being a source of plagiarism or not.

4.2.1 Query Generation

To generate queries from a given document, the text of each document was first partitioned in paragraphs, with each paragraph containing 5 sentences that were tagged by the Stanford Tagger [28]. The words in each paragraph extracted this way were then POS-tagged using the Stanford POS Tagger and, following [15], only nouns, verbs and adjectives were retained while all words were filtered out. Queries were then constructed from the remaining nouns, verbs and adjectives by combining every non-overlapping sequence of 10 words, which resulted in a set of queries for each paragraph, of which only the first three were retained while the others were discarded. The intuition behind this is that a paragraph is likely to be plagiarized from a single source and that the first 3 queries from a paragraph are likely to sufficiently capture enough information about the paragraph.

4.2.2 Query Submission and Result Labeling

The queries generated for each paragraph were submitted to the ChatNoir search engine [24] and the first three results returned by each query were retrieved. This number was selected since it was empirically found to lead to good results [30]. ChatNoir contains an *Oracle* that can be consulted to determine whether or not a search result is a source of plagiarism for a given suspicious document ID. The Oracle is a service provided by the PAN plagiarism detection competition organizers that provides a binary output of whether a document is a source of plagiarism for a given suspicious document. This label is not provided by prediction but is stated as a fact and is meant to be used for evaluating retrieval methods. We use the Oracle to label each search result appropriately.

4.2.3 Training Data

In total 2737 queries were constructed from the 20 training documents and 5740 search results were returned when querying the search engine with these queries with each search result being labelled based on the feedback from the ChatNoir Oracle. Of the 5740 search results, 4240 were labelled as negative (i.e., not sources of plagiarism) and the remaining 1500 being labelled as sources of plagiarism. Thus, the data was heavily skewed towards negative samples, which made up 73.87% of the data.

4.2.4 Validation Data

In total 1331 queries were constructed from the 10 validation documents and 2940 search results were returned and labelled when querying the search engine. 2365 of these were negative samples and the remaining 575 were positive. Negative samples made up 80.44% of the validation data.

4.2.5 Testing Dataset

A total of 1303 queries were constructed from the 10 testing documents and 2991 search results were returned. 2174 of these were negative samples and the remaining 817 were positive. Thus, the distribution of the testing data follows a similar ratio to the training data with negative samples making up 72.68% of the data.

4.3 Features

For each labelled search result the following features were extracted (some of which were provided by the ChatNoir search engine). All of these features are available at search result time and do not require the search result to be retrieved, which allows for classification to be performed as the search results become available.

1. **Readability.** The readability of the result document as measured by the Flesh-Kincaid grade level formula [22] (ChatNoir).
2. **Weight.** A weight assigned to the result by the search engine (ChatNoir).
3. **Proximity.** A proximity factor [24] (ChatNoir).
4. **PageRank.** The PageRank of the result (ChatNoir).
5. **BM25.** The BM25 score of the result (ChatNoir).
6. **Sentences.** The number of sentences in the result (ChatNoir).
7. **Words.** The number of words in the result (ChatNoir).
8. **Characters.** The number of characters in the result (ChatNoir).
9. **Syllables.** The number of syllables in the result (ChatNoir).
10. **Rank.** The rank of the result, i.e. the rank at which it appeared in the search results.
11. **Document-snippet 5-gram Intersection.** The set of 5-grams from the suspicious document are extracted as well as the set of 5 grams from each search result snippet, where the snippet is the small sample of text that appears under each search result. A document-snippet 5-gram intersection score is then calculated as:

$$Sim(s,d) = S(s) \cap S(d), \tag{1}$$

where s is the snippet, d is the suspicious document and $S(\cdot)$ is a set of 5-grams.

12. **Snippet-document Cosine Similarity.** The cosine similarity between the snippet and the suspicious document, which is given by:

$$Cosine(s,d) = \cos(\theta) = \frac{V_s \cdot V_d}{||V_s||||V_d||}, \tag{2}$$

where $V.$ is a term vector.

13. **Title-document Cosine Similarity.** The cosine similarity between the result title and the suspicious document (Eq. 2).

14. **Query-snippet Cosine Similarity.** The cosine similarity between the query and the snippet (Eq. 2).

