Application of Machine Learning Techniques to Paper-Author Identification Problem

Yitao Li

Project Report for Course TCSS 702

Institute of Technology
University of Washington, Tacoma

Master of Science in Computing and Software Systems

Committee: Dr. Senjuti Basu Roy, Institute of Technology – Chair
Dr. Ling Ding, Institute of Technology
Dr. Ankur M. Teredesai, Institute of Technology

Last Modified: 08:19:27 UTC, 10/07/2013

Contents

Abstract

Author name ambiguity is known to be one of the major challenges faced by academic search engines covering large numbers of publications: while several different authors can have similar or even identical names, each author may also publish using different names, both of which potentially causing publications to be associated with incorrect authors. Fortunately, with informative features such as title and keywords of each paper, names of conferences or journals related to papers, and name(s) and affiliation(s) of each author, as well as some ground-truth consisting of a set of known correct paper-author associations and a set of known incorrect ones, the problem of attributing publications to their correct authors can be addressed with a variety of machine learning approaches, and the purpose of this project is to compare the performance of these approaches.

1 Introduction

This project investigates the utility of machine learning algorithms in inferring authorship of publications based on a set of noisy associations between each publication and its possible authors. The list of possible authors for each publication, which is a superset of the actual author(s) of each publication, will be provided as an input to the algorithm. The list may include multiple authors with similar or even identical names, and the goal of the algorithm is to infer the actual author(s) of each publication among its list of possible authors. Other inputs to the algorithm will include, but are not limited to the name and affiliation of each author, and the title, keyword, and conference or journal related to each publication. Although these attributes will be helpful for disambiguation, they are not guaranteed to be present for all authors and publications in the dataset.

The ambiguity of author names is a commonly encountered problem in academic search engines. For instance, when a user queries for publications using the name of an author as keyword, it is possible that publications by different authors with similar or identical names are returned as results, and the user may need to further refine the search results with additional attributes such as the affiliation of the author. Oftentimes, however, a simple filtering based on additional user-supplied criteria may not be sufficient or even possible, as metadata describing the publications can originate from heterogeneous sources and do not necessarily associate each publication with a full set of searchable attributes of its author(s), and the data may also contain errors caused by similarity in author names.

Challenges arising from noisy and missing data of academic search engines may be addressed by machine learning methods that make best use of the information available along with a limited set of known correct paper-author attributions and a set of incorrect ones (i.e., the ground-truth) as references for classifying correct and incorrect paper-author associations. In this project, a set of relevant attributes and derived attributes will be selected from a noisy dataset

of papers, authors, and paper-author associations and will be normalized appropriately to be used as features for training and testing support vector machine classifiers and k-nearest neighbor classifiers. A test set (which is, by definition, disjoint from the training dataset) containing paper-author associations with known correctness labels will also be used to compare the performance of both the SVM method and the KNN method.

While the SVM method will simply output binary results predicting whether a publication is written by a given author, an additional goal of this project is to rank the list of possible publications for each author by relevance (i.e., publications truly written by an author should precede all other publications). When using the SVM method, a possible way to obtain a ranking is the following: suppose a given author A has a list of possible publications P, and a classifier has predicted true instance(s) $T \subseteq P$ and false instance(s) $F \subseteq P$ (with T, F satisfying $T \cup F = P$, $T \cap F = \emptyset$), then with a metric R_A measuring the relevance of a publication p to author A, one can order all $t \in T$ by $R_A(t)$ in descending order, followed by all $f \in F$ ordered by $R_A(f)$ in descending order. It is empirically observed that greater mean average precision can be obtained by ranking T and F separately by R_A, as described above, rather than simply ranking all $p \in P$ by $R_A(p)$. Details about this observation is included in section 4 of this proposal.

Unlike the SVM method, the KNN method provides result containing not only the classification label of a given data point, but also a density estimation of different classes of data points near the given data point. With such density estimations, one can rank a list of publications possibly written by a given author by likelihood of authorship rather than simply by binary classification labels.

Although the SVM method and the KNN method are two well-known supervised machine learning techniques, it is worthwhile to note that the problem of ranking a list of publications according to their likelihood of being written by an author can also be addressed by unsupervised machine learning techniques, especially clustering algorithms. For example, a list of publications possibly written by an author can be clustered according to their subject areas, and then additional information such as author affiliation may be used to determine which cluster(s) of publications are likely to be composed by the author in question. Alternatively, a set of associations between authors and publications can be divided into clusters and additional information from the conferences or journals of each publication may be used to infer clusters that are likely to contain false associations. For completeness, this project will also include an investigation of different clustering techniques and their effectiveness in retrieving most relevant publications from a given author among all publications from authors with similar or identical names.

