KDD Cup 2013 - Author-Paper Identification Challenge: Second Place Team

Dmitry Efimov
Moscow State University
Vorobievy Gory, 1
Moscow, Russia
+74959391801
diefimov@gmail.com

Lucas Silva
PWH
Rua Lavras 894/101
Belo Horizonte, Brazil
+553199258791
lucas.eustaquio@gmail.com

Benjamin Solecki
175 S Madison Ave 12
Pasadena CA, 91101
+16263765013
bensolucky@gmail.com

ABSTRACT

This paper describes our submission to the KDD Cup 2013 Track 1 Challenge: Author-Paper Indentification in the Microsoft Academic Search database. Our approach is based on Gradient Boosting Machine (GBM) of Friedman ([5]) and deep feature engineering. The method was second in the final standings with Mean Average Precision (MAP) of 0.98144, while the winning submission scored 0.98259.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Algorithms, feature engineering, collaborative filtering, ensembling, decision trees, cross-validation

1. INTRODUCTION

The goal of the Author-Paper Identification Challenge was to predict whether an author has written a paper assigned to them in the Microsoft Academic Search database (the detailed description of the dataset can be found in [9]). Thus, the task was a collaborative filtering task with a set of labeled training data and an unlabeled set of test data. We interpreted this problem as a supervised machine learning problem with a binary target function (1 - if a paper is written by given author, 0 - otherwise). The huge size of the provided database did not allow the of use many existing methods. Our approach contains two main steps:

1. Deep features engineering.
2. Using the decision trees ensembling method named Gradient Boosting Machine (GBM) ([5]).

Note that boosting ensembling was proposed by Freund and Schapire [4]. It builds base learners sequentially while emphasizing the hardest examples. It is achieved by the following weight updating rule. A set of weights is maintained over the training data. Examples which have been misclassified by the previous learner get their weights increased in the next iteration. Consequently, harder examples possess larger weights and have a higher possibility of being selected for the current training [10].

Additional complexity in this challenge arose from the error criterion (Mean Average Precision, or MAP) chosen by the organizers.

2. METHOD

2.1 Feature classification

To generate features we looked at an author-paper graph constructed from the dataset (fig. 1). The dataset can be described in terms of a bipartite graph where each node represents an author or paper and each edge represents whether an author has written a given paper. Thus, the problem can be formulated as follows: define false edges in a bipartite graph of authors and papers. Based on this graph we have introduced the following classifications for generated features:

1. Author-based: calculated for authors only.
2. Paper-based: calculated for papers only.
3. Author-paper-based: calculated for author-paper pairs.

Features of the first type can be calculated only for individual authors and those of the second type only for individual papers. Features of the last type can be calculated only for author-paper pairs.

The author-based and paper-based features can be classified as either primary or secondary features. To show the difference between primary and secondary types we consider an example: suppose we have calculated an author-based feature "count papers" (number of published papers) and a paper-based feature "count keywords" (number of keywords in paper). Based on these two we can calculate a secondary author-based feature "count author's keywords" by calculating "count keywords" for all papers of the given author and taking the sum. The first two features are primary features, derived directly from the dataset, while the last feature is a secondary feature generated from two primary features of different types.

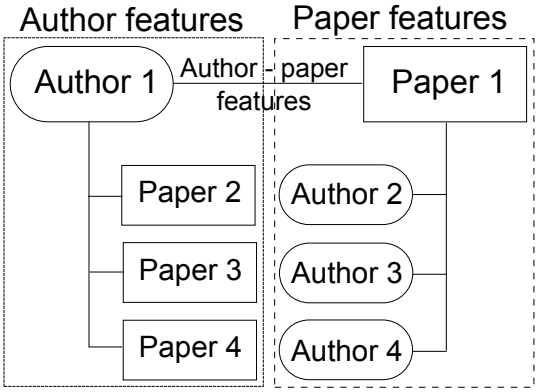

Figure 1: Author-paper graph for feature engineering.

2.2 Feature engineering: author-based features

We have divided author-based features into three groups:

1. **Count features.**

   Examples:
   - count of different journals (conferences) where author published;
   - count of all papers of the author;
   - count of all coauthors;
   - count of all keywords or distinct keywords in the author's papers.