15. **Query-title Cosine Similarity.** The cosine similarity between the query and the result title (Eq. 2) [13].

16. **Title length.** The number of words in the result title.

17. **Wikipedia source.** Boolean value for whether or not the source was a Wikipedia article (based on the existence of the word "Wikipedia in title).

18. **#Nouns.** Number of nouns in the title as tagged by the Stanford POS Tagger [28].

19. **#Verbs.** Number of verbs in the title as tagged by the Stanford POS Tagger.

20. **#Adjectives** Number of adjectives in the title as tagged by the Stanford POS Tagger.

5. SEARCH RESULT CLASSIFICATION AND RANKING

A number of different supervised classification methods using the features and data described in the previous section were compared. These supervised methods were also compared to a baseline method. The baseline method achieved the highest F_1 score in the source retrieval task at PAN 2013 [23]. The supervised methods include: linear discriminant analysis, logistic regression, random forests, AdaBoosting with decision trees and ensembles of these classifiers. Ranking involves determining the order in which to retrieve results that have been classified as being candidate sources of plagiarism.

5.1 Baseline

The baseline method classifies each search result as being a potential source of plagiarism based on the document-snippet 5-gram intersection (feature 11). Documents for which $Sim(s, d) = S(s) \cap S(d) \geq 5$, are classified as candidate sources of plagiarism and documents are ranked by $Sim(s, d)$ in descending order. This method achieved the highest F_1 score in the source retrieval task at PAN 2013 [23].

5.2 Supervised Methods

5.2.1 Linear Discriminant Analysis

A Linear Discriminant Analysis (LDA) classifier attempts to find a linear combination of features for classification. For LDA, two different ranking cases are considered for LDA: no ranking where positive results are retrieved in the order they were classified; and *ProbRank* where results are ranked by their probability of being a source of plagiarism. This probability value is output by the classifier.

5.2.2 Logistic Regression

Logistic regression is a form of binary classification where the classification decision is made based on:

$$p(\vec{x}) = \frac{1}{1 + e^{\beta_0 + \beta_1 \vec{x}}}, \tag{3}$$

where $Y = 1$ is predicted when $p(\vec{x}) \geq 0.5$ and $Y = 0$ otherwise. In this study, we use L-1 regularization when learning the logistic regression model and, as with LDA, both no ranking and the *ProbRank* ranking method are both considered.

5.2.3 Random Forest

A random forest is an ensemble classifier made up of a set of decision trees [1]. Each tree is built with a bootstrap sample from the dataset and splitting in the decision tree is based on a random subset of the features rather than the full feature set [5]. In this study, the number of trees in the ensemble is set to 10 since this was empirically found to perform well and the number of features randomly selected is equal to $\sqrt{n\_features}$. At each level in the decision trees, variables are selected for splitting with the Gini index being used as a splitting criterion. The Gini index is defined as follows:

$$I_G(i) = \sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2, \tag{4}$$

where K is the number of classes and p_j is the proportion of instances belonging to class j in node i. If a node i is pure (only contains one type of class), then $I_G(i) = 0$. The Gini index is used in decision tree learning for selecting the variable to split on at each node, with the split that leads to the largest reduction in the Gini index being selected.

The other parameters for the decision trees in the random forest were found using a grid search over the training set with the validation set used for testing. The parameters were:

- Maximum tree depth, $d = \text{None}, 1, 2, 3, 4$
- Minimum samples to split a node, $s = 1, 2, 3, 4$
- Minimum samples per leaf, $l = 1, 2, 3, 4$

Once the grid search was performed, the parameters that resulted in the highest F_1 score on the validation set were used for training the final model.

As with other methods, two ranking options were considered for the random forest: the baseline ranking method (i.e., no ranking) and *ProbRank*.

5.2.4 AdaBoosting with Decision Trees

AdaBoost is another ensemble method that iteratively fits modified versions of data to a set of weak classifiers. For a set of weak classifiers $G_m(x), m = 1, 2, ...M$, the prediction, $G(x)$, for the target value of a sample is based on a weighted majority vote as [10]:

$$G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m G_m(x)), \tag{5}$$

where α_m weighs the contribution of each classifier with more accurate classifiers being assigned more weight. During training, the data is modified at each iteration. Initially, the weight of each sample $w_1, w_2, ..., w_n$ is set to $\frac{1}{N}$; however, with each iteration these weights are modified with the weight being increased for samples incorrectly classified and decreased for samples correctly classified. This has the effect of making each successive classifier focus on the incorrectly classified examples since the error for a classifier is calculated based on these weights [10]. At the end of a training iteration, the weight α_m of a classifier $G_m(x)$ is based on this error rate.

