# Tri-Training for Authorship Attribution with Limited Training Data

**Tieyun Qian**
State Key Laboratory
of Software Eng.,
Wuhan University
430072, Hubei, China
qty@whu.edu.cn

**Bing Liu**
Dept. of Computer Science, Univ. of Illinois at
Chicago
IL, USA, 60607
liub@cs.uic.edu

**Li Chen**
State Key Laboratory of
Software Eng.,
Wuhan University
430072, Hubei, China
ccnuchenli@163.com

**Zhiyong Peng**
Computer School,
Wuhan University
430072, Hubei, China
peng@whu.edu.cn

## Abstract

Authorship attribution (AA) aims to identify the authors of a set of documents. Traditional studies in this area often assume that there are a large set of labeled documents available for training. However, in the real life, it is often difficult or expensive to collect a large set of labeled data. For example, in the online review domain, most reviewers (authors) only write a few reviews, which are not enough to serve as the training data for accurate classification. In this paper, we present a novel three-view tri-training method to iteratively identify authors of unlabeled data to augment the training set. The key idea is to first represent each document in three distinct views, and then perform tri-training to exploit the large amount of unlabeled documents. Starting from 10 training documents per author, we systematically evaluate the effectiveness of the proposed tri-training method for AA. Experimental results show that the proposed approach outperforms the state-of-the-art semi-supervised method CNG+SVM and other baselines.

## 1 Introduction

Existing approaches to authorship attribution (AA) are mainly based on supervised classification (Stamatatos, 2009, Kim et al., 2011, Seroussi et al., 2012). Although this is an effective approach, it has a major weakness, i.e., for each author a large number of his/her articles are needed as the training data. This is possible if the author has written a large number of articles, but will be difficult if he/she has not. For example, in the online review domain, most authors (reviewers) only write a few reviews (documents). It was shown that on average each reviewer only has 2.72 reviews in amazon.com, and only 8% of the reviewers have at least 5 reviews (Jindal and Liu, 2008). The small number of labeled documents makes it extremely challenging for supervised learning to train an accurate classifier.

In this paper, we consider AA with only a few labeled examples. By exploiting the redundancy in human languages, we tackle the problem using a new three-view tri-training algorithm (TTA). Specifically, we first represent each document in three distinct views, and then tri-train three classifiers in these views. The predictions of two classifiers on unlabeled examples are used to augment the training set for the third classifier. This process repeats until a termination condition is met. The enlarged labeled sets are finally used to train classifiers to classify the test data.

To our knowledge, no existing work has addressed AA in a tri-training framework. The AA problem with limited training data was attempted in (Stamatatos, 2007; Luyckx and Daelemans, 2008). However, neither of them used a semi-supervised approach to augment the training set with additional documents. Kourtis and Stamatatos (2011) introduced a variant of the self-training method in (Nigam and Ghani, 2000). Note that the original self-training uses one classifier on one view. However, the self-training method in (Kourtis and Stamatatos, 2011) uses two classifiers (CNG and SVM) on one view. Both the self-training and tri-training are semi-supervised learning methods. However, the proposed approach is not a simple extension of the self-training method CNG+SVM of (Kourtis and Stamatatos, 2011). There are key differences.

First, in their experimental setting, about 115 and 129 documents per author on average are used for two experimental corpora. This number of labeled documents is still very large. We consider a much more realistic problem, where the size of the training set is very small. Only 10 samples per author are used in training.

Second, CNG+SVM uses two learning methods on a single character n-gram view. In contrast, besides the character n-gram view, we also make use of the lexical and syntactic views. That is,

*Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 345–351,
Baltimore, Maryland, USA, June 23-25 2014. ©2014 Association for Computational Linguistics

three distinct views are used for building classifiers. The redundant information in human language is combined in the tri-training procedure.

Third, in each round of self-training in CNG+SVM, each classifier is refined by the same newly labeled examples. However, in the proposed tri-training method (TTA), the examples labeled by the classifiers of every two views are added to the third view. By doing so, each classifier can borrow information from the other two views. And the predictions made by two classifiers are more reliable than those by one classifier.