2. **NLP features.**

   The titles and keywords of many papers were given in Microsoft's dataset, allowing us to use Natural Language Processing (NLP) for feature generation. For each paper we united the keyword and title field and removed duplicated and rare (occurring in less than 30 papers) words. All author-based NLP features were based on the term frequency - inverse document frequency (tf-idf) measure ([8]). The term frequency (tf) measure is the number of times each term (word) occurs in each document. The inverse document frequency (idf) measure is the weight of each term with respect to its frequency in the whole set of documents. And so the tf-idf measure is the sum of all terms of the tf measure multiplied by the idf measure. We had to modify the definition of the tf measure because in the vast majority of papers each word occurred only one time in the title and keyword field. We give formal definitions of tf, idf and tf-idf measures below:

   Definition 1. $\mathrm{tf} = \frac{1}{N_p}$, where N_p is a number of words in the paper.

   Definition 2. $\mathrm{idf} = \log\left(\frac{N}{N_w}\right)$, where N is a number of papers, N_w is a number of papers where the word is occured.

   Examples of author-based NLP features:
   - tf-idf measure by all author's keywords with respect to the author's papers (N is the number of author's papers);
   - tf-idf measure by all author's keywords with respect to all Microsoft dataset papers (N is the total number of papers in the dataset).

3. **Multiple source features.** We noticed that in the Microsoft dataset there are a lot of duplicated records (with the same author id and paper id). We had interpreted this as being the result of multiple sources from which Microsoft obtained their data. Further in the paper we will discuss features calculated from such sources of information. In our approach to feature engineering we also used the fact that there are some duplicated authors (the same author with multiple ids) and duplicated papers (the same paper with multiple ids). To find duplicated authors we used our algorithm from the "KDD Cup 2013 Track 2 - Author Disambiguation" challenge (see our paper from Track 2 with algorithm description). To find duplicated papers we grouped all papers based on title (when the title was not empty) and assigned the minimum id for all papers in a group. We refer to such duplicated authors and papers as author duplicates and paper duplicates.

   Examples of author-based multuple source features.
   - number of author duplicates for each author;
   - number of times papers are missing among all papers of the author and the author's duplicates (if the paper assigned to the author we looked at all author's duplicates and checked how many author's duplicates do not have this paper assigned to them);
   - binary feature (1: if current author id is the highest (lowest) of all the author's duplicate ids).

2.3 Feature engineering: paper-based features

Paper-based features can be classified as follows:

1. **Count features.**

   Examples:
   - count of authors of the paper;
   - count of papers in the same journal (conferences);
   - count of authors in the same journal (conferences);
   - count of keywords in the paper.

2. **NLP features.**

   Examples of paper-based NLP features:
   - tf-idf measure of paper's keywords with respect to all journal's (conferences's) papers (N in Definition 2 is a number of papers in the same journal (conference));
   - tf-idf measure of paper's keywords with respect to all Microsoft dataset papers (N in Defintion 2 is the total number of papers in the dataset).

3. **Multiple source features.**

   Examples:
   - binary feature (1: if the current paper id is the highest (lowest) of all the paper's duplicate ids);
   - number of (distinct) paper duplicates.

4. **Additional features.**

   This class of features was obtained by a method of reverse feature engineering. The idea of this method is

the following:

(a) Make a cross-validation prediction on the training set using chosen algorithm (in our case, GBM).
(b) Extract samples of training set which are predicted incorrectly.
(c) Construct different statistics based on the incorrectly predicted samples.
(d) Create features that can classify "bad" samples to the correct class.

Example of additional features:
- paper type (0: if the paper was published to a conference, 1: if paper was published in journal);
- year of paper;
- binary feature (1: if keywords (title, year) field for the paper is not empty);
- categorical feature: discretized ratio of the number of papers in correct conferences (with positive conference id) to the total number of papers;
- categorical feature: discretized ratio of the number of papers in correct journals (with positive journal id) to the total number of papers.

2.4 Feature engineering: author-paper-based features

In contrast to author-based and paper-based features which describe the nodes of the author-paper bipartite graph, the author-paper-based features describe the graph edges. As the constructed model is aimed to predict false edges, this group of features was the most important in our model. We classified them as follows:

1. **Mutliple source features.**

   Examples:
   - number of times the author-paper pair appeared in the Microsoft dataset (based on this feature we constructed a similar feature from only the pairs where the author had a non-empty affiliation);
   - number of (distinct) repeated words from author's name among the other authors of the paper;
   - target of author's duplicates (using KDD Cup Track 2 results we looked for the author's duplicate as a coauthor and checked if the target variable is given in the training data for this duplicate; the value of the feature is this known target value);
   - number of author-paper records for all of the author's and the paper's duplicates.