The weak learners used are decision trees and the same parameters as in the random forest were used. Similarly, the same ranking options (no ranking and *ProbRank*) were used.

5.2.5 Majority Voting Ensemble

We also experiment with a voting ensemble of the four supervised and baseline classifiers described above. A necessary condition for an ensemble of classifiers to perform better than the individual classifiers is that the individual classifiers are accurate and diverse [4]. In this case, the ensemble can reduce the risk of selecting a bad classifier for a given problem, reduce the impact of local optima, and expand the number of possible hypothesis representations [4]. In this study, we use a simple voting ensemble. In a voting ensemble, each classifier in the ensemble individually classifies a result and then casts a vote, which may or may not be weighted. The final decision as to the class of a result can then be based on the majority vote. We define the majority vote $M(x)$ among classifiers as:

$$M(x) = \sum_{i=1}^{n} w_i C_i(x), \tag{6}$$

where n is the number of classifiers in the ensemble, w_i is the weight assigned to the i-th classifier and $C_i(x)$ is the classification produced by the i-th classifier. When $M(x) > 0$, the weighted majority vote is positive and thus result is classified as a source of plagiarism. Similarity, when $M(x) < 0$, the result is classified as a non-source of plagiarism. In this study, we include the top $n = 3$ and $n = 5$ classifiers in ensembles based on their F_1 score (Equation 9) and all classifiers are weighted equally, i.e. $w_i = 1, i \in 1, 2, ...n$. Given the fact that an odd number of classifiers are used, $M(x)$ is guaranteed to be non-zero since the output of each classifier is either 1 (source of plagiarism) or -1 (non-source of plagiarism).

6. EXPERIMENTS

6.1 Experiment Methodology

The source retrieval strategy as described in Algorithm 1 was executed on the test set without classification and ranking (line 9), i.e., all results were retrieved for every query, and an interaction log was generated. The interaction log recorded each document name, paragraph number and results for each query. This allowed for different classification and ranking strategies to be compared without needing to re-submit the queries to the search engine. Performance was measured using precision, recall and the F_1 score, which are defined as follows:

$$Precision = \frac{tp}{tp + fp}, \tag{7}$$

$$Recall = \frac{tp}{tp + fn}, \tag{8}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{9}$$

where tp, fp and fn refer to true positives, false positives and false negatives respectively.

Precision measures the proportion of results that were retrieved that were actually sources of plagiarism and thus measures how good the algorithm is at correctly identifying true sources of plagiarism. Recall measures the proportion of the total number of plagiarized results that were retrieved when the known set of plagiarized results is known. To calculate recall, the set of all retrieved URLs was maintained and once all retrieval had completed this set was compared to the set of URLs that were known to be sources of plagiarism.

There is a tradeoff between precision and recall since high precision can be achieved by only retrieving documents for which there is high confidence that they may be sources of plagiarism, though this comes at the cost of recall. Similarly, high recall can be achieved by retrieving a large set of documents with low precision. The tradeoff between these two metrics is given by the F_1 score, which is the harmonic mean of the two measures.

From the perspective of plagiarism detection, it could be argued that recall is more important than precision since one may not mind examining a few extra documents in order to increase the chances of retrieving a source of plagiarism. However, at a large scale such as that of the Web, it is important to maintain good precision so that the number of documents that need to be analyzed in detail does not become prohibitive.

