A profile-based solution for KDD Cup 2013's author identification problem

Carlos COLMENARES
Yandi LI

June 4, 2013

Abstract

This report contains a description of the techniques used to solve the problem proposed for the KDD Cup 2013, which consists chiefly in determining whether an author wrote a given paper. The methodology used was based mainly in creating a trustable user-profile by bootstrapping means, so when it is desired to check whether an author wrote a paper the characteristics of this paper are matched against the author's profile and a set of features are extracted, then a classifier was trained for deciding if the article was written by the author or not. The report is organized as follows: in the first section the problem is introduced, in the second section the provided data is explained and also the data preparation techniques applied to it, in the third and fourth sections the methodology used for solving the problem and obtained results are presented respectively, the final section contains the conclusions and possible extensions of the work.

1 Problem description

The problem we choose to solve comes from an annual data mining competition, KDD Cup 2013[1], organized by ACM SIGKDD[2]. This year, the competition is sponsored by Microsoft Research and hosted on Kaggle[3], the world's leading platform for predictive modeling competition.

The featured dataset is provided by the Microsoft Academic Search, Microsoft's free academic search engine that covers 50 million publications and over 19 million authors across a variety of domains. The general mission is to automate authorship disambiguation of academic papers. The problem is divided into two tracks. Track 1[4] is to design an algorithm that will accurately confirm or deny which papers are written by a particular author; and Track 2[5] is to determine which authors in a given data set are duplicates. We choose to only work on Track 1.

The major challenge lies in the large number of data sources, the limited number of metadata[6] and the amount of noise in publication data. Results can be easily uploaded to Kaggle platform, tested with

[1] Knowledge Discovery and Data Mining
[2] Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining
[3] http://www.kaggle.com/
[4] http://www.kaggle.com/c/kdd-cup-2013-author-paper-identification-challenge
[5] http://www.kaggle.com/c/kdd-cup-2013-author-disambiguation
[6] The rule says that no other external datasets are allowed, since the spirit of the competition is that any advantage is made from modeling, not from enriched data. [Kaggle]

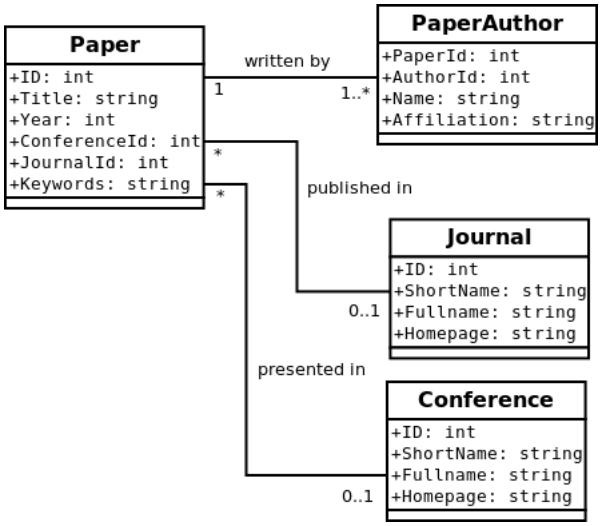

Figure 1: Schema of the featured dataset.

certain evaluation criterion and compared with the results given by other participants.

2 Data description and preparation

The data provided for the challenge was supplied by Microsoft Corporation, and it is an excerpt of the Microsoft Academic Search database. The dataset consists basically of descriptions of several papers and their co-author graph.

Figure 1 contains a description of the schema, the relations that comprise it are the following ones:

- **Paper:** contains roughly 2.2 million entries. The information in the table is noisy or incomplete: only around 25% of the articles have keywords, some papers have an erroneous year assigned (e.g. a year in the future) or no year at all, some papers have meaningless keywords (e.g. "reply"), around 24% of the papers are not associated with a journal or conference, etc.

- **PaperAuthor:** contains the list of authors for each paper. Once again, the information is noisy: the table contains inconsistent data since the same author can appear under the same author-ID but with different names, or an author could have several related author-IDs, or two completely different authors could share the same author-ID. The table contains about 2.1 million different author-IDs, and 65% of its entries do not have a related affiliation.

