Intrinsic Author Identification Using Modified Weighted KNN

Notebook for PAN at CLEF 2013

M.R. Ghaeini

Department of Computer Engineering and Information Technology,
Amirkabir University of Technology, Tehran, Iran
mr.ghaeini@aut.ac.ir

Abstract In this paper we describe our approach and task for the PAN 2013 Author Identification. Best thing for an intelligent application is a large corpus, but when it is small, it would be helpful to extract many features from dataset and study on them. So, we extract many features from documents like: word bag, stop word bag, punctuation bag, part of speech (POS) bag and etc. It is so important that you select proper features for your learning algorithm; therefore, we decide to learn proper features too. Next, we identify author using a modified weighted KNN since it can have a good performance for a small corpus.

Keywords: Author Attribution, Identification, Intrinsic, Weighted KNN, N_gram

1 Introduction

Authorship attribution supported by statistical or computational methods has a long history starting from 19th century and marked by the seminal study of Mosteller and Wallace (1964) on the authorship of the disputed Federalist Papers. During the last decade, this scientific field has been developed substantially taking advantage of research advances in areas such as machine learning, information retrieval, and natural language processing [1]. Authorship attribution is an important issue and it can be helpful in applied areas such as history, law and etc. Where knowing documents author is important. Human readers can realize that a document belongs to a specific author or not by comparison between document styles. It is the main idea of intrinsic author identification. In Intrinsic algorithms we try to learn author's writing style and it can be hard, so we can use so many different features to recognize author's style; for example we can use stylometric features like lexical features; syntactic features and semantic features. After features extraction, we must calculate document styles similarity (or distance). There are different methods for this calculation like cosine similarity, Minkowski distance, etc. Intrinsic algorithms can be so helpful when we don't have a large corpus. Corpus is so important in learning of intelligent applications.
In this task (Author Identification) we have a small dataset for learning in three languages and it makes that a little hard to reach a proper result. KNN is a good method in this situation since we have a small dataset, we can make decision locally and use whole of our information as good as possible. But we must use weighted KNN to balance the

effect of each feature; otherwise, its result can be unacceptable and awful.
Also, it is so important to avoid wrong announcement, so we must use some methods to decrease wrong announcement chance, therefore we can use some decision maker to reach to a final decision.
Some related works have been done in this area before, and we will describe some of them briefly in blow:
Usually, there are extremely few training texts at least for some of the candidate authors or there is a significant variation in the text-length among the available training texts of the candidate authors. Moreover, in this task usually there is no similarity between the distribution of training and test texts over the classes, that is, a basic assumption of inductive learning does not apply. It presents methods to handle imbalanced multi-class textual datasets [2].
Text representation is a necessary procedure for text categorization tasks. Currently, bag of words (BOW) is the most widely used text representation method but it suffers from two drawbacks. First, the quantity of words is huge; second, it is not feasible to calculate the relationship between words. Semantic analysis techniques help BOW overcome these two drawbacks by interpreting words and documents in a space of concepts. They propose a concise semantic analysis technique for text categorization tasks [3].

2 Feature Selection and Extraction

It is so important that you have proper and enough features for learning something and making decision about it. In text analyzing, we have different parts like: paragraph, sentence and word. We can extract some features from them; also we can consider syntactic or grammatical properties as a feature. Below we describe some features, which we use them in our application.

2.1 Lexical Features

Paragraphs. Authors have their habits about writing, some authors write many paragraphs, but their paragraphs are short. Others don't write many paragraphs, but their paragraphs are long, or other styles. We can consider paragraph number and average of paragraph length as a feature.

Sentences. Sentence is like paragraph in our observation. We can consider number and average length of them as a feature too. It really can be helpful, since number and length of sentences related to writer's mind.

Words. Words are so useful in our approach. But they are too much and if we use them without any preprocessing, our features can be specific and it isn't good at all. So we use Stemming to reach more general words in our approach; since there are different words which all have the same stem and concept.
Besides, stop words aren't important and they are so common between authors. So we ignore them in word processing to make words more meaningful.

In word processing, we consider words average length, unique word number average in a sentence and word number average in a sentence as a feature (in this section, we process on document words without stemming or removing stop words.
Moreover, we make up bag of words in two different ways. In First way, value of a word equals to its frequency in documents, and in second way, value of a word equals to its TF.IDF amount.