6.2 Data Sampling

As discussed, the training data was imbalanced with negative samples making up 73.87% of the data and it has been noted that most learning algorithms expect an equal class distribution and do not work well on imbalanced data [11]. Over sampling the minority class has successfully been used to address the imbalanced data problem and the SMOTE method for oversampling [2] is used in this study. The SMOTE method creates synthetic or artificial examples of the minority class based on existing samples. For each sample x_i, the K nearest neighbors are identified and one of those nearest neighbors $\hat{x_i}$ is randomly selected. The difference between x_i and $\hat{x_i}$ is then multiplied by a random number $r \in [0, 1]$ and this value is added to x_i to create a new point that falls on the line segment joining x_i and $\hat{x_i}$ [11]:

$$x_{new} = x_i + (\hat{x_i} - x_i) \times r. \tag{10}$$

In this study, we set the number of nearest neighbors to consider $k = 3$ and increase the number of positive training samples by 200%. Since the SMOTE method randomly selects nearest neighbors, we train 5 models and the results reported are the average of the 5 models.

6.3 Results

Experimental results are shown for 3 cases: when no ranking is used; when the probabilistic outputs of the classifiers are used for ranking; and when the ensemble method is used.

6.3.1 No Ranking

Table 2 shows the performance when no ranking is used for the supervised methods. In this case, the results are retrieved in the order that they were classified as being sources of plagiarism, which is based on the ordering produced by

Table 2: Precision, recall and the F_1 score for the baseline method and different supervised methods. No ranking of results is used, i.e. they are retrieved in the order they were classified.

Method	Precision	Recall	F_1 Score
Baseline	0.3735	0.8543	0.5198
LDA	**0.3894**	**0.8803**	**0.5399**
Logistic	0.3848	0.8629	0.5322
Random Forests (RF)	0.3625	0.8725	0.5122
AdaBoost	0.3811	0.8414	0.5246

Table 3: Precision, recall and the F_1 score for the baseline and different supervised methods. The search results were ranked by the probabilistic output of the classifiers.

Method	Precision	Recall	F_1 Score
Baseline	0.3735	0.8543	0.5198
LDA+ProbRank	**0.4063**	**0.8681**	**0.5535**
Logistic+ProbRank	0.4019	0.8553	0.5469
RF+ProbRank	0.3833	0.8651	0.5311
AdaBoost+ProbRank	0.4018	0.8367	0.5429

the search engine. For the baseline method, the results are ranked and ordered by the value of their snippet-document 5-gram intersection (feature 11).

As can be seen from Table 2, the highest precision is achieved by the LDA classifier, which is about 0.5% higher than the second highest precision achieved by the Logistic Regression classifier and about 1.6% higher than the precision achieved by the baseline. The highest recall is also achieved by the LDA classifier, which beats the baseline method by 2.6%. In fact, all of the supervised methods, except the AdaBoost classifier achieve higher recall than the baseline method. Similarly, all classifiers except the random forest classifier achieve higher precision than the baseline method. The highest F_1 score, which measures the tradeoff between precision and recall, was achieved by the LDA classifier and was just above 2% higher than the baseline. In fact, all of the supervised method except random forests achieve a better F_1 score than the baseline method. This comparison shows that using supervised methods to classify results as potential sources of plagiarism leads to an improvement in performance over the baseline even though the baseline makes use of an implicit ranking method based on the snippet-document similarity. The next experiment repeats this experiment with a ranking based on the probabilistic outputs of the classifiers.

6.3.2 Ranking by Probabilistic Output of Classifiers

Table 3 shows the performance when ProbRank is used. With *ProbRank*, the probabilistic outputs of the classifiers are used to infer a ranking or ordering of the search results for each paragraph, with results being ranked in terms of their probability of being a source of plagiarism. Once again, the baseline results are ranked and ordered by the value of their snippet-document 5-gram intersection (feature 11) for the baseline method.

As can be seen from Table 3, the use of ProbRank leads to an improvement in the performance of all of the supervised methods. The highest precision is achieved by LDA, and represents an almost 2% improvement over LDA without ProbRank and an improvement of 3.28% over the baseline. However, this improvement in precision comes at the cost of recall, which drops by over 1% compared to LDA with no ProbRank, though it still performs better than the baseline method. The F_1 score is higher with ProbRank than it is without and is an improvement on the baseline of 3.37%. The same pattern is observed for all supervised classifiers when ProbRank is used: ProbRank leads to an improvement in precision at the cost of recall, though the final F_1 measure is higher. Furthermore, all supervised methods outperform the baseline when ProbRank is used. From these results it can be argued that ProbRank is useful for improving the overall performance of the source retrieval strategy since it leads to a relatively large improvement in precision. Furthermore, ProbRank is a relatively simple ranking strategy and thus it could be possible to improve the results further with more advanced ranking strategies.

