Journal of Machine Learning Research 1 (2014) 1-48 Submitted 4/00; Published 10/00

Combination of Feature Engineering and Ranking Models for Paper-Author Identification in KDD Cup 2013

Chun-Liang Li R01922001@NTU.EDU.TW
Yu-Chuan Su R01922159@NTU.EDU.TW
Ting-Wei Lin R01944011@NTU.EDU.TW
Cheng-Hao Tsai R01922025@NTU.EDU.TW
Wei-Cheng Chang B99902019@NTU.EDU.TW
Kuan-Hao Huang B99902059@NTU.EDU.TW
Tzu-Ming Kuo B99902073@NTU.EDU.TW
Shan-Wei Lin B99902023@NTU.EDU.TW
Young-San Lin B97902055@NTU.EDU.TW
Yu-Chen Lu B98902105@NTU.EDU.TW
Chun-Pai Yang B99902109@NTU.EDU.TW
Cheng-Xia Chang R01944041@NTU.EDU.TW
Wei-Sheng Chin D01944006@NTU.EDU.TW
Yu-Chin Juan R01922136@NTU.EDU.TW
Hsiao-Yu Tung B98901044@NTU.EDU.TW
Jui-Pin Wang R01922165@NTU.EDU.TW
Cheng-Kuang Wei B98901037@NTU.EDU.TW
Felix Wu B99902090@NTU.EDU.TW
Tu-Chun Yin D00922023@NTU.EDU.TW
Tong Yu R01922141@NTU.EDU.TW
Yong Zhuang R01922139@NTU.EDU.TW
Shou-de Lin SDLIN@CSIE.NTU.EDU.TW
Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
National Taiwan University
Taipei 106, Taiwan

Editor:

Abstract

This paper describes the winning solution of team National Taiwan University for track 1 of KDD Cup 2013. The track 1 in KDD Cup 2013 considers the paper-author identification problem, which is to identify whether a paper is truly written by an author. First, we conduct feature engineering to transform the various types of provided text information into 97 features. Second, we train classification and ranking models using these features. Last, we combine our individual models to boost the performance by using results on the internal validation set and the official Valid set. Some effective post-processing techniques have also been proposed. Our solution achieves 0.98259 MAP score and ranks the first place on the private leaderboard of the Test set.

Keywords: Paper-Author Identification, Feature Generation

1. Introduction

In recent years, different open platforms such as Microsoft Academic Search,[1] Google Scholar,[2] and DBLP[3] have been constructed for providing various papers and authors information for the research community. One of the main challenges of providing this service is, by collecting the information from different sources on the Internet, author profiles may be incorrectly assigned to papers that are not written by them. This situation could be caused by author-name ambiguity, the same name shared by different authors, and the wrong paper-author information from the source. The research problem to address this challenge is called *paper-author identification*, which is to identify which papers are truly written by an author.

We briefly review existing approaches for this problem. Some have modeled it as a link prediction problem in social networks. For example, Sun et al. (2011) introduce Heterogeneous Bibliographic Network, which contains multiple types of nodes, including authors, papers, and topics. The links among these nodes represent different relations between authors and papers. Then several topological features could be extracted from the network to assist supervised learning techniques for link prediction. Sun et al. (2011) systematically extract some heterogeneous-network features and demonstrate that they are more effective than traditional homogeneous-network features.

Sun et al. (2012) generalize the concept of heterogeneous bibliographic networks to general heterogeneous networks. Their model leverages the interaction between different types of nodes to mine more semantic information of the network. Yang et al. (2012) apply probabilistic approaches and explore the temporal information on the network. Their experimental results on co-authorship prediction demonstrate the effectiveness. Lee and Adorna (2012) modify a heterogeneous bibliographic network by highlighting important relations in the network. Kuo et al. (2013) further study the heterogeneous network under the unsupervised settings with aggregative statistics. Besides the link prediction problem, the heterogeneous network has also been applied to other related problems, such as citation prediction (Sun et al., 2011; Yu et al., 2012).

Another problem related to paper-author identification is authorship contribution (Juola, 2006; Stamatatos, 2009). The goal of authorship contribution is to infer the characteristics of authors from given texts. Then we can distinguish the texts written by different authors.

KDD Cup is currently one of the most important data mining competitions. In 2013, track 1 of KDD Cup considers a problem of paper-author identification. The dataset is provided by Microsoft Academic Search. Participants are given thousands of authors and their publications. However, for any author, some papers may be wrongly assigned to him/her. Therefore, the goal of this competition is to identify which paper is truly written by an author from the given publications.

The paper describes the winning solution of team National Taiwan University. Our approach treats the problem as a binary classification or ranking problem. Therefore, we conduct feature engineering transforming the given text information into features and then apply the state-of-art binary classification and ranking algorithms. Last, we ensemble

1. http://academic.research.microsoft.com/
2. http://scholar.google.com.tw/
3. http://dblp.uni-trier.de/

several classification models and conduct a post-processing procedure to further boost the performance. According to the announced results, our approach achieves the best result with 0.98259 MAP score.

The paper is organized as follows. Section 2 introduces the track 1 problem of KDD Cup 2013. Section 3 outlines the framework of our approaches. Section 4 describes the approaches to transform the given text information into meaningful features. Section 5 discusses the models that we used. Section 6 describes how we combine different models and post-process the combined result to boost the performance. Finally, we conclude and discuss potential issues in Section 7.

Our implementation is available at `https://github.com/kdd-cup-2013-ntu/track1`. A preliminary version of the paper appeared in the KDD Cup 2013 Workshop (Li et al., 2013).

2. Track 1 of KDD Cup 2013

The dataset of track 1 of KDD Cup 2013 (Roy et al., 2013) is provided by Microsoft Academic Search. To address the paper-author identification problem, Microsoft Academic Search provides an interface allowing authors to confirm or delete the papers in their profiles. Confirmation means authors acknowledge they are the authors of the given paper; in contrast, deletion means authors claim that they are not the authors of the given papers (Roy et al., 2013). The dataset contains the information about authors and their confirmed/deleted papers. Based on author IDs, the organizers split the dataset to three parts, including *Train*, *Valid*, and *Test* sets.

The *Train* set (`Train.csv`) contains 3,739 authors. For each author, the AuthorId, ConfirmedPaperIds, and DeletedPaperIds are provided. The *Valid* set (`Valid.csv`) of 1,486 authors, each with an associated sequence of assigned paper IDs without confirmation or deletion, is for public leaderboard evaluation. The answers (confirmation/deletion) in the *Valid* set were released two weeks before the end of the competition. Participants were allowed to refine their algorithms based on the released answers of the *Valid* set, and were required to submit their models one week before the end of the competition. After the submission, the *Test* set (`Test.csv`) of 2,245 authors was used for private leaderboard evaluation.

In addition to the *Train* set, the following information is also provided.

- `Author.csv` contains author names and their affiliations.
- `Paper.csv` contains paper titles, years, conference IDs, journal IDs, and keywords.
- `PaperAuthor.csv` contains paper IDs, author IDs, author names, and affiliations.
- `Journal.csv` contains short names, full names, and home page information of journals.
- `Conference.csv` contains short names, full names, and home page information of conferences.