2. **Count features.**

   Examples:
   - number of papers in the same journal (conference) written by authors of the same affiliation;
   - number of papers with the same coauthors (we also calculated this feature based on papers with non-empty keywords and based on a dataset with paper duplicates removed);
   - number of papers written by the coauthors where the author's duplicates was not one of the authors;
   - number of keywords in papers written by the coauthors;
   - number of common words in the websites or titles of the journals (conferences) (this feature is calculated based on all papers of all coauthors, we also used a cleaning procedure to extract meaningful pieces of journals' and conferences' websites);
   - number of author's papers of the same year;
   - number of unique author's keywords (keywords that appeared only in that paper);
   - number of frequent author's keywords (keywords that appeared in other papers of the author);
   - number of coauthors with the same affiliation;
   - number of papers of the same author in the same journal (conference) (modifications of this feature included counting the number of papers with non-empty titles, non-empty keywords or correct year of publication).

3. **Likelihood features.**

   Definition 3. The likelihood ratio of target variable y with respect to feature x_i is a probability function

   $$P(\theta) = P(y = 1 | x_i = \theta),$$

   where $\theta \in \Theta$ and Θ is the set of possible values of feature x_i.

   The initial idea was to use likelihood ratios to generate new features for our machine learning algorithm. Unfortunately, this approach does not work in many cases (particularly, if there are a lot of samples with unique values x_i). An example of successfull application of likelihood ratios can be found in [2].

   Let A be the set of authors, Ω be the set of values of journals (optionally conference, year or affiliation). To construct likelihood features and avoid overfitting we generated the shrunken likelihood. The main idea of this method is to find likelihood in the form:

   $$P^s_{\omega,a} = \alpha P_\omega + (1 - \alpha) P_{\omega,a}, \quad (1)$$

   where $\omega \in \Omega$, $a \in A$, P_ω is a global likelihood of papers in the journal (total confirmed rate in the journal), $P_{\omega,a}$ is an author-journal likelihood (confirmed rate of author's papers in the journal), $P^s_{\omega,a}$ is a shrunken likelihood. This equation (1) can be transformed to:

   $$P^s_{\omega,a} = P_{\omega,a} + \alpha(P_\omega - P_{\omega,a}). \quad (2)$$

   To find the coefficient α we applied a mixed-effects model with likelihood $P_{\omega,a}$ as a target variable (lme4 package in R). The detailed description of mixed-effects models can be found in [6]. To use the mixed-effects model we have to rewrite formula (2) as follows:

   $$P_{\omega,a} = E(P_{\omega,a}) + \alpha(P_\omega - P_{\omega,a}), \quad (3)$$

   where $E(P_{\omega,a})$ is a mean of $P_{\omega,a}$. Then the second term in the right-hand side of formula (3) represents random effects which can be predicted using mixed-effects model. As we need to predict α in formula (3) the coefficients $(P_\omega - P_{\omega,a})$ can be interpreted as weights in the mixed-effects model. To implement the described idea we used the following algorithm.

(a) Calculate log-likelihoods for each author $a \in A$ and journal $\omega \in \Omega$:

$$P^{pos}_{\omega,a} = \frac{N^{pos}_{\omega,a}}{N_{\omega,a}} * \log(1 + N_{\omega,a}),$$

$$P^{neg}_{\omega,a} = \frac{N^{neg}_{\omega,a}}{N_{\omega,a}} * \log(1 + N_{\omega,a}),$$

where $P^{pos}_{\omega,a}$, $P^{neg}_{\omega,a}$ are log-likelihoods, $N^{pos}_{\omega,a}$ is the number of confirmed papers of author a in journal ω, $N^{neg}_{\omega,a}$ is the number of deleted papers of author a in journal ω, and $N_{\omega,a}$ is a number of all papers of author a in journal ω.

(b) Calculate weights $W_{\omega,a} = P_{\omega} - P_{\omega,a}$ for the mixed-effects model (3):

$$W_{\omega,a} = (P^{pos}_{\omega} - P^{pos}_{\omega,a}) + (P^{neg}_{\omega} - P^{neg}_{\omega,a})$$

In this formula $P^{pos}_{\omega} = \sum_{a \in A} P^{pos}_{\omega,a}$, $P^{neg}_{\omega} = \sum_{a \in A} P^{neg}_{\omega,a}$.