- **Journal:** contains information about 15,151 different journals.

- **Conference:** contains information about 4,545 different conferences.

For the task to be performed in the cup, a very small subset consisting of around 5 thousand authors was extracted from the previous dataset, for each of these authors a true name is provided together with a list of most of the papers where his/her author-ID appears in the PaperAuthor table. In other words, a very small set of authors with potential publications is extracted from the given database.

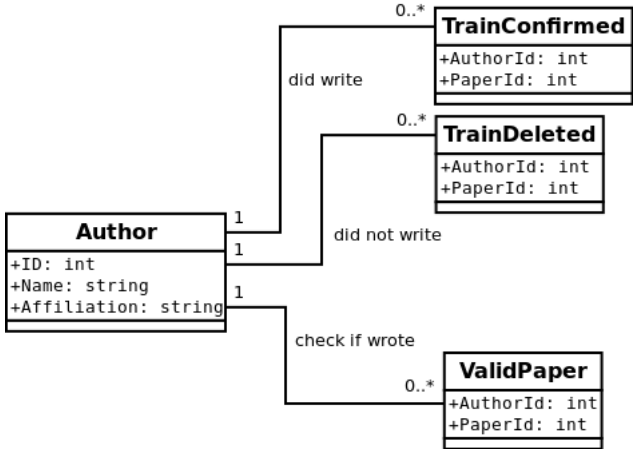

Figure 2: Schema of the training and validation dataset.

This set of authors with potential publications is afterwards divided in two subsets: one for training purposes (70% of the authors) and one for validation purposes (30% of the authors). The subset for training was annotated by hand: each of the papers in an author's potential publication list was marked as confirmed (i.e. the author wrote it indeed) or deleted (i.e. the author did not write it). Therefore the cup's challenge consists in annotating in a similar fashion the subset that was left for validation.

Figure 2 contains the schema of the provided data, the meaning of the entities in it is the following one:

- **Author:** contains information about roughly 250 thousand authors, among them the 5 thousand authors extracted for training/validation purposes. It should be assumed that the name and affiliations of the authors in this table is a ground truth, however, only 20% of these selected authors have a related affiliation.
- **TrainConfirmed and Traindeleted:** these tables contain information about 70% of the 5 thousand authors extracted from the database. The former table contains the IDs of the papers that an author indeed wrote, while the latter table contains the IDs of the papers that he did not write although they are listed as so in the PaperAuthor table.
- **ValidPaper:** contains 1,496 different author-IDs and for each a list of papers that the author potentially wrote. The cup's challenge consist in predicting whether each of these papers was indeed written by the author or it is some sort of mistake and s/he was not the one who wrote it.

2.1 Data cleaning

As mentioned before, the data given for the task is highly noisy. Therefore the following cleaning tasks were performed but only on the group of 5 thousand authors in the set for training/validation, and the 300 thousand papers they were related to.

- **author and co-author names:** many of the names had UTF-8 characters (e.g. chinese characters or special accents). Therefore all the names were "romanized" to ASCII lowercase characters and

non-alphabetic characters such as points and hyphens were replaced by spaces.

- **paper title:** the title of the papers were also "romanized" and the non-alphanumeric characters were replaced by spaces.
- **paper years:** some papers had as year a number between 50 and 99, so it was added 1900 to these years (e.g. if a paper was marked as been published on the year 82, it was replaced by 1982). Then all papers that had a year smaller than 1800 or greater than 2013 were assigned a null year (equal to zero).
- **paper keywords:** many of the few papers with keywords had noise on them, for example variations of the words "keyword" would appear, or had separators such as commas or colons. Therefore the field was cleaned by erasing these non-alphabetic characters and variations of the term "keyword" (e.g. key-words).
- **paper journal and conference:** some papers had assigned as conference/journal an ID not present in the Journal/Conference table. In these cases the ID was assigned by a null one (equal to zero).