	# of authors	# of papers	# of confirmed papers	# of deleted papers
`Train.csv`	3,739	224,459	224,459	108,794
`Valid.csv`	1,486	86,755	41,024	47,081
`Test.csv`	2,245	129,427	–	–
`Paper.csv`	–	2,257,249	–	–
`Author.csv`	247,203	–	–	–
`PaperAuthor.csv`	2,143,148	2,258,482	–	–

Table 1: Statistics of the given files.

		Mean	Std.	Median	Min	Max	Q_1	Q_3
`Train.csv`	Confirmed	33.02	52.72	15	1	860	5	38
`Train.csv`	Deleted	30.08	107.34	6	1	2933	2	21
`Train.csv`	All	63.09	124.81	28	2	2977	11	68
`Valid.csv`	Confirmed	32.32	56.66	14	1	1324	5	38
`Valid.csv`	Deleted	27.79	78.12	5	1	1872	2	22
`Valid.csv`	All	60.22	103.15	28	2	2048	11	69
`Test.csv`	All	60.45	100.68	28	2	1371	11	68

Table 2: Statistics on the number of papers per author in the data set, where Q_1 and Q_3 are the first and third quartiles, respectively.

Unfortunately, the provided additional data are noisy and have missing values. For instance, `PaperAuthor.csv` contains the relations of authors and papers, but papers may be incorrectly assigned to an author. More statistics of the data are provided in Tables 1 and 2.

The goal of the competition is to predict which given papers are written by the given author. To be more specific, given confirmation and deletion records of authors as the training data (`Train.csv`), participants of the competition must predict which papers in the given paper list of each author in the test data (`Test.csv`) are truly written by him or her. The evaluation criterion is mean average precision (MAP), which is commonly used for ranking problems. Before answers of the *Valid* set were released, each team was allowed to submit their results on the *Valid* set five times per day and MAP scores were shown on the public leaderboard. During the last week of the competition, each team was allowed to submit multiple results on the *Test* set, and select one result for the final standing.

At National Taiwan University, we organized a course for KDD Cup 2013. Our members include three instructors, three TAs, and 18 students. The students were split into six sub-teams. Every week, each sub-team presented their progress and discussed with other sub-teams. The TAs helped to build an internal competition environment such that each sub-team could try their ideas before submitting their results to the competition website. Following the competition rules, the whole team share a single account for submitting results. According to the announced results, our approach achieves the best result on the *Test* set with 0.98259 MAP score.

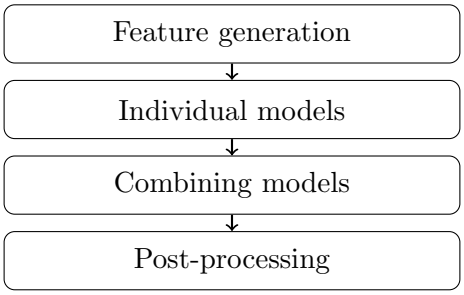

Figure 1: The framework of our approach.

3. Framework

This section first provides the framework of our system. Then we discuss the self-split internal validation set from the *Train* set. The internal validation is not only useful for off-line validating the model performance and combining different models, but also important for avoiding over-fitting the *Valid* set.

3.1 System Overview

We mentioned in Section 1 that we take a supervised learning approach. Our system can be divided into four stages: generating features, training individual models, combining different models, and post-processing as shown in Figure 1. The framework is similar to the one proposed in Yu et al. (2010), which is effective in data-mining applications.

In the first stage, we transform `Train.csv` into a binary classification training set. For each author in `Train.csv`, the list of confirmed papers and deleted papers are provided as described in Section 2. For each paper on the list, we can generate a corresponding author-paper pair, and each pair is treated as a training instance. The confirmation of an author-paper pair is a training instance with label 1; the deletion of of an author-paper pair is a training instance with label -1. We explore different approaches to generate features, which capture various aspects of the given text information.

In the second stage, we mainly employ three models, including Random Forests, Gradient Boosting Decision Tree and LambdaMART. For each individual model, to avoid over-fitting, we carefully conduct the parameter selection by using the internal validation set. In the third stage, we combine the three different models by using results on the internal validation set and the official *Valid* set. In the last stage, we post-process the combined result to further improve the performance by exploiting duplicated information which is not fully utilized by the models.

3.2 Validation Set

A validation set independent from the training set is useful for evaluating models. Given that the answers of the official *Valid* set are not available in the early stage of the competition, we construct an internal validation set for verifying our models. It is also useful to avoid over-fitting leaderboard results on the *Valid* set. In this competition, official *Train*, *Valid*

and *Test* sets are generated by first randomly shuffling authors, and then separate them into three parts with ratio 5:2:3 respectively. Therefore, we randomly split the *Train* set to have 2,670 authors as the internal training set and 1,069 authors as the internal validation set. In our experiments, the MAP score on the internal validation set is usually consistent with the one computed by five-fold cross validation on the official *Train* set.

4. Feature Engineering

To determine the confirmation or deletion of each author-paper pair, we treat each author-paper pair in `Train.csv` as a training instance with label 1 or -1 that represents confirmation or deletion, respectively. We then generate 97 features for each instance and apply the learning algorithms described in Section 5. Subsequently, in describing the feature generation for each author-paper pair, we refer to the author and the paper as the target author and the target paper, respectively.

In this section, we describe our approaches of transforming the given information into features. For the full feature list, please refer to the Appendix.

4.1 Preprocessing

Since many features are based on string matching, we conduct simple preprocessing to clean the data. We first replace the Latin alphabet with the English alphabet, such as replacing ó with o; we also delete some Greek alphabet letters, such as π. Then, we remove stop words in affiliations, titles and keywords, where the stop-word list is obtained from the NLTK package (Bird et al., 2009). Finally, we convert all characters into lowercase before comparison.

4.2 Features Using Author Information

This type of features stems from user profiles, such as user names or affiliations. Based on the information we try to capture, these features can be classified into the following three groups.

4.2.1 Confirmation of Author Profiles

An intuitive method to confirm that a paper is written by a given author is to check whether the name appears in the author section of the paper. However, a more careful setting is to check also the consistency of other information such as affiliations. In the competition, author affiliations are provided in `Author.csv` and `PaperAuthor.csv`. One basic assumption about `Author.csv` and `PaperAuthor.csv` is that `Author.csv` contains the author profiles maintained by Microsoft Academic Search, while the author information in `PaperAuthor.csv` is extracted from the paper without confirmation. The assumption is based on our observation on the given files as well as the online system. When there exists a conflict between `Author.csv` and `PaperAuthor.csv`, the author information in the online system is usually the same as that in `Author.csv`. Therefore, we generate features by comparing author names and affiliations between `Author.csv` and `PaperAuthor.csv`. The comparisons are done by string matching, and various string distances are used as features, including Jaro distance (Jaro, 1989, 1995), Levenshtein distance (Levenshtein,

1966), Jaccard distance (Jaccard, 1901a,b) (of words) and character match ratio. These features are simple but useful; for example, by using only the affiliation Levenshtein distance as a feature, we can achieve 0.94 MAP score on the *Valid* set.

An issue in author-name matching is to handle abbreviated names, which are very common in `PaperAuthor.csv`. In contrast, author names in `Author.csv` are usually in a complete format. The string distance between an abbreviated name and a full name may be large even if the two names are the same. Two different approaches are used to overcome the problem. The first one is to convert all names into an abbreviated format before the comparison; in our approach, the conversion is done by retaining only the last name and first character of first and middle names. The second approach is to split the author name into first, last and middle names, and compare each of them separately. The two approaches are applied independently to obtain different features.