(c) Calculate log-likelihood for journal ω and author a:

$$P_{\omega,a} = \frac{\sum_{a \in A} P^{pos}_{\omega,a} - P^{pos}_{\omega,a}}{W_{\omega,a}}$$

(d) Take unique pairs ($P_{\omega,a}$, $W_{\omega,a}$) and feed them to the linear mixed-effects algorithm (package lme4 in R) using the calculated $P_{\omega,a}$ as the target function and $W_{\omega,a}$ as weights. This gives the modified likelihood features.

4. **Additional features.**

Like section 2.3, this class of features was obtained by a reverse feature engineering method.

Examples:
- percentage of the paper's authors with the same affiliation;
- percentage of common affiliation words among the paper's authors;
- year rank feature (first year of publishing for the author is coded as 1, second year as 2 and so on);
- binary feature (1: if the author's names in all tables of Microsoft dataset are the same);
- binary feature (1: if the author's affiliations in all tables of Microsoft dataset are the same).

2.5 Mean Average Precision as a loss function

The organizers of the KDD Cup Track 1 challenge have chosen Mean Average Precision (MAP) as a loss function. The MAP metric is analogous to the Area Under Curve (AUC) metric for ranking problems. It can be defined as the average of areas under the precision-recall curve (average precision).

Definition 4. Average precision

$$AveP = \frac{\sum_{k=1}^{N} (P(k) \times rel(k))}{N_{pos}},$$

where N is the number of samples (author-paper pairs), N_{pos} is the number of confirmed samples, $P(k)$ is the precision at cut-off k, $rel(k)$ is an indicator function equal to 1 if the sample at rank k is confirmed, 0 otherwise.

2.6 Model

During many experiments we found that the gain from ensembling was minimal (less than 2%) in this challenge. Rather than ensembling various algorithms we decided to concentrate on deep feature engineering and use Gradient Boosting Machine (GBM) with Bernoulli distribution (package gbm in R) as our main algorithm. The detailed description of this method can be found in [5]. GBM algorithm uses a large number of small decision trees; the weights of each decision tree are found using the gradient descent optimization method. We chose our set of parameters (shrinkage, number of trees, interaction depth) by implementing grid search in the parameter space. Interestingly, the ranking algorithms we tried (such as LambdaRank [1], LambdaMART [7], RankBoost [3]) produced a worse MAP score than GBM with Bernoulli distribution.

3. CONCLUSIONS

Using a combination of deep feature engineering and GBM we finished second in the final standings with an MAP of 0.98144. The winning submission score 0.98259.

Our approach of deep feature engineering seemed to work very well for this problem and may be promising for other like it. The idea of feature classification based on the bipartite author-paper graph can give new ideas in feature engineering. One could also investigate the author-paper graph from the graph theory point of view. As an example, new features obtained from graph topology (by looking at different graph characteristics such as density or clustering coefficients) may ultimately improve prediction accuracy.

4. ACKNOWLEDGEMENTS

Thanks to Kaggle, Microsoft Academic Search and KDD for hosting, creating and supporting this competition.

5. REFERENCES

[1] C.J.C.Burges, R.Ragno, and Q. Le. Learning to Rank with Nonsmooth Cost Functions. In *NIPS*, pages 193–200, 2006.

[2] M. Diez, A. Varona, M. Penagarikano, L. Rodriguez-Fuentes, and G. Bordel. On the use of phone log-likelihood ratios as features in spoken language recognition. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 274–279, 2012.

[3] Y. Freund, R.Iyer, R.E.Shapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[4] Y. Freund and R. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *J. Comput. System Sciences*, 55:119–139, 1997.

[5] J. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001.

[6] A. Galecki and T. Burzykowski. *Linear Mixed-Effects Models Using R.* Springer, 2013.

[7] Q.Wu, C.J.C.Burges, K.M.Svore, and J.Gao. Ranking, boosting, and model adaptation, 2008.

[8] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing*

and Management, 24(5):513–523, 1988.

[9] S.B.Roy, M. Cock, V.Mandava, B.Dalessandro, C.Perlich, W.Cukierski, and B.Hamner. The microsoft academic search dataset and kdd cup 2013, 2013.

[10] S.Wang, H.Chen, and X.Yao. Negative correlation learning for classification ensembles. In *WCCI 2010, Barcelona, Spain*, pages 2893–2900. IEEE, 2010.