2 Background

2.1 Support Vector Machine

Support vector machine (SVM), a supervised learning method, is a form of maximum margin classifier that utilizes a fixed feature space transformation defined in term of a kernel function to obtain representations of data points in kernel space. When applied to binary classification problems, a support vector machine chooses among all possible hyperplanes separating the two classes in kernel space a hyperplane that maximizes its margins to each class as the decision boundary for classification [1]. The original SVM algorithm using only linear kernel function was first invented by Vladimir N. Vapnik in 1963 [21] and a version of the SVM algorithm applying non-linear kernel functions (the "kernel-trick") was suggested by Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik in 1992 [2]. Corinna Cortes and Vladimir N. Vapnik also proposed a dual formulation of the SVM problem using soft-margins in 1995 [5]. The soft-margin formulation allows an SVM classifier to wrongly classify some of the data points in training data while maximizing the margin of the decision boundary relative to all other points in the training data and is useful for classifying data points that are not completely linearly separable in kernel space. The soft-margin formulation is also used in the SVM classifier of this project as a way to avoid over-fitting of decision boundaries caused by outliers and noisy or mislabeled data points in training set.

2.2 K-Nearest Neighbors

To classify a data point using the K-nearest-neighbor rule, one uses a distance measure to find K points in the training dataset that are closest to the given data point, assign that data point to the class having the largest number of representatives among the K closest training data points, and break ties arbitrarily [1]. The K-nearest-neighbor method can be viewed as a form of lazy learning, where the training data itself, rather than any decision boundary or classification rule derived from the training data, is used for classification. It should be noted that while the K-nearest-neighbor method is simple to implement and classifies each data point based on estimations of densities of different classes near that point, it has two drawbacks: first of all, one must be able to query the entire training data when classifying new data points; in addition, the outcome of the classifier, which is determined by a distance measure, can also be influenced by the scales of different features, hence the way in which features are normalized may have significant impact on the results of the KNN classifier.

2.3 Term frequency-inverse document frequency

Term frequency-inverse document frequency (Tf-idf) is a measure of the importance of a term's appearance in one document relative to its appearance in a collection of documents [14]. For each term t, document d, term frequency refers

to any statistic that reflects a term's occurrence in a document. Possible choices include a boolean value representing whether or not a term occurs in the document, the number of occurrence of a term in a document, a length-normalized frequency $tf(t,d) = \frac{freq(t,d)}{\max_{w \in d} freq(w,d)}$ for term t and document d [16], which prevents upward bias for long texts with repeating terms, and an augmented normalized term frequency $tf(t,d) = 0.5 + 0.5\frac{freq(t,d)}{\max_{w \in d} freq(w,d)}$ [20] that halves the amount of variation in $tf(t,d)$ caused by differences in $freq(t,d)$ of all possible terms t within a given document d. The augmented term frequency is used in this project. The inverse-document frequency for a term t within a collection of documents D is computed as $idf(t,D) = \log \frac{|D|}{|\{d \in D, t \in d\}|}$, which is simply the self-information associated with the outcome that a uniformly randomly chosen document $d \in D$ contains term t, hence $idf(t,D)$ assigns higher weights to terms that have rare occurrence in the collection of documents D. Therefore, the term-frequency-inverse document frequency of a term t in a document $d \in D$, which is computed as $tf\text{-}idf(t,d) = tf(t,d) \cdot idf(t,D)$, will be a quantity that gives more weight to terms occurring frequently in a particular document d but occurring rarely in all other documents $d' \in D \setminus \{d\}$, and may be considered as a measure of importance of the event $t \in d$ given $d \in D$.

3 Related Work

In the past, researchers have proposed a variety of methods to address the problem of authorship attribution. The particular problem addressed by this project can be related to the classical example of disputed authorships described in both [17] and [18]. Bayesian methods for authorship analysis are described in [18]. Techniques relying on statistical representations of authors' writing styles are studied in [13]. However, unlike methods based on full-text features, the algorithms implemented in this project attempt to apply machine learning techniques to resolve ambiguity in authorship using only existing conference or journal information of all publications, title and keyword features of each publication, and names and affiliations of all possible authors of each publication. The author attribution problem is also discussed in [12], in which a clustering based approach is used for separating works from multiple authors with similar names. The problem of determining whether authors of a pair of publications refer to the same person, a problem closely connected to the problem addressed in this project, is addressed in [8], in which a solution based on supervised machine learning using first order features of papers and authors is proposed.

In addition, the problem of authorship attribution based on noisy data of academic publications is also closely related to numerous data cleaning challenges such as attribute extraction, fuzzy matching of keywords, fuzzy grouping of records, and matching records via string transformations. Examples of effective and practical data cleaning techniques can be found in [6], [7], and [11].