Another challenge of name matching comes from the inconsistency of the name order. There are two main name orders in the provided data, the Western order and the Eastern order. The Western order means that given names precede surnames; in contrast, the Eastern order means that surnames precede given names. While most of the names are in the Western order, names in the Eastern order also frequently appear to cause failed comparisons. Although it is possible to check the name order and transform the Eastern-order names to Western-order ones before comparisons, such checking might be difficult and is prone to error. Instead, two different features are generated for the same distance measure. One assumes that names from `Author.csv` and `PaperAuthor.csv` are in the same name order. The other assumes that names are in the opposite order, so the name order in `Author.csv` is changed before string comparisons. Specifically, the order change is done by exchanging the first word and the last word in the name. However, this setting may wrongly consider two different author names as the same; for example, **Xue Yan** (PID:1224852) and **Yan Xue** (PID:482431) are considered as the same person in the generation of the second feature. Fortunately, because the number of Eastern-order name is relatively small in the data set, our approach still improves the overall performance.

4.2.2 Coauthor Name Matching

Features matching coauthor names are inspired by observing the dataset: in many deleted papers, there exist coauthors with names similar to the target author. For example, two authors (174432 and 1363357) of the deleted paper 5633 are the same as the target author **Li Zhang**. Therefore, having such coauthors is an important trait of deleted papers. To capture the information, we take the minimum string distance of names between the target author and his/her coauthors as a feature. Similar to the feature generation in Section 4.2.1, we also need to address the issue of abbreviated names and name orders.

Another problem for matching coauthor names is to decide names for comparison. For a given author identifier, corresponding names may appear in both `Author.csv` and `PaperAuthor.csv`. In fact, multiple names under the same identifier may appear in `PaperAuthor.csv`. These names may be different because of abbreviations, typos or even parsing errors of the Microsoft system. For example, author 1149778 is **Dariusz Adam Ceglarek** in `Author.csv`, while it corresponds to **Dariusz Ceglarek** and **D. Ceglarek** under paper 770630 in `PaperAuthor.csv`. Besides, some authors in `PaperAuthor.csv` do

not appear in `Author.csv`. To handle the problem, multiple features are generated, where each feature is computed by using different combinations of name sources. For instance, the target author name could be from `Author.csv` and `PaperAuthor.csv`, and coauthor names could be from `PaperAuthor.csv`. Then the distances of all possible combinations of the author and each coauthor names from different sources are computed. We select the minimum distance among all possible combinations to represent the name distance between the author and his/her coauthors. We give some examples to illustrate this type of features. One of the features is the maximal Jaro distance between the target author and all coauthors in the target paper. The list of coauthors is from the information in `PaperAuthor.csv`. To extract useful information from the names, we consider different forms of names for computing the distance: full name, abbreviated name, first name, last name and name under the order change (see section 4.2.1). We also employ other distance measures to obtain more features; see a complete list in Appendix A.1

4.2.3 Author Consistency

Understandably, information in the dataset should be consistent across papers and authors. Author-consistency features try to measure such information in author profiles. In particular, we measure the coauthor-affiliation consistency and research-topic consistency as features. Affiliation consistency is based on the assumption that authors with the same affiliation are more likely to co-work on a paper; therefore, we compute the affiliation string distance as well as the number of coauthors with the same affiliation as the target author. Similar to coauthor name matching, the affiliation may come from different sources, so we compute multiple features.

Research-topic consistency assumes that the author should work on related topics across different papers. Although the research topic or field information is not given in the dataset, we infer it from the paper titles and keywords. Therefore, we compute the title and keyword similarity between the target paper and other papers of the target author as features.

4.2.4 Missing Value Handling

Missing values cause difficulties in conducting string matching. A common situation in comparing author affiliations or author names is that both strings are empty. The resulting zero string distance wrongly indicates an identical match. As a result, papers with missing values tend to be ranked higher in prediction. To overcome this problem, we consider values other than zero in calculating the distance. If both strings for comparison are empty, we define their Jaro distance as 0.5, Jaccard distance as 0.5 and Levenshtein distance as the average length of the field. Besides, we use some indicators as features; examples include the number of coauthors without affiliation information.

4.3 Features Using Publication Time

Publication-time features are related to the publication year provided in `Paper.csv`. The intuition of these features is that an author can be active in a specific period, and papers written outside this period are likely authored by others. We include several features to capture the publication-time information, such as the exact publication year, publication-time span and publication year differences with other papers of the target author.

Determining whether the provided year is valid is an issue to resolve before we can generate year features. In the dataset, some papers' publication years such as 0, -1, and 800190 are obviously invalid. Besides, experiments on the internal validation set show that excluding publication years earlier than 1800 A.D. improves the overall performance. Therefore, we set the valid interval to be between 1800 A.D. and 2013 A.D. and ignore publication years outside the interval.

Removing invalid publication years incurs the missing value problem. To fill the missing year values, we utilize the publication-year information of coauthors. The basic concept is to replace a missing value with the average of the mean publication years of all coauthors of the paper. This average, however, is not computable because coauthors may also have missing information on publication years. An iterative process is used to solve the problem as follows. First, papers with invalid years are ignored and mean of available publication years is calculated for each author. The mean value is then used to fill the missing value of the author. These new values can be incorporated to calculate the new mean value of the publication years. Therefore, the mean publication years and missing values are computed alternatively until convergence. We list the procedure as follows. Please refer to our implementation for detailed steps.

1. Let $\mathcal{P}$ be the set of papers with valid years, and $m_{\mathcal{P}}$ be the map that maps each paper $p \in \mathcal{P}$ to its publication year.

2. Let $\mathcal{A}$ be the set of authors of $\mathcal{P}$ and $m_{\mathcal{A}}$ be the map that maps each author $a \in \mathcal{A}$ to his/her mean publication year calculated based on $m_{\mathcal{P}}$.

3. Let $\mathcal{P}'$ be the set of papers with invalid years, and $m_{\mathcal{P}'}$ be the map that maps each paper $p \in \mathcal{P}'$ to the average of mean publication years of its authors in m_a. If the paper $p \in \mathcal{P}'$ does not have any author in $\mathcal{A}$, the publication year is assigned to 0 in $m_{\mathcal{P}'}$.

4. Let $m_{\mathcal{P}''} = m_{\mathcal{P}'}$.

5. Let $\mathcal{A}'$ be the set of authors having papers in $\mathcal{P} \cup \mathcal{P}'$, and $m_{\mathcal{A}'}$ be the map that maps each author $a \in \mathcal{A}'$ to his/her mean publication year calculated from $m_{\mathcal{P}}$ and $m_{\mathcal{P}'}$.

6. Update $m_{\mathcal{P}'}$ by mapping $p \in \mathcal{P}'$ to the average of mean publication year of its authors according to $m_{\mathcal{A}'}$.

7. If the mean squared differences between years of $m_{p'}$ and $m_{p''}$ is smaller than a given threshold, any zero entry of $m_{p'}$ is replaced by the mean year of m_p and $m_{p'}$. Then stop the procedure and return $\mathcal{P}'$ and $m_{\mathcal{P}'}$.