2.2 Data preprocessing

One of the major problems we face is how to generate semantic meanings out of the textual attributes as titles and keywords of papers. We know that each paper belongs to a certain scientific field and an author of his papers can be characterized by the set of fields s/he is active in. However, such information does not exist in the original database. So is it possible to restore the scientific fields for each paper using its textual information (i.e. title and keywords)?

The solution is Latent Dirichlet Allocation (LDA) [Blei et al., 2003], one of the major topic extraction techniques in handling text corpus. LDA is a generative probabilistic model, in which each document of the corpus is modeled as a finite mixture over an underlying set of topics, and each topic is a probability distribution over the underlying set of words. In our case, the text corpus is the collection of title and keywords; whereas the extracted topics corresponds to the hidden scientific fields we are looking for. The topic distribution thus provides an explicit representation of the topic profile of each paper.

The procedure of building topic profile is as follows:

1. For each paper in the Paper table, concatenate the text of title and keyword, remove empty items, and then perform the aforementioned data cleaning procedure.
2. Remove English and French stopwords from the resulting title-keyword document.
3. Perform Porter stemming over the documents [Kurt Hornik, 2013].
4. Use lda package in R [Jonathan Chang, 2012] which uses Gibbs sampling to perform inference in the model. To initialize the parameters, we used $k = 30$ topics, $\alpha = 0.1$ and $\eta = 0.01$, and we ran the sampler for 50 iterations. The parameters were tuned with respect to 30 targeted documents, in order to offset the skewness of topic distribution and to keep an appropriate purity within each document.

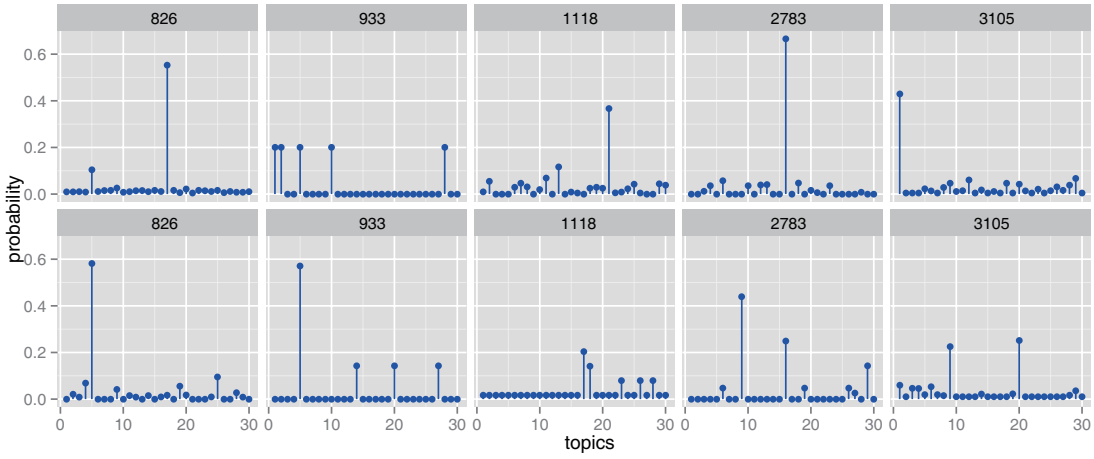

Figure 3: Topic distribution of the first five authors. The row above corresponds to the confirmed papers; the row below corresponds to the denied papers.

A selection of the resulting topic distribution of the first 5 authors in the training set is given in Figure 3. The row above is the average topic distribution aggregated over papers one confirmed; the row below is the distribution over papers one denied. Note that the distribution for the confirmed paper are rather homogeneous, with less than three peaks[7]. It is clear that the topic profile is a feature that can effectively differentiate papers one truly writes from those not, provided we know the ground truth.

3 Method used

In general terms, for solving the problem a profile was built for each of the 5 thousand authors under analysis, then for each author-paper tuple in the training/validation set several features that describe each tuple were extracted, finally a classifier was fit using the annotated training set and it was used to predict the validation one. However, then a bootstrap technique was used to refine the author profiles and re-start the process. This section explains in detail this process, which is divided in three phases: (1) initial profile construction, (2) feature selection and classification, and (3) bootstrap.