4 Technical Proposal

4.1 Methodology

The paper-author dataset used for both training and testing the SVM model and KNN model in this project will minimally include the one listed in [9], a real-world dataset describing a set of publications and their possible lists of authors. Figure 1 shows a diagram representing the schema of this dataset

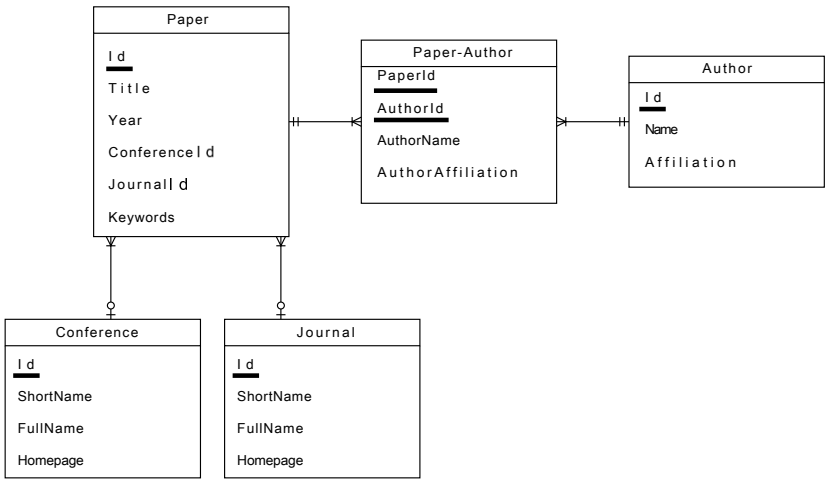


Figure 1: ER-diagram of the Kaggle dataset

This dataset will be loaded into a relational database management system, in which necessary primary and secondary indexes will also be constructed to enable efficient queries. Most relational database management systems index all primary keys by default. In addition, the `Paper-Author` table should be indexed by `PaperId` and by `AuthorId` so that all possible authors of a publication and all possible publications by an author can be retrieved efficiently. Other secondary indexes may be created as needed (e.g., a secondary index on `ConferenceId` should be created if it is necessary to query for all publications of a conference).

In this project, a significant amount of effort will be devoted to selecting and normalizing most relevant features from the dataset. Many features that are useful indicators of authorship can be derived from the dataset: for instance, for each publication p and candidate author A, A may be considered as a "non-ambiguous" candidate author of p if no other possible coauthors of p has the same last name as A. Hence all publications with no ambiguous author occurring in the paper-author dataset can be found by the following SQL query:

```
SELECT DISTINCT ‘Id‘ FROM (
  SELECT ‘PaperId‘, COUNT(DISTINCT ‘AuthorId‘) AS ‘cnt‘ FROM
  ‘PaperAuthor‘ GROUP BY ‘PaperId‘, LOWER(SUBSTR(‘AuthorName‘,
  LENGTH(‘AuthorName‘) + 1 - LOCATE(’ ’, REVERSE(‘AuthorName‘))))
```

```
) AS ‘PaperInQuestion‘ WHERE ‘cnt‘ = 1;
```

.

Although an author A is a non-ambiguous candidate author of publication p still does not imply that A is in fact an author of p, it is empirically observed from the dataset that among 1000 randomly sampled publications, there exists only 1 publication such that an unambiguous candidate author among its list of all possible authors is in fact not an author. This observation suggests a non-ambiguous candidate author is highly likely to be an author, and a reliable way to collect keywords associated with an author A in a noisy paper-author dataset is to only use keywords from A's affiliation and keywords from papers for which A is a non-ambiguous author. Keywords collected from a paper are simply words from the `title` attribute and from the `keywords` attribute of that paper, minus common stop words. All keywords should be normalized by their tf-idf scores. For each matching keyword between an author and a publication, the match score for that keyword can be computed as the product of its tf-idf score with the corpus being all non-ambiguous publications by the author and its tf-idf score with the corpus being a stratified random sampling of all publications that is representative of the publications in the entire dataset. Other weighing schemes are also possible, and their effect may be explored as part of this project.