2.2 Syntactic Features

One reason that syntactic features perform well is because they are topic-independent. A person's preferred syntactic constructions can also be cues to his/her authorship [4].

Parts-of-Speech scoring by n-gram. A simple and proper way to capture syntactic construction of an author is parts-of-speech tagging. In this case, we have construction of a document. Also it is important that we generalize our POS types, for example we can classify every type of noun like plural, as noun. Now, we use n-gram to capture author style. We use trigrams, 5-grams and 7-grams to reach a proper result.
Moreover, we make up a bag for each of them (like bag of words) by using frequency of them in a document, but in this case we use specific type of them, since we look at the whole document (not sequence like n-gram) and we don't need to generalize them anymore and their specific types can be helpful, for example an author maybe use passive verbs too much, and if we generalize that, we lose an important knowledge about him/her.

Punctuation habits. For unedited texts, we can identify an author based on frequency of distinctive punctuation habits [5]. We inspect some popular punctuation and obtain frequency of them. It can help us in author identification.

Stop words. Stop words describe relationships between content words, they are meaningless and we can remove them in word processing, but their frequencies can help us in author identification, for example some authors use "the" too much. Authors can use different stop words in their document, for example we can use "thus" instead of "so" in many situations. Furthermore, there are different stop words with the same meaning but noticeable different length, like: "furthermore" and "also" or "consequently" and "so".
We consider stop words average length and stop words average number in a sentence as a feature. Moreover, we made up a bag of stop words (like bag of words) by usage frequency of them in a document.

2.3 Other Features

language. In this task, we must study on three languages (English, Spanish and Greek), so language can be a feature in our task, since every language can has its properties. We use this feature in decision making, since maybe it isn't good to use train result of a language to identify author with another language without any observation.

known documents number. In this competition we have some sub folder and we must recognize that unknown document belongs to known document(s) author or not. But we have varying number of known documents in them. Number of known documents can be a feature because if it was low, we would need more probability to making decision about unknown document; therefore, we use this feature in making decision too.

3 Algorithm

We can divide our approach into 3 main part: PreProcssing, Comparison Step and Decision Making.

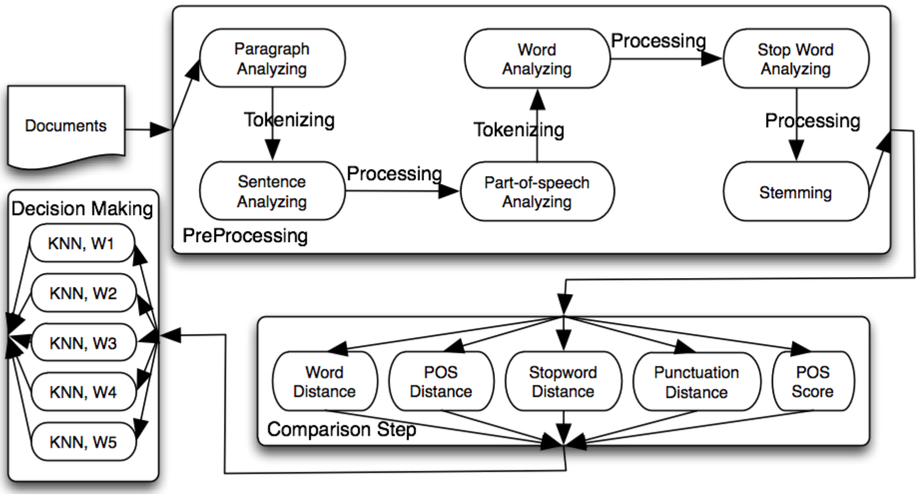

Figure 1. Abstract diagram of our approach

3.1 PreProcessing

We extract documents features in preprocessing step. We described this step in section 2 of this paper. This step done for all known and unknown author documents in a folder (every document which related to a problem). At the end of this step, we prepare all features that we need, like: bag of word, stop word, parts-of-speech and etc.

3.2 Comparison Step

In this step we have two types of work. First calculating distance of our bags (like bag of word, stop word, etc.) from unknown document bags; second, calculating parts-of-speech score for unknown document.