8. Let $m_{\mathcal{P}''} = m_{\mathcal{P}'}$ and go to step 5.

4.4 Features Using Heterogeneous Bibliographic Networks

Sun et al. (2011) introduce the concept of Heterogeneous Bibliographic Network to capture the different relations between authors and papers, and demonstrate the effectiveness of

link prediction. In this competition, finding whether a paper is written by a given author becomes predicting a link between an author and a paper. According to our study, the relationship between authors and their publications, or coauthors is very useful for linking prediction. This observation is consistent with the claim in Sun et al. (2011). Because such information can be captured by Heterogeneous Bibliographic Network, and by computing certain structures of the network as features, we can obtain the relation from the network to improve the prediction accuracy.

Heterogeneous Bibliographic Network is a graph $G = (V, E)$, where V is the vertex set and E is the edge set. According to the given data, the vertex set $V = \mathcal{P} \cup \mathcal{A} \cup \mathcal{C} \cup \mathcal{J}$ contains the set of papers $\mathcal{P}$, the set of authors $\mathcal{A}$, the set of conferences $\mathcal{C}$ and the set of journals $\mathcal{J}$. The set E consists of two kinds of edges. Based on `PaperAuthor.csv`, if author a_i writes paper p_j, then we create the edge e_{ij}; based on `Papers.csv`, if paper p_m belongs to conference c_n or journal j_n, then we create the edge e_{mn}. Note that, because information in `PaperAuthor.csv` may be incorrect, some links are wrongly generated in the network.

After generating the network, we could extract basic features, such as the number of publications of an author, and the number of total coauthors of an author.

To utilize the network structure, we further define the "path" to describe node relationship. Given the paper-author pair (p_i, a_j), a length-k *meta path* is defined as $(p_i \leftrightarrow v_1 \leftrightarrow \cdots \leftrightarrow v_{k-1} \leftrightarrow a_j)$, where $v_1, \cdots, v_{k-1} \in V$ and $\leftrightarrow$ means two nodes are connected by an edge. Various paths of the graph are extracted as features. In Appendix A.3, we list all kinds of meta paths used to generate features. Although these paths are extracted from the graph structure, they have clear physical meaning and can be interpreted easily. For example, the sixth feature on the list corresponds to the size of the following meta-path set: $S_{mn} = \{(p_m \leftrightarrow j \leftrightarrow \bar{p} \leftrightarrow a_n)\}$, where (p_m, a_n) are given and length-3 meta paths capture all papers of author a_n published in the same journal j as p_m. Take the eighteenth feature as another example. Under given (p_m, a_n), this feature indicates the number of different p_j's on meta paths $(a_n \leftrightarrow p_m \leftrightarrow a_i \leftrightarrow p_j)$. It captures the total number of papers written by coauthors in the target paper.

Further, given an author pair (a_i, a_j), a length-k *pseudo path* is defined as $(a_i \sim a_1 \sim \cdots \sim a_{k-1} \sim a_j)$, where $a_1, \cdots, a_{k-1} \in \mathcal{A}$. Because there is no edge between two author nodes in our network, $\sim$ is a pseudo edge. If author node a_j is reachable from a_i on the network by traversing non-author nodes, then we consider there is a pseudo edge between a_i and a_j. In other words, the pseudo edge describes the possible co-authorship between two authors. By considering the pseudo paths, we can grasp different co-authorship information. The second feature in Appendix A.3 uses the pseudo-edge information directly by computing the number of neighboring a_i's of the target author a_n. This means the number of coauthors of the target author. The pseudo edge is also used implicitly by many other features. For example, the sixteenth feature in Appendix A.3 relies on pseudo edges to identify the coauthors of the target author and then computes the average number of papers of the coauthors.

5. Models

After generating features, we apply classification and ranking methods to train the data set. To enhance the diversity, we explore tree-based classifiers and linear classifiers. The tree-based algorithms including Random Forests (Breiman, 2001), Gradient Boosting Decision Tree (Friedman, 2002), and LambdaMart (Wu et al., 2010). The linear classifier we have studied is RankSVM (Herbrich et al., 2000). However, because RankSVM does not make any improvement in the ensemble stage as described in the Section 6, it is not used in generating our final results.

5.1 Random Forests

Random Forests is a tree-based learning method introduced by Breiman (2001). The algorithm constructs multiple decision trees using randomly sub-sampled features and instances. For prediction, the output is by averaging the results of individual trees. The use of multiple trees reduces the variance of prediction, so Random Forests is robust and useful in this competition.

We use the implementation in the scikit-learn package (Pedregosa et al., 2011). The package provides a parallel module to significantly speed up the tree building process. Note that the scikit-learn implementation combines classifiers by averaging probabilistic predictions instead of a voting mechanism in Breiman (2001). To construct each tree in the forest, the same number of training samples as in the original training set are sampled with replacement. Thus, the expected number of training instances for each tree is $1 - \frac{1}{e}$ times the original training set size, while some instances sampled more than once have higher weights.

In this competition, the variance may influence the standing on the leaderboard significantly. For example, with different random seeds and fewer trees, the performance of Random Forests can vibrate from 0.981 to 0.985 on the *Valid* set. On the public leaderboard, the scores of top 20 places are from 0.98130 to 0.98554. Moreover, the improvement on the *Valid* set by changing the random seed may not be consistent with the result on the self-split internal validation set. Therefore, changing the random seeds may cause over-fitting. Our experiments show that using more trees leads to better and consistent validation scores on both *Valid* set and the internal validation set due to lower variance. Because of the time limit, we use a subset of 55 features,[4] $12,000$ trees and a fixed random seed 1 in our Random Forests model after some trials. In addition to the number of trees, we also tune the minimal number of training samples in a leaf of each decision tree. This setting achieves 0.983340 MAP score on the *Valid* set. The parameters we have used are listed in Table 5.1. For unlisted parameters, we use the default values in Pedregosa et al. (2011).

5.2 Gradient Boosting Decision Tree

Gradient Boosting Decision Tree (GBDT) (Friedman, 2002), also called MART, is a tree-based learning algorithm. The goal of GBDT is using $(y, \mathbf{x})$, where $\mathbf{x}$ is the known feature vector and y is the corresponding label, to find a classifier $H^*(\mathbf{x})$ to minimize the expected

4. Because some features take more time for generation and debugging, we only use 55 stable ones to train the final Random Forests model.

Parameter	Value
Number of trees	12,000
Minimal number of samples in a leaf	10

Table 3: Tuned parameters for Random Forests.

value of the given error function $\text{err}(y, H(\mathbf{x}))$. Therefore, the target classifier is defined as

$$H^*(\mathbf{x}) = \arg\min_{H(x)} E_{y,\mathbf{x}}[\text{err}(y, H(\mathbf{x}))].$$

From the functional gradient descent perspective (Friedman, 2002), we could approximate $H^*(\mathbf{x})$ by combining several "weak" classifiers $h_t(\mathbf{x})$ as follows,

$$H_T(\mathbf{x}) = \sum_{t=0}^{T} \alpha_t h_t(\mathbf{x}),$$

where $T+1$ is the number of weak classifiers. Then we can boost the performance in an iterative manner. After we train an initial classifier h_0, for each iteration t, where $t \geq 1$, we solve the following optimization problem,

$$(h_t(\mathbf{x}), \alpha_t) = \arg\min_{h(\mathbf{x}),\alpha} \sum_{i=1}^{N} \text{err}(y_i, H_{t-1}(\mathbf{x}_i) + \alpha h(\mathbf{x}_i)),$$

where α_t is a scalar and N is the number of training instances. Then the update rule is $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$. The GBDT is one variant of the functional gradient descent algorithm. The base (weak) classifier used in GBDT is the regression tree with constant predictions; that is, for each leaf node L, the prediction is $\frac{1}{|L|}\sum_{(\mathbf{x}_i,y_i)\in L} y_i$. To avoid overfitting, we usually use a learning rate η to shrink the effect of α_t. Therefore, the update rule becomes $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \eta\alpha_t h_t(\mathbf{x})$.