A preliminary result is obtained by the author of this proposal on the Kaggle dataset using the following steps: for each publication p and candidate author A, features used for training and testing a support vector machine are

1. 16 highest tf-idf scores of matching keywords of p and A
2. 8 highest tf-idf scores of matching keywords of affiliation of A and affiliations of authors from training set with confirmed publications in conference or journal of p
3. sum of tf-idf scores of matching keywords of affiliation of A and affiliations of other candidate authors of p
4. number of candidate authors of p
5. number of candidate authors of p having the same last name as A
6. number of candidate papers of A
7. the total number of keywords of p and the total number of keywords of A

By combining the predictions of the SVM with the sums of keyword matching scores between each publication and each author, the list of a given author's possible publications can be first grouped by the prediction of the SVM with predicted true instance preceding predicted false onces, and then ordered by the sum of matching scores in descending order within each group. The ranked result outputted by this model on the test set was then submitted to Kaggle [10], and was able to obtain a mean average precision of 0.91942, while ranking only by

total score of matching keywords had a mean average precision of 0.83397, which suggests that the classification results from the SVM are helpful in improving the mean average precision of the results. Both of the results above also set a good baseline for performance of all possible models that may be trained and tested in this project, and an important objective of this project is to select features and weighing schemes that may improve the mean average precision of the ranking. Although this project will mostly focus on using the SVM and KNN classifiers with appropriate features to obtain accurate classification results, alternative approaches such as the random-forest classifier, the Naïve Bayes classifier, and unsupervised learning algorithms will also be implemented, and their performance will be compared with the performance of the SVM and the KNN classifiers.

4.2 Evaluation Plan

The performance of all models will be measured by their mean average precision in ranking confirmed publications before deleted publications of a set of authors with known ground-truth labels. From an information retrieval point of view, each author, along with his or her list of possible publications, forms a query, and a publication will be considered as a relevant result for that query if and only if it is actually written by the author in question. The mean average precision measured in this project is therefore the arithmetic mean of the average precisions of such queries for a set of authors.

As part of this project, each model will be tested with a variety of parameter choices. For models having multiple parameters, standard parameter selection methods (e.g., grid search) will be performed and each combination of parameters will be evaluated using the k-fold cross-validation method on training datasets. The evaluation section of the final project report will include the mean average precisions of each model on both training and testing datasets under different parameter choices, generalization error of each model, as well as a comparison of optimal performance across different models.

5 Project Timeline

This project will take 2 quarters (20 weeks) to complete, as shown in the timeline below:

Week	Activity
1-3	Literature review and research on supervised and unsupervised classification techniques and statistical inference methods
4-5	Determine possible features to be extracted from input data, implement and test feature extraction and normalization procedures
6-7	Implement and test a SVM classifier
8-10	Compare classification performance of different kernels in the SVM classifier For each type of kernel function, use grid search method to select optimal combination of soft-margin penalty weight and kernel parameters
11-12	Implement and test a KNN classifier
13-14	Compare classification performance of different kernels in the KNN classifier, for each type of kernel, use grid search method to select optimal parameters
15-18	Implement alternative classifiers e.g., random forests and/or document retrieval methods and unsupervised machine learning approaches, and compare performance
18-19	Analyze possible trade-offs between speed and classification accuracy or mean average precision performance of both supervised and unsupervised approaches
20	Finalize project report and presentation of results and findings

6 Deliverables

Deliverables of this project includes a written report consisting of the following items:

1. Implementation details of the support vector machine classifier, the K-nearest-neighbor classifier, and unsupervised clustering methods that are used in this project [1]

2. Description of dataset(s) used for training and evaluating the performance of all classifiers

3. Procedures for extracting and normalizing relevant features from the input data

[1] The source tree is currently hosted at `https://github.com/yl790/source_code_for_course_TCSS_702`

4. Procedures for determining the optimal kernel functions and parameters of the support vector machine classifier and the K-nearest-neighbor classifier for a given training dataset

5. Analysis and performance evaluation of all algorithms implemented in this project

6. Conclusion and future work

along with all source code and detailed numerical results related to this project. A presentation of results and findings is also expected at the conclusion of this project.

7 Feature Selection

As part of the feature engineering procedure in this project, a similarity score is assigned to 2 lists of <keyword, tf-idf score> tuple(s) as the following: suppose $L = \{(w_1, s_1), \ldots, (w_m, s_m)\}$, $L' = \{(w'_1, s'_1), \ldots, (w'_n, s'_n)\}$, where $w_1, \ldots, w_m, w'_1, \ldots, w'_n$ are keywords outside of a list of common stop words in English, and $s_1, \ldots, s_m, s'_1, \ldots, s'_n$ are their respective tf-idf scores, then the similarity score of L, L' is computed as

$$\texttt{sim}(L, L') = \sum_{(i,j) \in (([1,m] \cap \mathbb{N}) \times ([1,n] \cap \mathbb{N})) : S(w_i, w'_j) = 1} s_i \cdot s'_j,$$

where S is a boolean function that decides whether 2 keywords can be considered as similar. For the purpose of this project, 2 keywords w, w' are considered similar if and only if the Levenshtein distance of w, w' (denoted as $d(w, w')$) less than or equal to 4 and the similarity measure $1 - d(w, w') / \max\{\texttt{len}(w), \texttt{len}(w')\} > 0.6$. In this project, necessary pre-processing steps such as removal of all whitespace and non-printable characters and conversion of keywords to all-lower-case format are applied before the similarity computation. For example, with the boolean function S defined by the above criteria, the keyword "physicist" would be considered as similar to the keyword "physics", the keyword "mathematics" would be considered as similar to the keyword "mathematician", whereas the keyword "signature" would not be considered as similar to the keyword "significant".