Distance Calculation. We have at least 4 bags (you see them in Figure1) for each document, and if we want to use them in learning, we must calculate their distances from corresponding unknown document bags. For doing that, we use cosine similarity to determine distance or similarity between two arrays or bags (Equation 1).

$$similarity = cos(\theta) = \frac{\overrightarrow{A}.\overrightarrow{B}}{\parallel \overrightarrow{A} \parallel . \parallel \overrightarrow{B} \parallel} \tag{1}$$

Cosine similarity is a popular vector based on similarity measure in text mining and information retrieval. Cosine similarity is a good method for determining distance of two vectors, because length of vectors don't affect the similarity of them, thus we can simply ignore difference in document length by this feature.
In many problems we have more than one known document, thus we must determine a way to reach one amount for distance between unknown document and known documents. There are many ways (like multiplying) to reach that, but we use average amount of them. For more explanation, after calculating POS distance for each known document from unknown document, we set average of them as POS distance of known documents from unknown document.

Parts-of-speech Score. When we are processing known documents, we capture frequency of each POS n-gram (trigram, 5-gram and 7-gram separately), which exists in them (and at last we normalize their frequencies). Next, in processing unknown document, we search each POS n-gram, which exists in that in our POS n-gram list. If we find that in our list, we add frequency of that to POS score of unknown document, and if we can't find that, we subtract 1 point from unknown document POS score.

3.3 Weighted KNN

We use KNN algorithm for our application. KNN is a proper method in this task because we have a small corpus. We can make decision locally and use all of our information as good as possible by KNN. But we must use weighted KNN to balance the effect of each feature; otherwise, its result can be unacceptable and awful. Thus, we use some coefficients when we want to calculate distance between test data and train data (Equation 2).

$$distance_j = \sqrt{\sum_{k=0}^{Dimensions} ((testData_k - trainData_{jk})^2 * w_j)} \tag{2}$$

At last, if test data, and train data j belong to different languages, we add a constant to distance of them.
But it isn't enough yet, determining weights is so significant; therefore, we learn them too. We use gradient descent approach to find best weights, which we can. In this case we just need to calculate cost (Equation 3) partial derivation based on weight of each dimension (Equation 4). It is noticeable that y^* means real and true result of train data (label of data) in equations.

$$Cost = \sum_{i=0}^{testsetsize} (y_i - y_i^*)^2 \tag{3}$$

$$\frac{\partial Cost}{\partial w_j} = \sum_{i=0}^{testsetsize} (2 * (y_i - y_i^*) * \frac{\partial y_i}{\partial w_j}) \tag{4}$$

Calculation of partial derivation of cost function is a little long; therefore, we can't explain it here.
Moreover, we set K equals to 3, in the other hand we use 3NN, because we want to search more locally in our dataset. But we change decision-making method in our KNN (or 3NN) too. Distance affects on our decision too and we don't only look at the label of neighbors (Equation 5).

$$y_i = \frac{\sum_{j=0}^{K} (\frac{y_j^*}{distance_{ij}})}{\sum_{j=0}^{K} (\frac{1}{distance_{ij}})} \tag{5}$$

In Equation 5, we assumed that arrays has been sorted ascending based on $distance_{ij}$ (distance between test data i from train data j). So, $distance_{i0}$ is minimum distance of test data i from our dataset, and y_0^* is its label.
In this case, y_i equals to probability that unknown document belongs to known document(s) author.

3.4 Decision Making

It is so important to avoid wrong announcement in our application. We learned feature weights by our learning method and in best case, we reached to %85.71 hit (30 problems of 35 problems identified correctly) with different weights and different fails (we couldn't identify different 5 problems correctly). Accordingly, we use 5 decision makers with different weights to reach better and more robust result. We select 5 decision makers with different weights that can cover their fails as good as possible. We try to cover each miss with at least 3 hit to make sure that our algorithm can identify better in that decision area.
Weight series were so different. We saw that some features got 0 weight in some weight series, thus we tried to select different weight series with different attention to features to reach best result that we could.
At last, we consider average of decision maker probability as probability of unknown document. Next we make final decision based on known document number. For each range of known document number, we determine a threshold to announce our result, since known document number affects to certitude of our result (if we have more known documents, we can capture author style and features better and with more probability). If the probability doesn't reach to test data threshold, we don't announce any thing.

3.5 Resources

We use three APIs for POS tagging: "Stanford-postagger" for English tagging, "Tree Tagger" for Spanish tagging and "AUEB_POS_tagger" for Greek tagging.