Table 4: Precision, recall and the F_1 score for the baseline and ensemble classifiers.

Method	Precision	Recall	F_1 Score
Baseline	0.3735	0.8543	0.5198
Ensemble-Top3	**0.3874**	0.8681	0.5357
Ensemble-Top5	0.3868	**0.8825**	**0.5379**

6.3.3 Voting Ensemble

Table 4 shows the performance of the majority voting ensemble classifiers and the baseline method. The majority voting ensembles are built with the top 3 and 5 performing classifiers as measured by their F_1 score without ranking.

As can be seen from Table 4, the use of the ensembles leads to an improvement in precision and recall over the baseline method. Furthermore, the ensemble consisting of all 5 classifiers leads to a slight improvement in the recall achieved by any classifier individually both with and without ProbRank. However, this comes at the cost of a decrease in precision. The performance of the ensemble consisting of all 5 classifiers performs similarly overall to the LDA classifier without ranking though not better than LDA with ProbRank. Given that the ensemble classifier does not lead to a large improvement in performance suggests that the classifiers are not sufficiently diverse to benefit from being combined in an ensemble.

6.3.4 Discussion

Overall it was found that the supervised classification of search results leads to an improvement in performance compared to the baseline method. Classifying search results without applying any ranking leads to similar precision while leading to an improvement in recall. Applying ProbRank to those results in general led to an improvement in precision, though at a slight cost in recall. It could be argued that recall is more important than precision for source retrieval and that the cost of missing a true source of plagiarism exceeds the cost of mistakenly retrieving a false source. The difference in recall between the baseline method and best performing supervised method was 2.6%. In the testing set, a total of 817 results were sources of plagiarism and a 2.6% increase in recall translates into potentially retrieving 21 additional sources of plagiarism while also improving precision. Given the importance of plagiarism detection, an increase in recall of only a few percent can be considered significant since it increases the chances of identifying plagiarism. Furthermore, at large scale, such as on the Web, a small increase in recall may translate into a significant increase in the number of sources of plagiarism retrieved.

6.4 Feature Analysis

Feature analysis was performed to gain insight into which features are important for source retrieval. This analysis provides insight into which of these features may be important not only in classifying search results, but also in understanding what plagiarizers may consider when choosing from which documents to plagiarize. This insight can be of practical use in constraining the plagiarism detection search space.

LDA was the best performing model; however, since LDA performs dimensionality reduction, its output is difficult to interpret. Thus, feature analysis is performed based on the random forest model where the importance of each feature is estimated based on the depth at which it occurs in the decision trees. This calculation is done using a built in method for calculating feature importance in the *scikit-learn* machine learning toolkit [19]. The feature importances are averaged for the 5 random forest models that were trained with different synthetic data generated by the SMOTE algorithm and are shown in ranked order in Table 5.

Table 5: Importance of different features in the random forest

Rank	No.	Feature	Importance
1	Doc-snippet intersection	11	0.39
2	Title-doc cosine	13	0.16
3	Wikipedia source	17	0.09
4	Snippet-doc cosine	12	0.07
5	#Adjectives	20	0.07
6	Proximity	3	0.06
7	Query-snippet cosine	14	0.03
8	Syllables	9	0.02
9	Sentences	6	0.01
10	BM25	5	0.01
11	Words	7	0.01
12	Title length	16	0.01
13	Query-title cosine	15	0.01
14	Characters	8	0.01
15	Weight	2	0.01
16	Readability	1	0.01
17	Rank	10	0.00
18	#Nouns	18	0.00
19	#Verbs	19	0.00
20	PageRank	4	0.00

An interesting observation from Table 5 is that the most important feature in the random forest is the exact same feature as used in the baseline method. This feature, which measures the intersection between the 5-grams in the suspicious document and the snippet contributes the largest amount to the final classification of the samples. The cosine similarity between the title of a result and the suspicious document is also a relatively important feature, suggesting that the titles of plagiarized sources may be strongly related to whether or not it is used as a source of plagiarism. This is intuitive since the title of a document is likely to be the first thing a user considers in judging whether a Web page is relevant to their query or not. The fact that the Wikipedia feature is the third most important feature suggests that whether or not a page is a Wikipedia page may have an impact on whether or not it is used as a source of plagiarism. This is intuitive since Wikipedia provides a general and easily accessible description of many topics and is often ranked highly in many public search engines. Interestingly, the number of adjectives in a search result title is a relatively important feature and is ranked much higher than the number of nouns. One possible reason for this is that nouns provide high level descriptions of the concepts of documents whereas adjectives help to better refine those concepts, which can be useful in deciding which among several documents on the same high level topic.