Compared with Random Forests, a GBDT model is built sequentially and it combines built trees to generate a powerful learner by an iterative boosting way under the functional gradient descent perspective. We use the same package scikit-learn (Pedregosa et al., 2011). The error function of GBDT implemented in Pedregosa et al. (2011) is to optimize "deviance" which is same as the objective of logistic regression. The main disadvantage of GBDT is that it cannot be trained in parallel, so we only use 300 trees to build the final ensemble model of GBDT. This is much smaller than 12,000 for Random Forests. The tuned parameters are listed in Table 4 while the unlisted parameters are set to the default values. With the tuned parameters, the GBDT model could achieve 0.983046 MAP score on the *Valid* set.

5.3 LambdaMart

We choose LambdaMART (Wu et al., 2010) because of its recent success on Yahoo! Learning to Rank Challenge (Chapelle and Chang, 2011). LambdaMART is the combination of GBDT (Friedman, 2002) and LambdaRank (Burges et al., 2006). Burges et al. (2006) propose to use a utility function whose derivative is the gradient of a typical pairwise

Parameter	Value
Number of trees	300
Learning Rate	0.08
Tree Depth	5
Minimal number of samples in a leaf	9

Table 4: Tuned parameters for Gradient Boosting Decision Tree.

Parameter	Value
Number of trees	1,000
Minimal sample ratio in a leaf	0.01
Number of leaves	32
Ratio of sampled instances	0.3

Table 5: Tuned parameters for LambdaMART.

error function times the difference of the desired evaluation criterion, such as NDCG, by exchanging the ranking order of a pair (i, j). In contrast, GBDT (MART) aims to model the gradient in each iteration. Therefore, the main advantage of LambdaMART is that it uses LambdaRank gradients of the proposed utility function in GBDT to consider highly non-smooth ranking metrics. We use the implementation in the JForests (Ganjisaffar et al., 2011), which optimizes the NDCG metric. The detailed parameters are listed in Table 5. Compared with Random Forests and Gradient Boosting Decision Tree, LambdaMART is a more aggressive ranking algorithm. To avoid over-fitting, we train 10 LambdaMART models with random seeds from 0 to 9, and average the output confidence scores. With the listed parameters and the bagging approach, the LambdaMART model could achieve 0.983047 MAP score on the *Valid* set.

5.4 RankSVM

Besides the above tree-based models, we also explore the commonly-used RankSVM (Herbrich et al., 2000), which is extended from standard support vector machines (Vapnik, 1998). Given the author a and two papers p_i and p_j, RankSVM aims to predict p_i with a higher score than p_j, if p_i is written by the author a while p_j is not. By defining a set of pairs of the author a as

$$\mathcal{P}_a \equiv \{(p_i, p_j) \mid p_i \text{ is written by } a \text{ while } p_j \text{ is not}\}.$$

We consider the following L1-loss SVM,

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{a \in \mathcal{A}} \sum_{(i,j) \in \mathcal{P}_a} \max(0, 1 - \mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j)),$$

where $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the regularization term and C is the regularization parameter. Due to the efficiency issue, we only study linear rather than kernel RankSVM. We consider the implementation in Lee and Lin (2014), which optimizes the L2-loss instead. The best parameter in our study is $C = 0.001$, which results in 0.97911 MAP score on the Valid set.

5.5 Summary

We summarize the results of the four studied algorithms in Table 5.5.

Algorithm	MAP Score
Random Forests	0.983340
Gradient Boosting Decision Tree	0.983046
LambdaMart	0.983047
RankSVM	0.979110

Table 6: The results of four studied algorithms on the *Valid* set (public leaderboard).

6. Ensemble and Post-Processing

To further boost our performance, we ensemble results of different models and conduct a post-processing procedure.

6.1 Ensemble

In many past competitions, such as Netflix and KDD Cup, winners have shown that an ensemble of individual models may significantly improve the prediction results (Tösscher et al., 2009; Yu et al., 2010; Wu et al., 2012). The main reason is that the diversification of models compensates the weakness of each model. Existing approaches to ensemble classifiers are based on some optimization techniques (Burges et al., 2005) because they often aim to combine a large number of models.

In our system, we calculate a simple weighted average after scaling the decision values of each model to be between 0 and 1. Because only four models described in Section 5 were built, we search a grid of weights to find the best setting rather than applying more complicated optimization techniques.

To see the performance under a setting of weights, we check the results on the internal validation set and the official *Valid* set. Specifically, we train four models on the internal training set, and predict on the internal validation set. Then we combine the results by adjusting weights to seek for improvements. Similarly, we train four models on the *Train* set (internal training set + internal validation set) and predict on the *Valid* set. Then we check whether results are further improved. The final weights are 0 for RankSVM (unused), 1 for both Gradient Boosting Decision Tree and LambdaMART, and 5 for Random Forests.

Based on the MAP scores reported in Section 5 and the weights for ensemble, tree-based models are more effective than the linear model in this task. This situation is similar to some ranking tasks discussed in Chapelle and Chang (2011).

6.2 Post-Processing

6.2.1 Duplicated Paper-Author Pairs

In Section 4.4, we describe the concept of Heterogeneous Bibliographic Network. Even if there is an edge between the author node a and the paper node p, a may not be the author of p because of the incorrect information in `PaperAuthor.csv`. To get confidence on each

link, we observe from `PaperAuthor.csv` that there are some duplicated paper-author pairs. For example, lines 147,035 and 147,036 record the same author-paper pair. We observe that duplicates highly correlate with the confirmation. Therefore, we let the number of duplicates be the weight of the edge between a paper and an author. We use weighted edges in two ways. First, we add a feature to illustrate the number of duplicates before the training procedure to obtain models described in Section 5. Second, according to the number of duplicates, we divide the given papers of each author into two groups: those having more than one duplicate and those having only one. Then in our prediction, we rank the first group before the second. For each group, we rank its members according to their decision values.

6.2.2 Duplicated Paper ID

In the *Test* set, the assigned papers of an author may contain duplicates. For example, author 100 has five papers 1, 2, 2, 3 and 4 to be ranked, and confirmed papers are 1, 2, 2 and 4. According to the algorithm provided by the competition organizer for calculating MAP, only one of these duplicated paper IDs will be calculated in MAP. Therefore, the list 1, 2, 4, 3, 2 has a higher MAP than the list 1, 2, 2, 4, 3 because the second paper with ID 2 is treated as a deleted paper in the evaluation algorithm. Based on this observation, we put all duplicated paper IDs to the end of the ranked list as deleted papers.