The main contribution of this paper is thus the proposed three-view tri-training scheme which has a much better generalization ability by exploiting three different views of the same document. Experimental results on the IMDb review dataset show that the proposed method dramatically improves the CNG+SVM method. It also outperforms the co-training method (Blum and Mitchell, 1998) based on our proposed views.

## 2 Related Work

Existing AA methods either focused on finding suitable features or on developing effective techniques. Example features include function words (Argamon et al., 2007), richness features (Gamon 2004), punctuation frequencies (Graham et al., 2005), character (Grieve, 2007), word (Burrows, 1992) and POS n-grams (Gamon, 2004; Hirst and Feiguina, 2007), rewrite rules (Halteren et al., 1996), and similarities (Qian and Liu, 2013). On developing effective learning techniques, supervised classification has been the dominant approach, e.g., neural networks (Graham et al., 2005; Zheng et al., 2006), decision tree (Uzuner and Katz, 2005; Zhao and Zobel, 2005), logistic regression (Madigan et al., 2005), SVM (Diederich et al., 2000; Gamon 2004; Li et al., 2006; Kim et al., 2011), etc.

The main problem in the traditional research is the unrealistic size of the training set. A size of about 10,000 words per author is regarded as a reasonable training set size (Argamon et al., 2007, Burrows, 2003). When no long documents are available, tens or hundreds of short texts are used (Halteren, 2007; Hirst and Feiguina, 2007; Schwartz et al., 2013).

Apart from the existing works dealing with limited data discussed in the introduction, our preliminary study in (Qian et al., 2014) used one learning method on two views, but it is inferior to the proposed method in this paper.

```
Input: A small set of labeled documents $L = \{l_1,\ldots, l_r\}$, a large
   set of unlabeled documents $U = \{u_1,\ldots, u_s\}$, and a set of test
   documents $T = \{t_1,\ldots, t_t\}$,
Parameters: the number of iterations $k$, the size of selected un-
   labeled documents $u$
Output: $t_k$'s class assignment

1  Extract views $L_c, L_l, L_s, U_c, U_l, U_s, T_c, T_l, T_s$ from $L, U, T$
2  Loop for $k$ iterations:
3      Randomly select $u$ unlabeled documents $U'$ from $U$;
4      Learn the first view classifier $C_1$ from $L_1$ ($L_1=L_c, L_l$, or $L_s$);
5      Use $C_1$ to label docs in $U'$ based on $U_1$($U_1=U_c, U_l$, or $U_s$)
6      Learn the second view classifier $C_2$ from $L_2$ ($L_2 \neq L_1$)
7      Use $C_2$ to label documents in $U'$ based on $U_2$ ($U_2 \neq U_1$);
8      Learn the third view classifier $C_3$ from $L_3$ ($L_2 \neq L_1, L_2$)
9      Use $C_3$ to label documents in $U'$ based on $U_3$ ($U_2 \neq U_1, U_2$);
10     $U_{p1} = \{u \mid u \in U', u.label$ by $C_2 = u.label$ by $C_3\}$;
11     $U_{p2} = \{u \mid u \in U', u.label$ by $C_1 = u.label$ by $C_3\}$;
12     $U_{p3} = \{u \mid u \in U', u.label$ by $C_1 = u.label$ by $C_2\}$;
13     $U = U - U', L_i = L_i \cup U_{pi}$ ($i$=1..3);
14 Learn three classifiers $C_1, C_2, C_3$ from $L_1, L_2, L_3$;
15 Use $C_i$ to label $t_k$ in $T_i$ ($i$=1..3);
16 Aggregate results from three views
```

Figure 1: The tri-training algorithm (TTA)

## 3 Proposed Tri-Training Algorithm

### 3.1 Overall Framework

We represent each document in three feature views: the character view, the lexical view and the syntactic view. Each view consists of a set of features in the respective type. A classifier can be learned from any of these views. We propose a three-view training algorithm to deal with the problem of limited training data. Logistic regression (LR) is used as the learner. The overall framework is shown in Figure 1.