Other insights can be gained from the less important features. For instance, the BM25 ranking method used by the ChatNoir search engine (rank 10, feature 5) and where among the top 3 results a result is ranked (rank 17, feature 10) do not seem to be important features. This finding supports the hypothesis in the introduction that the order of results returned by a search engine does not necessarily reflect the probability of them being sources of plagiarism. Similarly, the properties of the result document in terms of length, readability, etc., do not seem to be important for classification which seems to mostly rely on similarity-based features.

Overall, this analysis provides some insight into which features may be important to improve the performance of the supervised methods and that can be used to inform the design of new features, i.e. the similarity between a result snippet and the suspicious document (rank 1 & rank 4) and the relationship between a result title and the suspicious document. Given these findings, it may be useful to design new similarity features that can be used to better improve performance.

7. CONCLUSIONS

Source retrieval involves using a search engine to retrieve potential sources of plagiarism for a given suspicious document and can be considered as a first step in a plagiarism detection pipeline. In this study, we investigated the use of a supervised source retrieval strategy for classifying search engine results as candidate sources of plagiarism using only information available at search time. For a given suspicious document, queries were generated automatically and were used to query a search engine for plagiarism sources. Using this method, a search result dataset was created with a set of features available at search time. Several different supervised methods and ranking options were compared to a baseline method for classifying and ranking these results. The performance of the best performing supervised methods were shown to improve precision by up to 3.28%, recall by up to 2.6% and the F_1 score by up to 3.37% compared to the baseline method.

An analysis of features showed that the feature used in the baseline was in fact the most important feature for supervised classification followed by the cosine similarity between the title of a result and the suspicious document itself. Interestingly, the search engine ranking of the results did not seem to be important feature for classification, thereby suggesting that retrieving search results in the ordering produced by a search engine is not a good strategy for source retrieval and plagiarism detection.

A large number of the features used in this study were provided by the search engine; however, it generally cannot be assumed that a search engine will return a set of features that can be used for classification. Thus, an important consideration for future work would be the creation of new features for classification. Furthermore, as is currently the case, these features would need to be derivable without retrieving the search results so as to allow for real-time results classification.

Generally, the focus in Web search is on precision since it is desirable that relevant search results appear on the first page. However, we argue that recall is more important for plagiarism detection since even a small improvement in recall is significant. This is especially the case at Web scale since a small increase in recall could potentially translate into a significant number of additional sources of plagiarism being retrieved. However, it is still important to maintain precision so as to reduce the number of unnecessary comparisons made. The supervised method described in this paper is able to achieve both better precision and recall than the baseline method thus satisfying both of these goals.

Future work seeks to investigate how we can improve the supervised methods by investigating additional query generation strategies and new features for supervised classification.

Acknowledgments

We gratefully acknowledge partial support by the National Science Foundation under Grant No. 1143921.

8. REFERENCES

[1] L. Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.

[2] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[3] A. Dasdan, P. D'Alberto, S. Kolay, and C. Drome. Automatic retrieval of similar content using search engine query interface. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, pages 701–710, 2009.

[4] T. G. Dietterich. Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.

[5] W. Fan. On the optimality of probability estimation by random decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 336–341, 2004.

[6] D. Feng, D. Ravichandran, and E. Hovy. Mining and re-ranking for answering biographical queries on the web. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1283–1288, 2006.

[7] B. Gipp and N. Meuschke. Citation pattern matching algorithms for citation-based plagiarism detection. In *Proceedings of the 11th ACM symposium on Document engineering - DocEng '11*, pages 249–258, 2011.