7. Discussion and Conclusion

In this section, we discuss some issues related to our approach and/or the KDD Cup competition. We investigate the feature importance reported by Random Forests in Table 7 because of its best performance among all the single models we used. The two most important features are related to the number of duplicates, which justifies the validity of post-processing in Section 6.2. The next two are about the affiliation consistency. Their high ranks support our observation that some mis-assignments are caused by similar names in different institutes. The features ranked next are about the name similarity between the target author and co-authors with different affiliations. Note that for these features, first name and last name are not exchanged. These features are also related to name ambiguity. If the name of the target author is the same or almost the same as a co-author of the same paper, usually the assignment is wrong.

We discuss some potential issues and difficulties for applying our method to Microsoft Academic Search or any other real online systems in practice. One potential drawback of our method in terms of scalability is the feature generation step, which may have superlinear time complexity. In particular, several coauthor name-matching features require computing the string distances between the target author and all coauthors, and each author may have several different names depending on the number of publications the author has. The computation time will be a serious issue when an author has many publications, and a paper has many authors. This situation is very common in fields such as high-energy physics. Note that feature computation is also an important issue in the prediction stage because a real-time response for the system is required. Another drawback of our method is that it cannot be updated in an incremental manner. Instead, whenever the data set is updated, features must be recomputed and the model must be retrained. To adapt the

Rank	Feature	Average Importance	Standard Deviation	Rank	Feature	Average Importance	Standard Deviation
1	A.3.4	0.143027	0.003299	29	A.1.2.5	0.004102	0.000057
2	A.3.28	0.124315	0.001593	30	A.1.2.15	0.003954	0.000051
3	A.1.1.4	0.10853	0.002038	31	A.1.2.16	0.003909	0.00003
4	A.1.1.2	0.096152	0.001749	32	A.3.3	0.003824	0.000104
5	A.1.2.12	0.077095	0.000913	33	A.3.8	0.00374	0.000008
6	A.1.2.6	0.072966	0.00107	34	A.1.2.19	0.003506	0.000105
7	A.1.2.17	0.051346	0.001337	35	A.1.2.20	0.003265	0.000115
8	A.1.2.7	0.040475	0.000919	36	A.1.2.21	0.002957	0.000112
9	A.1.2.13	0.031379	0.000694	37	A.1.3.1	0.002922	0.00003
10	A.1.2.23	0.02523	0.000957	38	A.2.18	0.002588	0.000003
11	A.1.2.3	0.020658	0.000323	39	A.1.2.9	0.002302	0.000121
12	A.1.2.24	0.020075	0.000416	40	A.3.5	0.002172	0.000005
13	A.1.3.4	0.017408	0.000343	41	A.1.3.2	0.001948	0.000008
14	A.1.3.5	0.014549	0.000255	42	A.1.2.18	0.0019	0.00003
15	A.1.3.3	0.012566	0.000331	43	A.1.3.11	0.001681	0.000003
16	A.1.2.4	0.012349	0.000673	44	A.1.3.8	0.001638	0.000024
17	A.1.3.7	0.011437	0.0003	45	A.1.3.10	0.001531	0.000033
18	A.3.2	0.009053	0.000096	46	A.1.3.9	0.001498	0.00004
19	A.1.2.22	0.008349	0.000331	47	A.1.2.11	0.001468	0.000005
20	A.3.1	0.006641	0.000031	48	A.1.3.12	0.001289	0.000001
21	A.3.27	0.006436	0.000095	49	A.3.29	0.000965	0.000084
22	A.3.26	0.006392	0.000127	50	A.1.1.5	0.000917	0.00003
23	A.3.25	0.00527	0.000022	51	A.1.3.13	0.000789	0.000003
24	A.1.1.3	0.00514	0.00009	52	A.1.2.8	0.000284	0.000006
25	A.1.2.14	0.004899	0.000062	53	A.3.6	0	0
26	A.1.3.6	0.004572	0.000085	54	A.3.12	0	0
27	A.3.7	0.004355	0.000043	55	A.3.11	0	0
28	A.1.2.10	0.004185	0.000005				

Table 7: Mean and standard deviation of feature importance by training Random Forests with ten different random seeds.

proposed system for real applications, acceleration for feature computation such as using an indexing structure (Jin et al., 2005) or conducting name grouping (Cohen et al., 2003) is necessary.

Another issue for our system (and maybe systems of other teams in this competition) is the cost effectiveness. While we use 97 different features in our final system to achieve 0.98259 MAP, we can achieve around 0.94 MAP by using one single name-matching (string distance) feature. The 0.04 MAP gain comes at a high cost in both training and testing, but whether this gain enhances users' satisfaction remains to be further investigated.

The last issue is about the data. We discussed in Section 4 that some duplicates author-paper pairs and duplicated IDs appear in the data. Although the features considering duplicates are useful in the competition, they might not be effective in practice. The cause of duplicates may be because that the system crawls data from different sources without conducting any data cleaning. If this hypothesis holds, then the number of duplicates can represent certain confidence supported by different sources and our approaches might still be valid and useful in practice. If it does not, the cause of duplicates and the usefulness of the proposed approaches remain to be further studied.

We also discuss the potential future work of our approaches. In Section 4.2.3, we assume the coauthor-affiliation consistency and research-topic consistency. In practice, it is common that an author works on more than one research topic and co-works with different institutes. Further, the affiliations and research topics of an author may change along with time. Therefore, how to model different research topics and time information into features is a topic worth studying.

In conclusion, we introduce the approaches of team National Taiwan University for track 1 of KDD Cup 2013. We successfully transform the given text information into several useful features and propose techniques to address the issue of noisy texts for making features robust. We then apply several state-of-the-art algorithms on the generated features. To further improve the performance, we conduct a simple weighted-average ensemble and a post-processing procedure by utilizing some duplicated information. During each stage, we cautiously use the internal validation or the official *Valid* set to potentially avoid the over-fitting issue. This step is crucial for us to get the best performance on the private leaderboard for predicting data in the *Test* set. A detailed summary of our approach is in Figure 2.

Acknowledgments

We thank the organizers for holding this interesting competition. We also thank the College of Electrical Engineering and Computer Science as well as the Department of Computer Science and Information Engineering at National Taiwan University for their supports and for providing a stimulating research environment. The work was also supported by National Taiwan University under Grants NTU 102R7827, 102R7828, 102R7829, and by National Science Council under Grants NSC 101-2221-E002-199-MY3, 101-2628-E002-028-MY2, 101-2628-E002-029-MY2.