Table 1. Preliminary performances on the test data

Language	F_1	*Precision*	*Recall*
All	0.606	0.671	0.553
English	0.691	0.760	0.633
Greek	0.461	0.545	0.400
Spanish	0.667	0.696	0.640

Runtime 125655^{ms}

4 Evaluation

Unfortunately, we didn't get excellent results, because of two mistakes in our implementation. First, we set thresholds without any evidence, so it affected recall and precision, since we use a useless high certitude. Second, we didn't select proper weight series for decision-making step, since we forgot effect of probability in decision-making step.
About runtime; we print many strings in our algorithm, to it be traceable and print command is one of the slowest commands in programming; consequently, our runtime is more than the actual runtime of our algorithm.

5 Discussion

In our task, we must study on three languages: English, Greek and Spanish, but unfortunately, we were not so familiar with Greek and Spanish at all. Especially Greek, and we got poor result in this language. We just treated with them like English and we only used their POS tagger, their stop words and their stemmer (nothing else is different between them in our application).
More important, we set thresholds without any evidence and it really affected result of our application. We must learned proper thresholds too, or at least we must use some different tests set to reach proper thresholds.
Another mistake is that we tried to reach good weights for features without observing author language. Importance of our features (weights) can be change in different languages, thus we must learn weights for each languages separately.
Moreover, when we selected our making decision weights series, we didn't pay attention to probability of their responses, but we use average of their responses probability to make final decision (if one of them returns a wrong probability, which is far from the right answer, it badly affects our final decision and also can make us give a wrong answer). And when we ran our application again for testing this hypothesis, we saw that it is true (unfortunately).
Furthermore, our data set was too short and we didn't use any train data more than competition train dataset, but we could do that and if we did we could absolutely get a better result.

6 Conclusion & Future Works

According to Discussion section, we can say that the presented algorithm in this paper is a good and proper algorithm for author identification based on competition task, since when we don't have a large corpus, each train data can be important. Also, we can have some local decision areas; therefore, we can't use linear learning algorithms, thus our algorithm can be so good and proper in this situation (Moreover, if we can use linear learning algorithm, we can use this algorithm too without loosing anything). Some of KNN algorithm problems are:
1- KNN can take a long time to make decision and find K nearest neighbors. But in this task, we didn't have large dataset, so it takes a short time to find them; moreover, we can use some tree structures like BST to reduce this time.
2- If we have (a) non-related and helpful feature(s) among our features, KNN result can be bad and make unacceptable result. But we learn weights and importance of features, thus we don't have any problem about non-related features. Moreover, when we use weights for effect of features, we reach a good and meaningful distance for test data and it helps us to make a good decision very much.
3- If we only pay attention to K nearest neighbor without considering their distribution, it is possible that our distance don't show real distance of test data from their neighbors. But we pay attention to it too.
As you see, we don't have these problems in our algorithm, so it can be so good for this task.
But unfortunately, we have some mistakes in our implementation; therefore, we couldn't get an expected and proper result (top three) in competition.
Next time, we will absolutely use more train data and never use anything without acceptable evidence. Our ideas were good but we didn't use them in a correct way, we think that if we used them correctly, we could reach a proper result for sure.
Moreover, we will study on language properties and we will use different weights series for each language to get a better result. Besides we will find good thresholds for our algorithm by learning.
Furthermore, we can use some decision maker, but when we want to make decision, we use some weights for combining their result to reach proper result and paying attention to their responses probability.

References

1. E. Stamatatos. "A Survey of Modern Authorship Attribution Methods". American Society for Information Science and Technology. Vo. 60, No. 3, pages 538-556, March 2009.
2. E. Stamatatos. "Author identification: Using text sampling to handle the class imbalance problem". Information Processing and Management. Vo. 44, No. 2, pages 790-799, March 2008.
3. L. Zhixing, X. Zhongyang, Z. Yufang, L. Chunyong, L. Kuan. "Fast text categorization using concise semantic analysis". Pattern Recognition Letters. Vo. 32, No. 3, pages 441-448, February 2011.
4. P. Juola. "Authorship Attribution. Foundations and Trends in Information Retrieval". Vo. 1, No. 3, pages 233-334, December 2006.
5. C.E. Chaski. "Empirical evaluations of language-based author identification". Forensic Linguistics. Vol. 8, No. 1, pages 1-65, 2001.