3.1 Initial profile construction

The profile of a user consists in a set of papers that s/he wrote, however this set must not contain noise (i.e. papers that the author did not write) or a very small amount of it (e.g. less than 1%). The reason for this is that the decision of whether an author wrote a paper or not is based on similarity measures between an article and an author's profile, therefore the noise in this set will directly affect the classification process.

For building an initial profile for each of the 5 thousand authors under evaluation, the list of papers s/he potentially wrote is filtered and only the ones that contain a co-author with a name almost identical

[7]Author 933 has only one paper confirmed and one denied, thus the distribution has more peaks than the others.

to the one of the objective author are left. The procedure that evaluates if two names are almost identical is as follows:

1. The two (cleaned) names are tokenized into set of words.
2. The set of words with the least tokens is taken as a reference.
3. For each token in the set an identical match is searched in the other set. Two words match if and only if:
   (a) One of them is an initial and the first letter of the other is the same.
   (b) They are both identical strings.
   (c) The edit distance [Wikipedia] of both words is less or equal to 10% of the length of the largest word. This percentage was chosen arbitrarily.
4. If it is possible to match all the words in the reference set with a different one in the other set, then it is assumed that both words are identical or almost identical.

For example, the (cleaned) names "juan pablo rossi" and "j rossi" are considered as almost identical since the match "j"-"juan" and "rossi"-"rossi" is possible. But the names "steve black" and "stephen black" are not because the edit distance between "steve" and "stephen" is equal to 3, which represents around 42% of the longest word's length.

From the set of papers that is strongly believed that an author wrote, the following characteristics are extracted to conform his/her profile:

- Distribution of the non-null years of the published papers. This includes standard deviation, mean, non-null amount, null amount, and number of publications per year.
- Number of publications per journal (including null).
- Number of publications per conference (including null).
- Co-author network: number of publications in common with other authors.
- Co-author affiliations: affiliations of co-authors and times published.
- All keywords used by the author, and number of them.
- Average topic distribution of all the papers.
- Number of total publications.

As a summary, the initial profile of an author is built by first fetching all the papers where his/her author-ID is listed, this set of potential papers will be referred to as the "original publication set". Then this publication set is filtered by removing all the papers where there is no co-author with an identical or almost identical name as the one of the author under evaluation. Then the remaining list, referred as the "filtered publication set" is used then for building an authors' profile. This process is depicted in Figure 4.

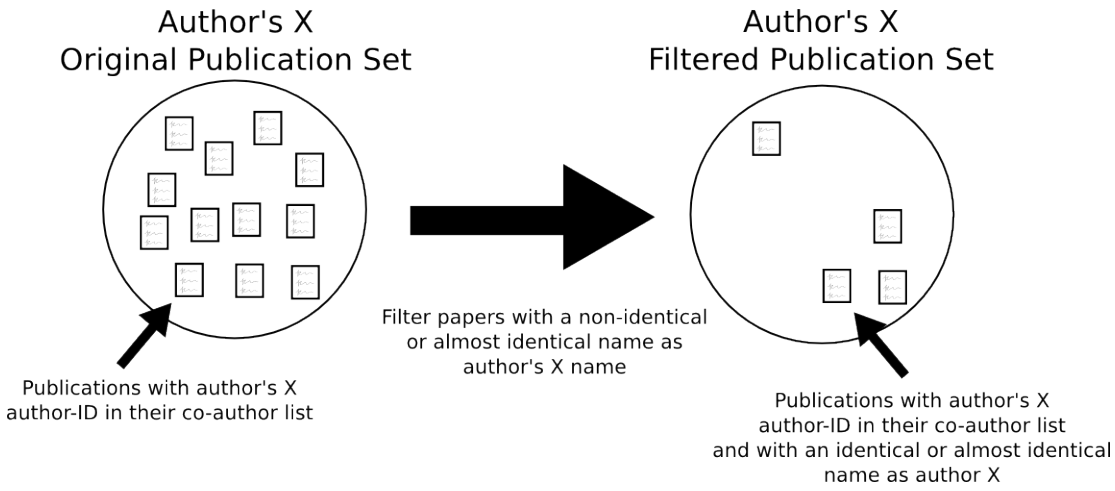

Figure 4: Building filtered publication set

3.2 Feature selection and classification

Once all author profiles are built, for each Paper-Author tuple in the training/validation set a group of features are extracted for describing the pair. These features will make a general description of the similarity between the paper under analysis and an author's profile, or in other words, the features try to describe how similar the paper under analysis is with respect to a set of papers the author wrote.