Given the labeled, unlabeled, and test sets $L$, $U$, and $T$, step 1 extracts the character, lexical, and syntactic views from $L$, $U$, and $T$, respectively. Steps 2-13 iteratively tri-train three classifiers by adding the data which are assigned the same label by two classifiers into the training set of the third classifier. The algorithm first randomly selects $u$ unlabeled documents from $U$ to create a pool $U'$ of examples. Note that we can directly select from the large unlabeled set $U$. However, it is shown in (Blum and Mitchell 2008) that a smaller pool can force the classifiers to select instances that are more representative of the underlying distribution that generates $U$. Hence we set the parameter $u$ to a size of about 1% of the whole unlabeled set, which allows us to observe the effects of different number of iterations. It then iterates over the following steps. First, use character, lexical and syntactic views on the current labeled set to train three classifiers $C_1$, $C_2$, and $C_3$. See Steps 4-9. Second, allow two of these three classifiers to classify the unlabeled set *U’* and choose *p* documents with agreed labels. See Steps 10-12. The selected documents are then added to the third labeled set for the label assigned (a label is an author here), and the *u* documents are removed from the unlabeled pool *U’* (line 13). We call this way of augmenting the training sets *InterAdding*. The one used in (Kourtis and Stamatatos, 2011) is called *SelfAdding* as it uses only a single view and adds to the same training set. Steps 14-15 assign the test document to a category (author) using the classifier learned from the three views in the augmented labeled data, respectively. Step 16 aggregates the results from three classifiers.

### 3.2 Character View

The features in the character view are the character n-grams of a document. Character n-grams are simple and easily available for any natural language. For a fair comparison with the previous work in (Kourtis and Stamatatos, 2011), we extract frequencies of 3-grams at the character-level. The vocabulary size for character 3-grams in our experiment is 28584.

### 3.3 Lexical View

The lexical view consists of word unigrams of a document. We represent each article by a vector of word frequencies. The vocabulary size for unigrams in our experiment is 195274.

### 3.4 Syntactic View

The syntactic view consists of the syntactic features of a document. We use four content-independent structures including n-grams of POS tags (*n* = 1..3) and rewrite rules (Kim et al., 2011). The vocabulary sizes for POS 1-grams, POS 2-grams, POS 3-grams, and rewrite rules in our experiment are 63, 1917, 21950, and 19240, respectively. These four types of syntactic structures are merged into a single vector. Hence the syntactic view of a document is represented as a vector of 43140 components.

### 3.5 Aggregating Results from Three Views

In testing, once we obtain the prediction values from three classifiers for a test document $t_k$, an additional algorithm is used to decide the final author attribution. One simple method is voting. However, this method is weaker than the three methods below. It is also hard to compare with the self-training method CNG+SVM in (Kourtis and Stamatatos, 2011) as it only has two classifiers. Hence we present three other strategies to further aggregate the results from the three views. These methods require the classifier to produce a numeric score to reflect the positive or negative certainty. Many classification algorithms give such scores, e.g., SVM and logistic regression. The three methods are as follows:

1) *ScoreSum*: The learned model first classifies all test cases in *T*. Then for each test case $t_k$, this method sums up all scores of positive classifications from the three views. It then assigns $t_k$ to the author with the highest score.
2) *ScoreSqSum*: This method works similarly to ScoreSum above except that it sums up the squared scores of positive classifications.
3) *ScoreMax*: This method works similarly to the ScoreSum method as well except that it finds the maximum classification score for each test document.

## 4 Experimental Evaluation

We now evaluate the proposed method. We use logistic regression (LR) with L2 regularization (Fan et al., 2008) and the *SVM*$^{multiclass}$ (SVM) system (Joachims, 2007) with its default settings as the classifiers.