For each publication p and candidate author A, a set of keyword(s) (excluding a list of common stop words in English) related to A is defined as the set of keywords that occur in an "non-ambiguous" publication of A[2], and the following features are selected in this project for training and testing the classification models that attempt to decide whether A is actually an author of p:

1. 16 highest tf-idf scores of matching keywords of p and A

[2]Recall for each publication p and candidate author A, A may be considered as a "non-ambiguous" candidate author of p if no other possible coauthors of p has the same last name as A.

2. 8 highest tf-idf scores of matching keywords of affiliation of A and affiliations of authors from training set with confirmed publications in conference or journal of p
3. 4 highest number of overlapping "non-ambiguous" publications of A and those of other possible authors of p (as specified in the noisy paper-author association dataset)
4. number of keywords related to A
5. number of keywords occurring in p
6. number of keywords (excluding common stop words in English) in affiliation of A
7. number of keywords (excluding common stop words in English) in affiliation of other possible authors of p (as specified in the noisy paper-author association dataset)
8. similarity score of affiliation (if any) of A when compared with the affiliations of other possible authors of the given paper (as specified in the noisy paper-author association dataset)
9. sum of tf-idf scores of matching keywords of affiliation of A and affiliations of other candidate authors of p
10. number of candidate coauthors of p
11. number of candidate coauthors associated with p in the noisy paper-author dataset that have same last name and first name initial as these of A
12. number of candidate papers of A

8 Implementation Details

All source code and scripts related to this project are hosted at `https://github.com/yl790/source_code_for_course_TCSS_702`.

8.1 SVM Classifier

This project utilizes the C++ version of LIBSVM [4], an excellent open-source software for support vector classification, to train and test a SVM classifier for the paper-author disambiguation problem. Since all features are numerical in this project, each feature is normalized to have mean 0 and standard deviation 1 when training the SVM model, and the normalization constants obtained while training the model are also used when normalizing the testing data.

8.2 KNN Classifier

This project uses ANN [19], a library written in C++, for nearest neighbor computations. While the ANN library provides not only exact, but also approximate nearest neighbor functionalities, this project currently only uses the exact nearest neighbor functionality of the ANN library in the implementation of the KNN model. However, it should be noted that the approximate nearest neighbor method implemented in the ANN library is guaranteed to limit its relative error below a pre-specified upper bound, and it can be significantly more efficient than the exact KNN procedure when the total number of data points is large. Similar with the SVM classifier implementation, the KNN classifier also requires input features to be normalized first.

8.3 Naïve Bayes Classifier

Since Naïve Bayes classification is relatively simple, this project did not utilize any external library when implementing the Naïve Bayes classifier. With all feature being numerical in this project, the Naïve Bayes classifier simply assumes each feature in a true/false instance of the training data is a value sampled from a particular normal distribution conditioned on the label of that instance and independently from all other features (i.e. the Naïve Bayes assumption), and it uses all training data to produce the maximum-likelihood estimations of the mean and standard deviation of the conditional distribution of each feature given the class label. The posterior probabilities of a candidate author being or not being an author of a given publication are then computed based on estimated prior probabilities of both outcomes from the training data, feature values extracted from test data, and the estimated conditional distributions of each feature value based on the training data, and with ties broken arbitrarily, the outcome with the higher posterior probability is outputted as the classification result.

8.4 Random Forest

This project uses the C++ implementation of LIBRF [15] to build a random forest, which consists of multiple decision trees, each of which constructed with a randomly chosen sequence of features based on the training data. It should be noted that when using continuous numerical features, each feature will be internally discretized by LIBRF based on the range and estimated distribution of that feature and the resulting discrete variable will then be used as a feature for building decision trees. The classification result of an instance is obtained based on votes of all trees in the forest. For each test instance, the output of the decision trees (i.e. the votes) also tends to indicate the level of confidence that the model has on classifcation of that instance: for example, an instance in the test set is considered to be very likely to be a true instance if the vast majority of trees in the forest have classified it as a true instance. While the ratio of true votes to false votes from the trees in a random forest does not reflect any

probability value, it can still be used to sort a list of candidate publications possibly written by a given author, with most likely true instances preceding less likely true instances, and such sorted list tends to produce better mean average precision than lists produced based only on binary classification results (e.g., the SVM method that was implemented in this project).