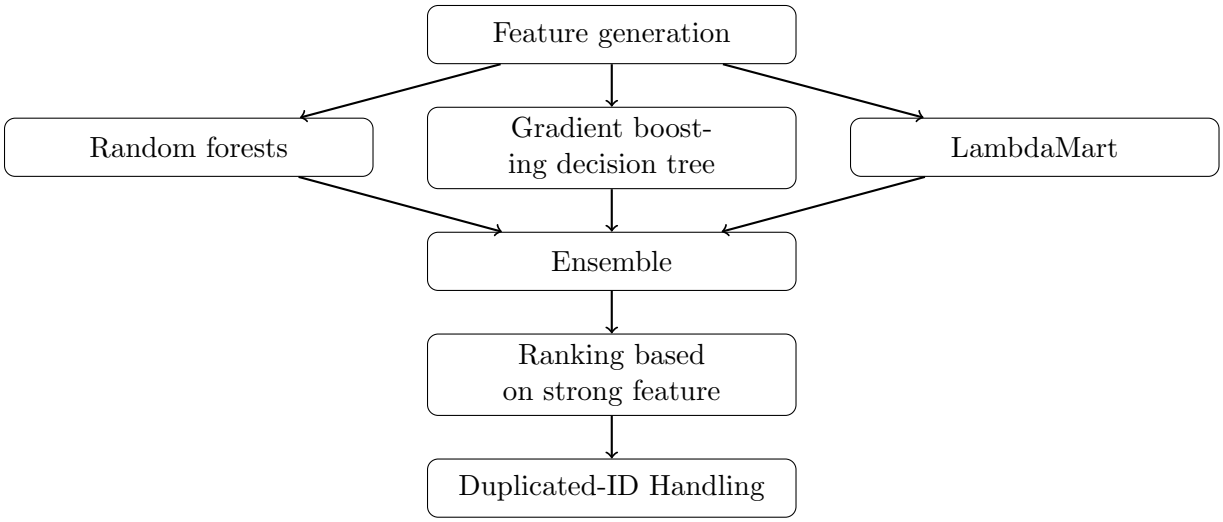

Figure 2: The detailed architecture of our approach.

Appendix A. Feature List

Since our team members are divided into several sub-groups internally, some features are repeatedly generated. For these features, we denote the n times repeats by (*n) at the end of the description.

A.1 Features Using Author Information

A.1.1 Confirmation of Author Profile

1. The Levenshtein distance between the names of the target author in `Author.csv` and `PaperAuthor.csv`.
2. The Levenshtein distance between the affiliations of the target author in `Author.csv` and `PaperAuthor.csv` (*2).
3. The ratio of matched substring between the names of the target author in `Author.csv` and `PaperAuthor.csv`.
4. The ratio of matched substring between the affiliations of the target author in `Author.csv` and `PaperAuthor.csv`.
5. The ratio of matched substring between the abbreviated names of the target author in `Author.csv` and `PaperAuthor.csv`.

A.1.2 Coauthor Name Matching

1. The maximum Jaro distances between the target author's name and each coauthor's name. The names are from `PaperAuthor.csv` under the target paper.

2. The maximum Jaro distances between the last names of the target author and each coauthor. The names are from `PaperAuthor.csv` under the target paper.

3. The maximum Jaro distances between the target author's name and each coauthor's name. The names are from `PaperAuthor.csv` under the target paper. Coauthors having the same affiliation with the target author are ignored during the comparison.

4. The minimum Levenshtein distances between the target author's name and each coauthor's name. The names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of the target author are ignored during comparison.

5. The number of authors having the same name as the target author in the entire dataset.

6. The maximum Jaro distances between the abbreviated names of the target author and each coauthor. The names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

7. The minimum among Levenshtein distances between the abbreviated names of the target author and each coauthor. The names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

8. The minimum substring matched ratios between the target author's last name and each coauthor's last name. The author's name is form `Author.csv`, and coauthors' names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

9. The minimum substring matched ratios between the target author's first name and each coauthor's first name. The author's name is form `Author.csv`, and coauthors' names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

10. The minimum substring matched ratios between the target author's reversed name and each coauthor's name. Middle name is ignored, and the target author's first name and last name are exchanged before comparison. The author's name is form `Author.csv`, and coauthors' names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

11. The minimum substring matched ratios between the target author's middle name and each coauthor's middle name. The author's name is form `Author.csv`, and coauthors' names are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

12. The maximum Jaro distances between the target author's last name and each coauthor's last name. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

13. The maximum Jaro distances between the target author's first name and each coauthor's first name. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

14. The maximum Jaro distances between the target author's name and each coauthor's name. Middle name is ignored, and the target author's first name and last name are exchanged before comparison. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

15. The maximum Jaro distances between the abbreviated names of the target author and each coauthor. Middle name is ignored, and the target author's first name and last name are exchanged before abbreviation. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

16. The maximum Jaro distances between the abbreviated names of the target author and each coauthor. Middle name is ignored, and the coauthor's first name and last name are exchanged before abbreviation. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

17. The minimum Levenshtein distances between the target author's last name and each coauthor's last name. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

18. The minimum Levenshtein distances between the target author's first name and each coauthor's first name. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

19. The minimum Levenshtein distances between the target author's name and each coauthor's name. Middle name is ignored, and the target author's first name and last name are exchanged before comparison. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

20. The minimum Levenshtein distances between the abbreviated names of the target author and each coauthor. Middle name is ignored, and the target author's first name and last name are exchanged before abbreviation. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

21. The minimum Levenshtein distances between the abbreviated names of the target author and each coauthor. Middle name is ignored, and the coauthor's first name and last name are exchanged before abbreviation. The names are from `PaperAuthor.csv` under the target paper. Coauthors in the same affiliation as the target author are ignored during comparison.

22. The maximum of affiliation Jaro distances times name Levenshtein distances between target author and coauthors. Both author name and affiliation are from `PaperAuthor.csv`.

23. The maximum Jaro distances between the target author's name and each coauthor's name. The name of target author is from `Author.csv`, and that of coauthors are from

`PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

24. The minimum Levenshtein distances between the target author's name and each coauthor's name. The name of target author is from `Author.csv`, and that of coauthors are from `PaperAuthor.csv` under the target paper. Coauthors that are in the same affiliation of target author are ignored during comparison.

A.1.3 Author Consistency

1. The maximum Jaro distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `Author.csv`.
2. The maximum Levenshtein distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `Author.csv`.
3. The maximum Jaro distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `PaperAuthor.csv` under the target paper.
4. The minimum Levenshtein distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `PaperAuthor.csv` under the target paper.
5. The maximum Jaro distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `PaperAuthor.csv` under all papers published by a given author.
6. The maximum Levenshtein distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `PaperAuthor.csv` under all papers published by a given author.
7. The maximum Jaccard distance between the affiliation of the target author and affiliations of coauthors in the paper. The affiliations are from `PaperAuthor.csv` under all papers published by a given author.
8. The number of coauthors in the same affiliation as the target author. The affiliations are from `PaperAuthor.csv` under the target paper.
9. The number of authors with no affiliation information in `PaperAuthor.csv` under the target paper.
10. The percentage of authors with no affiliation information in `PaperAuthor.csv` under the target paper.
11. Maximum paper title Jaro distance of the target paper and papers written by the author.
12. Minimum paper title Levenshtein distance of the target paper and papers written by the author.
13. Maximum keywords Jaccard distance of the target paper and papers written by the author.

A.2 Features Using Publication Time

1. Earliest publication year of the author (*2).
2. Latest publication year of the author (*3).
3. Publication year of the paper, and the invalid year is replaced by 0 (*3).
4. Indicator to see if the publication year of the paper is missing.
5. Publication year after filling missing value.
6. Mean publication year of all papers of the author.
7. Standard deviation of publication year of all papers of the author.
8. Mean publication year of the authors' papers in the same conference as the target paper.
9. Standard deviation of the publication year of the authors' papers in the same conference as the target paper.
10. Mean publication year of the authors' papers in the same journal as the target paper.
11. Standard deviation of the publication year of the authors' papers in the same journal as the target paper.
12. Mean publication year of all papers in the same conference as the target paper.
13. Standard deviation of the publication year of all papers in the same conference as the target paper.
14. Mean publication year of all papers in the same journal as the target paper.
15. Standard deviation of the publication year of all papers in the same journal as the target paper.
16. The difference between target author's the latest publication year and the earliest publication year.
17. The difference between the target paper's publication year and the median of the publication year of all the papers of the target author.
18. The maximum publication-year difference between the target paper and papers of the target author.