### 4.1 Experiment Setup

We conduct experiments on the IMDb dataset (Seroussi et al., 2010). This data set contains the IMDb reviews in May 2009. It has 62,000 reviews by 62 users (1,000 reviews per user). For each author/reviewer, we further split his/her documents into the labeled, unlabeled, and test sets. 1% of one author’s documents, i.e., 10 documents per author, are used as the labeled data for training, 79% are used as unlabeled data, and the rest 20% are used for testing. We extract and compute the character and lexical features directly from the raw data, and use the Stanford PCFG parser (Klein and Manning, 2003) to generate the grammar structures of sentences in each review for extracting syntactic features. We normalize each feature’s value to the [0, 1] interval by dividing the maximum value of this feature in the training set. We use the micro-averaged classification accuracy as the evaluation metric.

### 4.2 Baseline methods

We use six self-training baselines and three co-training baselines. Self-training in (Kourtis and Stamatatos, 2011) uses two different classifiers on one view, and co-training uses one classifier on two views. All baselines except CNG+SVM on the character view are our extensions.

*Self-training using CNG+SVM on character, lexical and syntactic views respectively:* This gives three baselines. It self-trains two classifiers from the character 3-gram, lexical, and syntactic views using CNG and SVM classifiers (Kourtis and Stamatatos, 2011). CNG is a profile-based method which represents the author as the *N* most frequent character *n*-grams of all his/her training texts. The original method applied only CNG and SVM on the character n-gram view. Since our results show that its performance is extremely poor, we are curious what the reason is. Can this be due to the classifier or to the view? In order to differentiate the effects of views and classifiers, we present two additional types of baselines. The first type is to extend *CNG+SVM* method to lexical and syntactic views as well. The second type is to extend *CNG+SVM* method by replacing *CNG* with *LR* to show a fair comparison with our framework.

*Self-training using LR+SVM on character, lexical, and syntactic views:* This is the second type extension. It also gives us three baselines. It again uses the character, lexical and syntactic view and *SVM* as one of the two classifiers. The other classifier uses *LR* rather than *CNG*.

*Co-training using LR on Char+Lex, Char+Syn, and Lex+Syn views:* This also gives us three baselines. Each baseline co-trains two classifiers from every two views of the character 3-gram, lexical, and syntactic views.

## 4.3 Results and analysis

### (1) Effects of learning algorithms

We first evaluate the effects of learning algorithms on tri-training. We use SVM and LR as the learners as they are among the best methods.

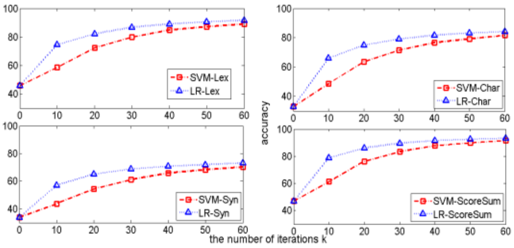

Figure 2. Effects of SVM and LR on tri-training

The effects of SVM and LR on tri-training are shown in Fig. 2. For the aggregation results, we draw the curves for ScoreSum. The results for other two stratigies are similar. It is clear that LR outperforms SVM by a large margin for tri-training when the number of iterations (*k*) is small. One possible reason is that LR is more tolerant to over-fitting caused by the small number of training samples. Hence, we use LR for tri-training in all experiments.

### (2) Effects of aggregation strategies

We show the effects of the three proposed aggregation strategies. Table 1 indicates that ScoreSum (SS) is the best.

| k | Single View Results | | | Aggregated Results | | |
|---|---|---|---|---|---|---|
| | Lex | Char | Syn | SM | SS | SQ |
| 0 | 45.75 | 32.88 | 33.96 | 41.11 | 46.85 | 44.61 |
| 10 | 74.63 | 66.05 | 56.99 | 73.41 | 78.82 | 76.41 |
| 20 | 82.30 | 74.92 | 65.05 | 81.63 | 86.19 | 84.05 |
| 30 | 86.86 | 79.12 | 68.85 | 85.29 | 89.69 | 87.74 |
| 40 | 89.16 | 81.81 | 70.85 | 87.83 | 91.52 | 89.99 |
| 50 | 90.56 | 83.14 | 72.06 | 89.11 | 92.58 | 91.17 |
| 60 | 91.69 | 84.13 | 73.23 | 90.05 | 93.15 | 91.82 |