8.5 Decision Tree with AdaBoost

In addition to using the C++ implementation of LIBRF [15] to generate decision trees based on random sequences of features, a separate procedure is also implemented in this project to combine such decision trees optimally into a single classification model using the adaptive boosting algorithm. Different from the random forest method, which simply uses all training data to construct each decision tree, the AdaBoost method constructs decision trees sequentially and adaptively assigns different weights to training data in each iteration. A consequence of the weight assignments implemented according to the AdaBoost algorithm is that each subsequent decision tree after the first one will be focusing more on correctly classifying instances that are misclassified by the previous trees.

8.6 Unsupervised Method

This project uses a simple version of the K-mean algorithm implemented in C [3] to divide the set of candidate publications of each author into 2 clusters based on the feature vectors associated with each publication (as described in section 7). Since distance computations can be affected by scaling of features, all feature vectors are required to be normalized first before being processed by the K-mean algorithm. The normalization procedure is same as the one used in the KNN and the SVM classifier implementations. With ties broken arbitrarily, each publication in the cluster having the highest average keyword matching score against keywords of the given author is considered as a publication that is actually written by that author. Publications in each of the 2 clusters are first sorted separately within their clusters by their keyword matching scores against keywords of the candidate author, and the 2 sorted lists are then combined, with the one having the highest average keyword matching score preceding the other one, and the combined list is then used as the final result, based on which the MAP value is computed.

9 Evaluations and Results

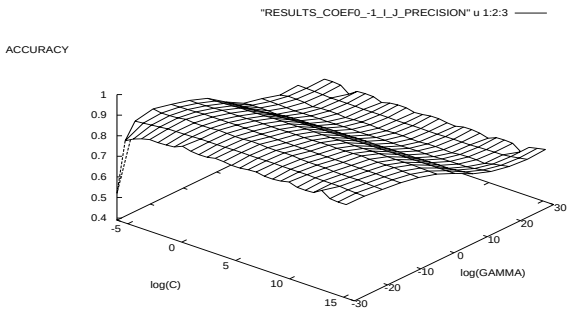
For the purpose of evaluation, all models implemented in this project are trained using a fixed training set that is randomly sampled from the Kaggle paper-author disambiguation data and tested with another randomly sampled set that is disjoint from the training set.

9.1 SVM Classifier

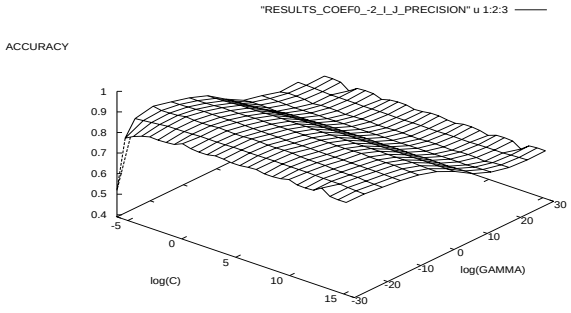
SVM classifiers using the sigmoid kernel and the Gaussian radial basis function (RBF) kernel are implemented in this project. For each type of kernel, a grid search is ran to estimate the optimal combination(s) of the kernel function parameters [3] and the soft-margin penalty weight (C) for that kernel. It should be noted that such combination is not necessarily unique, and is specific to the particular classification problem that the SVM classifier is trying to solve. Therefore such optimal parameters differ among various classification problems.

For the sigmoid kernel ($k(u, v) = \tanh(\gamma \cdot u^\intercal v + \texttt{coef0})$ where u, v are feature vectors), the following results are obtained:

With $\texttt{coef0} = -1$:



Similar results were obtained with $\texttt{coef0} = -2$:

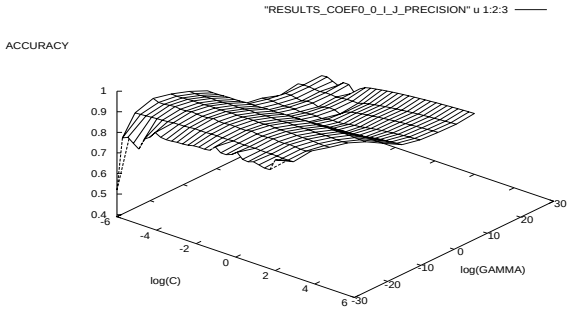


While both results do have slight numerical differences, they both appear to indicate that the optimal combination of the parameters (C, γ) occurs near

[3] namely, the γ for the RBF kernel and the γ and $\texttt{coef0}$ values for the sigmoid kernel