The extracted list of features can be split in three groups: author-only, paper-only, and author-paper. The first group contains features that depend solely on the author's profile and thus have no direct relation with the paper under analysis. The second group contains features that only concern the paper under analysis and have no direct relation with the author's profile. The third group contains features that concern both entities and measure directly their similarity. The specific features contained in each of the groups are explained in the following subsections:

Author-only features:

These features are extracted directly from an author's profile independently of the specific paper under analysis.

- Mean and standard deviation of the non-null publication years.
- Number of null and non-null publication years.
- Number of papers in the profile published in a non-null conference or journal.
- Number of publications in the profile.
- Number of different used keywords.

Paper-only features:

These features are extracted directly from the paper under analysis independently of the author's profile against which it will be matched.

- Five boolean features, indicating respectively whether the paper's title, year, conference-ID, journal-ID, and keywords are empty or null.
- Paper's publication year.
- Number of keywords in the paper.
- Number of listed authors in the article.

Author-paper features:

These features measure directly the similarity between the author's profile and the paper under analysis.

- Number of publications the author has made with any of the paper's co-authors.
- Number of publications the author has made with other authors that have a similar affiliation as the paper's co-authors.
- Number of publications the author has made on the same journal as the paper.
- p-value of a chi-square test between the paper's topic distribution and the author's average topic distribution.
- Number of keywords in the paper that the author has ever used.
- Number of co-authors in the paper with a name almost identical as the author's.
- Number of times the author's ID is marked as an author in the paper (sometimes the same author can be listed several times).

The author-only and paper-only features prove to be very useful as reference points for the Author-Paper ones. For instance, the number of keywords in the paper the author has already used (Author-Paper feature), together with the total number of keywords in the paper (Paper-only feature) and the total number of keywords used by the author (Author-only) proved to yield good results together, better than if only a subset of them was used.

Classification and cross-validation:

Once all the features were extracted (a total of 24), a random forest classifier with 100 predictors was fit with the training set, the target variable to predict was chosen as a boolean variable with value 0 when the author did not write the article and 1 when he did write it indeed. Then this classifier is used to predict the target variable for the validation set.

For testing purposes, a 5-fold cross-validation is made before fitting the classifier, the average accuracy obtained in the cross-validations is equal to 99.41%, which indicates a very good performance of the classifier.

3.3 Bootstrap

The main problem with the procedure explained before is that it relies completely in the correctness of the author profiles for making initial predictions. Since an author profile is composed of set of publications, if this set is incomplete or erroneous then many of the articles s/he wrote will not be classified as so, or even worse, the classifier will get biased while teaching it to classify positively Paper-Author tuples where the paper has a very low relation with an author's profile (due to incompleteness or noise in the profile).

For solving this problem, a bootstrap technique was developed. The strategy consists in building an initial user profile, training the classifier, and then using the same classifier for re-calculating the user's profile. The process is repeated a fixed number of times and then the predictions on the validation set are made.

Specifically, the bootstrap procedure is as follows:

1. Build initial author profiles as described in section 3.1. Recall that this process takes the "original publication set" of each authors and calculates a "filtered publication set" by removing publications where no co-author name is identical or almost identical as the author whose profile is being built.
2. Using the filtered publication set for each author, build their profiles.
3. For each Author-Paper tuple in the training set, extract features and train the classifier as described in section 3.2.
4. If it is desired to perform bootstrap, go to step 5. Else go to step 7.
5. Use the classifier for filtering the "original publication set" of each author, then a new "filtered publication set" will be obtained per author.
6. Go back to step 2 but use the new "filtered author profiles" for building the profile.
7. Extract features from Author-Paper tuples in the validation set and use the classifier for predicting the results. This ends the process.