Table 1. Effects of three aggregation strategies: ScoreMax(SM), ScoreSum(SS), and ScoreSq-Sum(SQ)

We also observe that both ScoreSum and ScoreSqSum (SQ) perform better than ScoreMax (SM) and all single view cases. This suggests that the decision made from a number of scores is much more reliable than that made from only one score. ScoreSum is our default strategy.

### (3) Effects of data augmenting strategies

We now see the effects of data adding methods to augment the labeled set in Fig. 3.

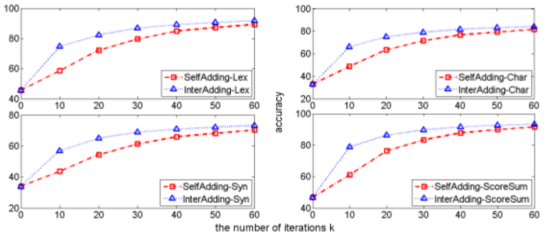

Figure 3. Effects of data augmenting methods on tri-training

We use two strategies. One is our *InterAdding* approach and the other is the *SelfAdding* approach in (Kourtis and Stamatatos, 2011), as introduced in Section 3.1. We can see that by adding newly classified samples by two classifiers to the third view, tri-training gets better and better results rapidly. For example, the accuracy for *k* = 10 iterations grows from 61.24 for SelfAdding to 78.82 for InterAdding, an absolute increase of 17.58%. This implies that by integrating more information from other views, learning can improve greatly.

### (4) Comparison with self-training baselines

We show the results of CNG+SVM in Table 2. It is clear that CNG is almost unable to correctly classify any test case. Its accuracy is only 1.26% at the start. This directly leads to the failure of the self-training. The reason is that the other classifier SVM can augment nearly 0 documents from the unlabeled set. We also tuned the parameter *N* for *CNG*, but it makes little difference.

| k | Self-Training on Char | | Aggregated Results | | |
|---|---|---|---|---|---|
| | CNG | SVM | SM | SS | SQ |
| **0** | 1.26 | 33.22 | 32.35 | 32.47 | 27.00 |
| **10** | 1.26 | 32.35 | 32.35 | 32.47 | 27.00 |
| **20** | 1.26 | 32.35 | 32.35 | 32.47 | 27.00 |
| **30** | 1.26 | 32.35 | 32.35 | 32.47 | 27.00 |
| **40** | 1.26 | 33.60 | 33.60 | 33.69 | 29.07 |
| **50** | 1.26 | 33.60 | 33.60 | 33.69 | 29.07 |
| **60** | 1.27 | 33.54 | 33.60 | 33.69 | 29.07 |

Table 2. Results for the *CNG+SVM* baseline

To distinguish the effects of views from classifiers, we conduct two more types of experiments. First, we apply *CNG+SVM* to the lexical and syntactic views. The results are even worse. Its accuracy drops to 0.58% and 1.21%, respectively. Next, we replace *CNG* with *LR* and apply *LR+SVM* to all three views. We only show their best results in Table 3, either on a single view or aggregation. The details are omitted due to space limitations. We can see significant improvements over their corresponding results of *CNG+SVM*. This demonstrates that the learning methods are critical to self-training as well.