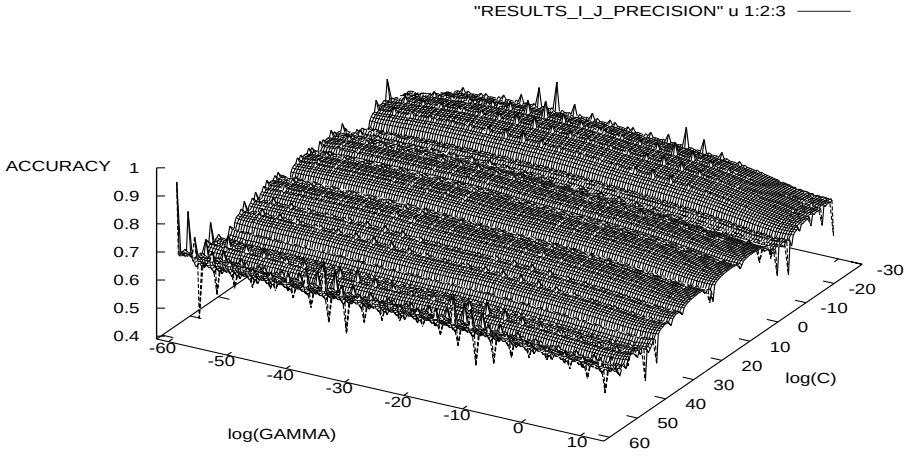
$(C, \gamma) = (2^{-5}, 2^{-20})$, and both choices of `coef0` leads to a classification accuracy of 0.9410.

Partial results were also obtained with `coef0` $= 0$ which reflects similar trends (but with minor numerical differences):



Unfortunately, because a more exhaustive search in parameter space is computationally expensive and also time-consuming for the sigmoid kernel, the search for optimal (C, γ) combination for other `coef0` values were not performed. It should be noted, however, such search is technically possible, and can be considered as part of the future work for this project.

A search in the parameter space for the RBF kernel is also performed:

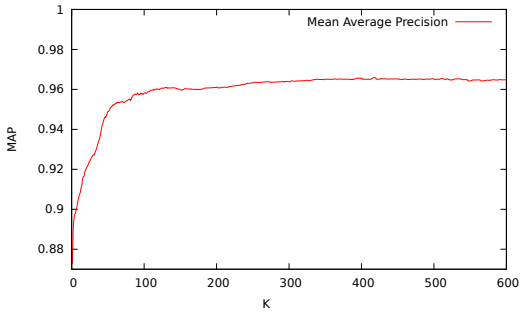


and the results indicate for the RBF kernel, the optimal parameter com-

bination appears to be near $(2^{-12}, 2^{-49})$, and with $(C, \gamma) = (2^{-12}, 2^{-49})$, the model achieves a classification accuracy of 0.9416.

9.2 KNN Classifier

The only tunable parameter in the KNN classifier is K, the number of nearest neighbors to examine in order to classify a data point. While bigger K value tends to result in better accuracy, an excessively large K value also incurs heavy computational costs, especially with non-trivial number of training examples. For the disambiguation problem in this project, $K = 60$ (or around 60) appears to provide a reasonable trade-off between classification accuracy and efficiency. The plot below shows the mean average precision obtained by the KNN classifier on a random test set using a fixed training set with a range of K values.



The plot also shows that little improvement in classification accuracy is obtained for $K = 100$ and beyond, and the maximum mean average precision obtained by the KNN model appears to be around 0.965.

9.3 Naïve Bayes Classifier

Compared to other classifiers implemented in this project, the Naïve Bayes classifier has the advantage of being simple and efficient in both the training and testing phases. Also unlike the SVM classifier and the KNN classifier, the Naïve Bayes classifier does not depend on any model with tunable parameters, hence does not require any search in parameter space. Another advantage of the Naïve Bayes classifier is that its output can be probability values, while the outputs of the SVM, KNN, and the random forest classifiers can only be discrete class labels.

On the test set that is randomly sampled from the Kaggle dataset, the Naïve Bayes classifier implemented in this project achieves a mean average precision of 0.9693.

9.4 Random Forest

Since the generation of random forest is not a deterministic process, one has to reduce the variance in the evaluation by repeating the random forest

generation process multiple times, each time with a different random seed, and take the average of the mean average precision values achieved by multiple random forests as an estimate of the overall performance for the random forest model. The following shows the average MAPs for random forests of size $10, 50, 100, 1000$, and 5000, where 10 random forest of each size are evaluated and their average MAP on the test set is reported:

RF size	Average MAP
10	0.951535
50	0.963124
100	0.963629
1000	0.964941
5000	0.969339

The results show that for the paper-author disambiguation problem, varying the size of the random forest did not drastically change the mean average precision achieved by the model.