A.3 Features Using Heterogeneous Bibliographic Network

1. Total number of papers published by the target author (*3).
2. Total number of coauthors of the target author (*4).
3. Number of authors of the target paper (*3).
4. Number of occurrences of the (PID,AID) pairs in `PaperAuthor.csv` (*2, and used for post processing).

5. Number of papers the author published in the conference of the target paper (*3).
6. Number of papers the author published in the journal of the target paper (*3).
7. Number of conference papers of the author (*2).
8. Percentage of conference papers of the author.
9. Number of conferences the author has papers in.
10. Number of journal papers of the author (*2).
11. Percentage of journal papers of the author.
12. Number of journals the author has papers in.
13. Average paper number of the author in conferences he/she has published in.
14. Average paper number of the author in journals he/she has published in.
15. Total number of papers written by coauthors of the target author.
16. Average paper number of coauthors of the target author.
17. The variance of paper number of coauthors of the target author.
18. Total number of papers written by coauthors in the target paper.
19. Average paper number of coauthors in the target paper.
20. The variance of paper number of coauthors in the target paper.
21. Indicator of journal papers.
22. Indicator of conference papers.
23. The difference between the number of conference papers and journal papers written by the target author.
24. The number of coauthors in the paper that have coauthored other papers with the target author.
25. The percentage of papers that are coauthored with at least one of the coauthors of the target paper.
26. Maximum number of coauthored papers with coauthors of the target paper.
27. Maximum percentage of coauthored papers (with respect to total number of papers written by the target author) with coauthors of the target paper.
28. Number of coauthors that appear more than once under the target paper in `PaperAuthor.csv`.
29. Indicator of whether the paper has only one author.

30. Number of papers published by the author which has duplicated (PID, AID) in `PaperAuthor.csv`.

31. Number of coauthored papers of the target author with all the coauthors of the target paper.

32. Number of coauthored papers of the target author with all the coauthors of the target paper, divided by the total number of coauthored papers of the target author with each coauthor of the target paper.

33. Number of coauthored papers of the target author with all the coauthors of the target paper (excluding the target paper).

34. Number of coauthored papers of the target author with all the coauthors of the target paper, divided by total number of coauthored papers of the target author with all coauthors of the target paper (excluding the target paper).

35. Total number of coauthored papers of the target author with all possible coauthors (*2).

36. Average number of coauthored papers of the target author with each coauthor of the target paper (*2).

37. Number of coauthored papers of the target author with all the coauthors of the target paper.

References

Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with Python, 2009.

Leo Breiman. Random forests. *Machine Learning*, 2001.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19*, 2006.

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - Proceedings Track*, 2011.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *KDD Workshop on Data Cleaning and Object Consolidation*, pages 73–78, 2003.

Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 2002.

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th International Conference on Research and Development in Information Retrieval*, 2011.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

Paul Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 1901a.

Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 1901b.

Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 1989.

Matthew A. Jaro. Probabilistic linkage of large public health data file. In *Statistics in Medicine*, 1995.

Liang Jin, Chen Li, Nick Koudas, and Anthony K. H. Tung. Indexing mixed types for approximate retrieval. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 793–804, 2005.

Patrick Juola. Authorship attribution. *Foundations and Trends in Information Retrieval*, 2006.

Tsung-Ting Kuo, Rui Yan, Yu-Yang Huang, Perng-Hwa Kung, and Shou-De Lin. Unsupervised link prediction using aggregative statistics on heterogeneous social networks. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, 2013.

Ching-Pei Lee and Chih-Jen Lin. Large-scale linear rankSVM. *Neural Computation*, 2014. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/ranksvm/ranksvml2.pdf`. To appear.

John Boaz Lee and Henry Adorna. Link prediction in a modified heterogeneous bibliographic network. In *Advances in Social Networks Analysis and Mining*, 2012.

VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 1966.

Chun-Liang Li, Yu-Chuan Su, Ting-Wei Lin, Cheng-Hao Tsai, Wei-Cheng Chang, Kuan-Hao Huang, Tzu-Ming Kuo, Shan-Wei Lin, Young-San Lin, Yu-Chen Lu, Chun-Pai Yang, Cheng-Xia Chang, Wei-Sheng Chin, Yu-Chin Juan, Hsiao-Yu Tung, Jui-Pin Wang, Cheng-Kuang Wei, Felix Wu, Tu-Chun Yin, Tong Yu, Yong Zhuang, Shou-de Lin, Hsuan-Tien Lin, and Chih-Jen Lin. Combination of feature engineering and ranking models for paper-author identification in kdd cup 2013. In *KDD Cup 2013 Workshop*, 2013.

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. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

Senjuti Basu Roy, Martine De Cock, Vani Mandava, Swapna Savanna, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. The microsoft academic search dataset and kdd cup 2013. In *KDD Cup 2013 Workshop*, 2013.

Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 2009.

Yizhou Sun, Rick Barber, Manish Gupta, Charu C Aggarwal, and Jiawei Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *Advances in Social Networks Analysis and Mining*, 2011.

Yizhou Sun, Jiawei Han, Charu C. Aggarwal, and Nitesh V. Chawla. When will it happen?: Relationship prediction in heterogeneous information networks. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, 2012.

Andreas Tösscher, Michael Jahrer, and Robert M. Bell. The bigchaos solution to the netflix grand prize. Technical report, 2009.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1998.

Kuan-Wei Wu, Chun-Sung Ferng, Chia-Hua Ho, An-Chun Liang, Chun-Heng Huang, Wei-Yuan Shen, Jyun-Yu Jiang, Ming-Hao Yang, Ting-Wei Lin, Ching-Pei Lee, Perng-Hwa Kung, Chin-En Wang, Ting-Wei Ku, Chun-Yen Ho, Yi-Shu Tai, I-Kuei Chen, Wei-Lun Huang, Che-Ping Chou, Tse-Ju Lin, Han-Jay Yang, Yen-Kai Wang, Cheng-Te Li, Shou-De Lin, and Hsuan-Tien Lin. A two-stage ensemble of diverse models for advertisement ranking in KDD cup 2012. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2012. URL `http://www.csie.ntu.edu.tw/~htlin/paper/doc/wskdd12cup.pdf`.

Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 2010.

Yang Yang, Nitesh V. Chawla, Yizhou Sun, and Jiawei Han. Link prediction in heterogeneous networks: Influence and time matters. Technical report, 2012.

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G. McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, Jui-Yu Weng, En-Syu Yan, Che-Wei Chang, Tsung-Ting Kuo, Yi-Chen Lo, Po T. Chang, Chieh Po, Chien-Yuan Wang, Yi-Hung Huang, Chen-Wei Hung, Yu-Xun Ruan, Yu-Shi Lin, Shou-De Lin, Hsuan-Tien Lin, and Chih-Jen Lin. Feature engineering and classifier ensemble for KDD cup 2010. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/courses/dmcase2010/kdd2010ntu.pdf`.

Xiao Yu, Quanquan Gu, Mianwei Zhou, and Jiawei Han. Citation prediction in heterogeneous bibliographic networks. In *SIAM International Conference on Data Mining*, 2012.