| k | Tri Train | SelfTrain:CNG+SVM | | | SelfTrain:LR+SVM | | |
|---|---|---|---|---|---|---|---|
| | | Char | lex | Syn | Char | Lex | Syn |
| **0** | 46.85 | 33.22 | 45.44 | 34.50 | 33.22 | 45.75 | 34.48 |
| **10** | 78.82 | 32.47 | 45.44 | 34.50 | 62.56 | 73.78 | 51.94 |
| **20** | 86.19 | 32.47 | 45.44 | 34.09 | 71.21 | 81.44 | 59.88 |
| **30** | 89.69 | 32.47 | 45.44 | 34.09 | 75.21 | 84.68 | 63.70 |
| **40** | 91.52 | 33.69 | 45.44 | 34.09 | 77.46 | 88.25 | 65.74 |
| **50** | 92.58 | 33.69 | 45.44 | 34.09 | 78.64 | 88.25 | 67.45 |
| **60** | 93.15 | 33.69 | 45.44 | 34.09 | 79.54 | 89.31 | 68.37 |

Table 3. Self-training variations

From Table 3, we can also see that our tri-training approach outperforms all self-training baselines by a large margin. For example, the accuracy for LR+SVM on the lexical view is 89.31%.Although this is the best for self-training, it is worse than 93.15% of tri-training.

The reason that self-training does not work well in general is the following: When the training set is small, the available data may not reflect the true distribution of the whole data. Then classifiers will be biased and their classifications will be biased too. In testing, the biased classifiers will not have good accuracy. However, in tri-training, and co-training, each individual view may be biased but the views are independent. Then each view is more likely to produce random samples for the other views and thus reduce the bias of each view as the iterations progress.

**(5) Comparison with co-training baselines**

We now compare tri-training with co-training (Blum and Mitchell, 1998) in Table 4. Again, tri-training beats co-training consistently. The best performance of co-training is 92.81% achieved on the character and lexical views after 60 iterations. However, the accuracy is worse than that of tri-training. The key reason is that tri-training considers three views, while co-training uses only two. Also, the predictions by two classifiers are more reliable than those by one classifier.

| k | Tri Train | Co-Train | | |
|---|---|---|---|---|
| | | Char+Lex | Char+Syn | Lex+Syn |
| **0** | 46.85 | 45.75 | 42.02 | 45.75 |
| **10** | 78.82 | 78.84 | 75.89 | 78.85 |
| **20** | 86.19 | 86.02 | 82.59 | 85.63 |
| **30** | 89.69 | 89.32 | 85.77 | 88.98 |
| **40** | 91.52 | 91.14 | 87.52 | 91.16 |
| **50** | 92.58 | 92.19 | 88.46 | 92.02 |
| **60** | 93.15 | 92.81 | 89.21 | 92.50 |

Table 4. Co-training vs. tri-training

In (Qian, et al., 2014), we systematically investigated the effects of learning methods and views using a special co-training approach with two views. Learning was applied on two views but the data augmentation method was like that in self-training. The best result there was 91.23%, worse than 92.81% of co-training here in Table 4, which is worse than 93.15% of Tri-Training.

Overall, Tri-training performs the best and co-training is better than self-training and co-self-training. This indicates that learning on different views can better exploit the redundancy in texts to achieve superior classification results.

## 5 Conclusion

In this paper, we investigated the problem of authorship attribution with very few labeled examples. A novel three-view tri-training method was proposed to utilize natural views of human languages, i.e., the character, lexical and syntactic views, for classification. We evaluated the proposed method and compared it with state-of-the-art baselines. Results showed that the proposed method outperformed all baseline methods.

Our future work will extend the work by including more views such as the stylistic and vocabulary richness views. Additional experiments will also be conducted to determine the general behavior of the tri-training approach.

## Acknowledgements

This work was supported in part by the NSFC projects (61272275, 61232002, 61379044), and the 111 project (B07037).

## References

S. Argamon, C. Whitelaw, P. Chase, S. R. Hota, N. Garg, and S. Levitan. 2007. Stylistic text classification using functional lexical features. *JASIST* 58, 802–822

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In: *COLT*. pp. 92–100

J. Burrows. 1992. Not unless you ask nicely: The interpretative nexus between analysis and information. *Literary and Linguistic Computing* 7:91-109.