[8] T. Gollub, M. Potthast, A. Beyer, M. Busse, F. Rangel, P. Rosso, E. Stamatatos, and B. Stein. Recent Trends in Digital Text Forensics and Its Evaluation Plagiarism Detection, Author Identification, and Author Profiling. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 282–302, 2013.

[9] V. Govindaraju and K. Ramanathan. Similar Document Search and Recommendation. *Journal of Emerging Technologies in Web Intelligence*, 4(1):84–93, 2012.

[10] T. Hastie, R. J. Tibshirani, and J. J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[11] H. He and E. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[12] S. Hoi and R. Jin. Semi-Supervised Ensemble Ranking. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 634–639, 2008.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, pages 133–142, 2002.

[14] C.-J. Lee and W. B. Croft. Generating queries from user-selected text. *Proceedings of the 4th Information Interaction in Context Symposium*, pages 100–109, 2012.

[15] F. Liu, D. Pennell, F. Liu, and Y. Liu. Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 620–628, 2009.

[16] T.-Y. Liu. *Learning to Rank for Information Retrieval.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[17] H. Maurer, C. Media, F. Kappe, and B. Zaka. Plagiarism - A Survey. *Journal of Universal Computer Science*, 12(8):1050–1084, 2006.

[18] D. L. Mccabe. Cheating among college and university students : A North American perspective. *International Journal for Educational Integrity*, 1(1):1–11, 2004.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapear, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] M. Pera and Y. Ng. Brek12: A book recommender for k-12 users. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1037–1038, 2012.

[21] A. Pereira and N. Ziviani. Retrieving similar documents from the web. *Journal of Web Engineering*, 2(4):247–261, 2004.

[22] M. Potthast, T. Gollub, M. Hagen, J. Graß egger, J. Kiesel, M. Michel, A. Oberländer, M. Tippmann, A. Barrón-cedeño, P. Gupta, P. Rosso, and B. Stein.

Overview of the 4th International Competition on Plagiarism Detection. pages 17–20, 2012.

[23] M. Potthast, M. Hagen, T. Gollub, M. Tippmann, J. Kiesel, P. Rosso, E. Stamatatos, and B. Stein. Overview of the 5th International Competition on Plagiarism Detection. In *CLEF 2013 Evaluation Labs and Workshop Working Notes Papers*, 2013.

[24] M. Potthast, M. Hagen, B. Stein, J. Graß egger, M. Michel, M. Tippmann, and C. Welsch. ChatNoir: A Search Engine for the ClueWeb09 Corpus. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, page 1004, 2012.

[25] M. Potthast, M. Hagen, M. Völske, and B. Stein. Crowdsourcing Interaction Logs to Understand Text Reuse from the Web. In *51st Annual Meeting of the Association of Computational Linguistics (ACL 13)*, pages 1212–1221, 2013.

[26] A. Singh and K. Nakata. Hierarchical Classification of Web Search Results Using Personalized Ontologies Background Document Organization. In *Proceedings of the 3rd International Conference on Universal Access in Human-Computer Interaction*, 2005.

[27] M. Surdeanu and M. Ciaramita. Learning to rank answers on large online QA collections. In *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 719–727, 2008.

[28] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, volume 1, pages 173–180, 2003.

[29] L. Weng, Z. Li, R. Cai, Y. Zhang, Y. Zhou, L. T. Yang, and L. Zhang. Query by document via a decomposition-based two-level retrieval approach. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, pages 505–514, 2011.

[30] K. Williams, H. Chen, S. Choudhury, and C. Giles. Unsupervised Ranking for Plagiarism Source Retrieval - Notebook for PAN at CLEF 2013. In *CLEF 2013 Evaluation Labs and Workshop Working Notes Papers*, 2013.

[31] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 34–43, 2009.

[32] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, 2007.

[33] Q. Zhang, Y. Zhang, H. Yu, and X. Huang. Efficient partial-duplicate detection based on sequence matching. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*, page 675, 2010.

[34] Y. Zhu, G. Wang, J. Yang, D. Wang, J. Yan, J. Hu, and Z. Chen. Optimizing search engine revenue in sponsored search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '09*, pages 588–596, 2009.

[35] Z. Zhu, M. Levene, and I. Cox. Ranking Classes of Search Engine Results. In *KDIR*, pages 294–301, 2010.