All the bootstrap process can be depicted in Figure 5. Bear in mind that after each bootstrap the articles that conform the "filtered publication set" of an author will change probably, normally augmenting the number of papers in the set.

The effect of the bootstrap process is that the set of publications that conforms an author's profile will gradually get improved, yielding every time better results.

4 Results

The accuracy of the model was directly measured by Kaggle, for this end the final predictions for the ValidPaper table had to be written in a file with a special format and sent to the website which would compare the file with its ground truth and provide a value for the quality of the uploaded result.

Table 1 contains a summary of how we managed to increase iteratively the model's precision, for each step it includes a brief explanation of the work performed for improving the results. It is important

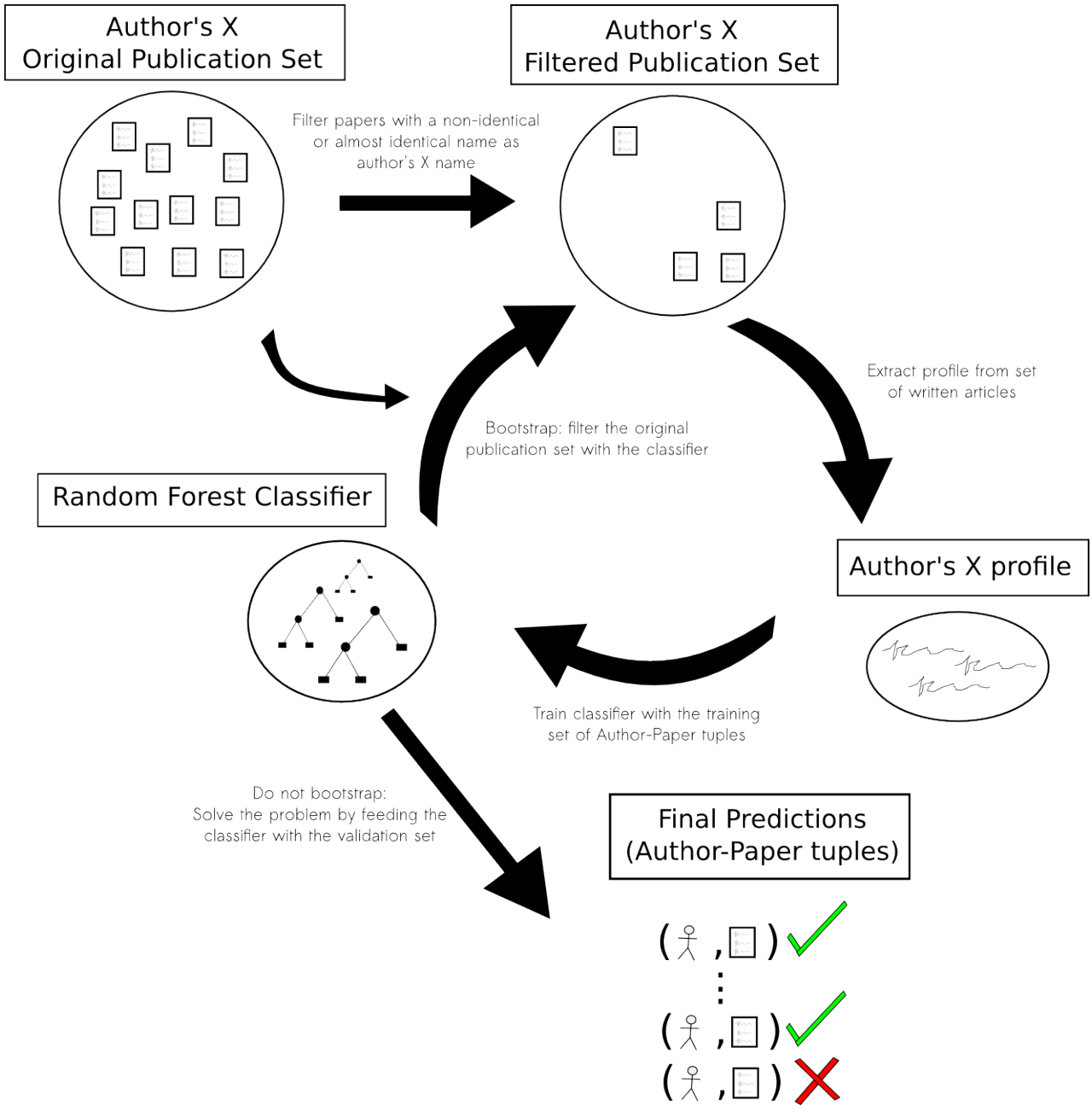

Figure 5: bootstrap process

Accuracy	Improvement
0.76816	First rough model. Uses the whole original publication set for building profiles. Only four features extracted: same journal/conference count, number of publications in the author's profile, number of coauthors in the paper, and number of common publications with co-authors.
0.80147	Performed all data cleaning operations on the database.
0.81892	Added features related to distribution of years in author's profile (mean, standard deviation, etc), added boolean features, count of author-id repetitions in the article.
0.95804	Performed the first filter of the original publication set for author profiles, as described in section 3.1 (initial profile construction).
0.96818	Added features related to keyword counts and comparison of co-authors affiliations with those in an author's profile.
0.96818	Calculated the topic models and added related features.
0.97084	Performed one-step bootstrap for improving the filtered publication set.

Table 1: Improvement

to mention that many experiments were performed and most of them did not generate positive results, therefore for simplicity reasons the table shown below only contains the evolutions of the method used for tackling the problem that yielded positive outcomes.

From the obtained results, two main points should be highlighted:

1. When the original publication set was filtered for the first time, a huge improvement was obtained (roughly 14%). This lead to the conclusion that there was a lot of noise in authors' profiles, and that the quality of such profiles made great impact on the overall performance of the method used for solving the problem.