J. Burrows. 2007. All the way through: Testing for authorship in different frequency data. *LLC* 22, 27–47

R-E. Fan, K-W. Chang, C-J. Hsieh, X-R. Wang, and C-J. Lin. 2008. Liblinear: A library for large linear classification. *JMLR* 9, 1871–1874

J. Diederich, J. Kindermann, E. Leopold, G. Paass, G. F. Informationstechnik, and D-S. Augustin. 2000. Authorship attribution with support vector machines. *Applied Intelligence* 19:109-123.

M. Gamon. 2004. Linguistic correlates of style: authorship classification with deep linguistic analysis features. *In COLING.*

N. Graham, G. Hirst, and B. Marthi. 2005. Segmenting documents by stylistic character. *Natural Language Engineering*, 11:397-415.

J. Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *LLC* 22:251-270.

H. van Halteren, F. Tweedie, and H. Baayen. 1996. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing* 11:121-132.

H. van Halteren. 2007. Author verification by linguistic profiling: An exploration of the parameter space. *TSLP* 4, 1–17

G. Hirst, and O. Feiguina. 2007. Bigrams of syntactic labels for authorship discrimination of short texts. *LLC* 22, 405–417

N. Jindal and B. Liu. 2008. Opinion spam and analysis. *In: WSDM*. pp. 29–230

T. Joachims. 2007. www.cs.cornell.edu/people /tj/svmlight/old/svmmulticlassv2.12.html

S. Kim, H. Kim, T. Weninger, J. Han, and H. D. Kim. 2011. Authorship classification: a discriminative syntactic tree mining approach. *In: SIGIR*. pp. 455–464

D. Klein and C. D. Manning. 2003 Accurate unlexicalized parsing. In: *ACL*. pp. 423–430

I. Kourtis and E. Stamatatos, 2011. Author identification using semi-supervised learning. *In: Notebook for PAN at CLEF 2011*

J. Li, R. Zheng, and H. Chen. 2006. From fingerprint to writeprint. *Communications of the ACM* 49:76-82.

K. Luyckx and W. Daelemans, 2008. Authorship attribution and verification with many authors and limited data. *In: COLING*. pp. 513–520

D. Madigan, A. Genkin, D. Lewis, A. Argamon, D. Fradkin, and L. Ye, 2005. Author Identification on the Large Scale. *In CSNA.*

K. Nigam and R. Ghani. 2000. Analyzing the effectiveness and applicability of co-training. *In Proc. of CIKM*, pp.86–93

T. Qian, B. Liu. 2013 Identifying Multiple Userids of the Same Author. *EMNLP*, pp. 1124-1135

T. Qian, B. Liu, M. Zhong, G. He. 2014. Co-Training on Authorship Attribution with Very Few Labeled Examples: Methods. vs. Views. In *SIGIR*, to appear.

R. Schwartz, O. Tsur, A. Rappoport, M. Koppel. 2013. Authorship Attribution of Micro-Messages. *EMNLP*. pp. 1880-1891

Y. Seroussi, F. Bohnert and Zukerman,.2012. Authorship attribution with author-aware topic models. *In: ACL*. pp. 264–269

Y. Seroussi, I. Zukerman, and F. Bohnert. 2010. Collaborative inference of sentiments from texts. *In: UMAP*. pp. 195–206

E. Stamatatos. 2007. Author identification using imbalanced and limited training texts. *In: TIR*. pp. 237–241

E. Stamatatos. 2009. A survey of modern authorship attribution methods. *JASIST* 60:538–556

Ö. Uzuner and B. Katz. 2005. A comparative study of language models for book and author recognition. *Proc. of the 2nd IJCNLP*, 969-980.

Y. Zhao and J. Zobel. 2005. Effective and scalable authorship attribution using function words. *In Proc. of Information Retrival Technology*, 174-189.

R. Zheng, J. Li, H. Chen, and Z. Huang. 2006. A framework for authorship identification of online messages: Writing style features and classification techniques. *JASIST* 57:378-393.