9.5 Decision Tree with AdaBoost

When running the AdaBoost algorithm with decision trees, one can alter the maximum number of boosting iterations and the number of trees. The results below indicate that while increasing the maximum number of boosting iterations or increasing the number of trees generally causes the model to produce higher mean average precision value on the test set, such increase is not significant compared to the increased running time of the algorithm.

Num Trees	Max Num Iters	Avg MAP	Avg Run time
100	5	0.926360	4.888s
100	50	0.946053	8.561s
100	100	0.952624	13.246s
500	5	0.930669	17.362s
500	50	0.946848	35.577s
500	100	0.954124	55.373s
1000	5	0.939782	33.138s
1000	50	0.940362	69.642s
1000	100	0.945536	108.633s

9.6 Unsupervised Method

While it is intuitive to have $K = 2$ in the K-mean algorithm for the unsupervised method, one can also have other K values, with clusters of publications being first sorted in ascending order by their average keyword matching scores with a given author, and then publications within each cluster being sorted in ascending order by each of their keyword matching score with the author. The

following table contains results obtained for different K values and shows $K = 2$ tends to result in the highest mean average precision for the unsupervised algorithm. Another interesting observation from the results shown is that for the disambiguation problem addressed in this project, the unsupervised approach appears to have slightly outperformed the supervised ones.

K	MAP
1	0.968576
2	0.979918
3	0.966460
4	0.957897
5	0.952947
6	0.946445

However, since K-mean clustering may arrive at a local minimal rather than a global one and is also computationally intensive when the cardinality of the test dataset is large, the unsupervised approach using K-mean clustering may not be practical when each author has significantly more possible publications than the ones shown in the Kaggle dataset.

10 Acknowledgements

The author of this proposal is grateful for professor Senjuti Basu Roy, professor Ankur M. Teredesai, and his mentor Vani Mandava for numerous helpful discussions and generous guidance on the paper-author identification problem. The author would like to thank the ACM Special Interest Group on Knowledge Discovery and Data Mining for organizing the KDD CUP 2013 – Author-Paper Identification Challenge. The Author-Paper Identification Challenge has been a fantastic machine learning competition that has greatly motivated this project, and it has also provided a valuable opportunity for obtaining preliminary results that are included in this proposal. In addition, the author is thankful for professor Ankur M. Teredesai, professor Robert Friedman, as well as other reviewers of this proposal for their advice on academic writing. This project is supported in part by a student internship program at Microsoft Research. This project is facilitated through the use of advanced computational, storage, and networking infrastructure of the Hyak supercomputer system, supported in part by the University of Washington eScience Institute and the National Science Foundation, including grant #PHY-0922770.

11 References

[1] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[3] E. Brodsky. C source code implementing k-means clustering algorithm. `http://www.medphysics.wisc.edu/~ethan/kmeans/`, 2011.

[4] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] A. Arasu et al. Incorporating string transformations in record matching. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1231–1234, New York, NY, USA, 2008. ACM.

[7] A. Arasu et al. Experiences with using data cleaning technology for bing services. *IEEE Data Eng. Bull.*, 35(2):14–23, 2012.

[8] A. Culotta et al. Author disambiguation using error-driven machine learning with a ranking loss function, 2007.

[9] B Hamner et al. Kdd cup 2013 – author-paper identification challenge (track 1). `http://www.kaggle.com/c/kdd-cup-2013-author-paper-identification-challenge/data`, 2013.

[10] B Hamner et al. Kdd cup 2013 – author-paper identification challenge (track 1) submission page. `http://www.kaggle.com/c/kdd-cup-2013-author-paper-identification-challenge/submissions/attach`, 2013.

[11] S. Chaudhuri et al. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 327–338. VLDB Endowment, 2007.

[12] Y. Qian et al. Combining machine learning and human judgment in author disambiguation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 1241–1246, New York, NY, USA, 2011. ACM.

[13] D. Holmes and R. Forsyth. The federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, 10(2):111–127, 1995.

[14] K. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[15] B. N. Lee. Librf: C++ random forests library. `http://mtv.ece.ucsb.edu/benlee/librf.html`, 2007.

[16] M. Moens. *Automatic Indexing and Abstracting of Document Texts*. The Information Retrieval Series. Springer, 2000.

[17] F. Mosteller and D. Wallace. *Inference and disputed authorship : the Federalist*. Addison-Wesley Reading, Mass, 1964.

[18] F. Mosteller and D. Wallace. *Applied Bayesian and classical inference: the case of the Federalist papers*. Springer series in statistics. Springer-Verlag, 1984.

[19] D. M. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`, 2010.

[20] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '88, pages 147–160, New York, NY, USA, 1988. ACM.

[21] V. Vapnik and A. Lerner. Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control*, 24, 1963.