2. Only a one-step bootstrap proved to yield positive results. The reason for this is that at each bootstrap step the training data was being overfit by the model, and the classifier would eventually only classify positively an Author-Paper tuple only if the paper belongs to the author's profile, which would disable the method to predict similarity between an article and a profile (since it will only evaluate inclusion).

5 Conclusions and future work

Several techniques were used for solving the KDD Cup author identification problem, the main strategy used was related to build iteratively author profiles, and training a classifier for assessing an Author-Paper similarity. The main drawbacks of the proposed solution are the following ones, and should be taken into consideration for further improvements of the work:

- **Author ambiguity problems:** the model builds an initial profile in based on name similarity. However it turns out that it can happen that two different authors have very similar names, there-

fore the profile would represent two different persons which would generate further classification errors.

- **Naive affiliation match:** one of the Author-paper features described in section 3.2 is related to the co-authors in the paper with the same affiliation as the author under analysis. The problem is that two affiliations are considered as similar if their string is identical: this should be relaxed for allowing small variations on the affiliation.
- **Hyphens in names:** it happened often in the set that asiatic author names had variations with or without a hyphen in the middle, for instance "Zhi-Ming" and "ZhiMing". In these cases both names are not classified as almost identical by the implemented algorithm, although they should. This problems hinders the performance of the whole process.
- **Classifier diversity:** the only classifier used was a random forest one, however, it should be tested if a blend of classifiers such as naive bayes, SVM, and others would improve the results.
- **Alternative topic modeling tools:** the algorithm performs basic LDA to extract topic out of title and keywords. However, some extended version are also possible which may produce better result. These models include Correlated Topic Model, which drop the assumption of independence of topics in each document; and Relational Topic Model, which generate topics according to the linkage relationships between documents.

References

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

Jonathan Chang. *lda: Collapsed Gibbs sampling methods for topic models.* CRAN-R project, 1.3.2 edition, 14 October 2012.

Kaggle. http://www.kaggle.com/c/yelp-recruiting/forums/t/4117/clarification-request-the-usual-questions-about-text-mining-and-rules-about.

Kurt Hornik. *Snowball: Snowball Stemmers.* CRAN-R project, 0.0-9 edition, 19 March 2013.

Wikipedia. http://en.wikipedia.org/wiki/Edit_distance.