Distance learning for Author Verification
Notebook for PAN at CLEF 2013

Paola Ledesma[1], Gibran Fuentes[2], Gabriela Jasso[2], Angel Toledo[2], and Ivan Meza[2]

[1]Escuela Nacional de Antropología e Historia (ENAH)
http://www.enah.edu.mx
[2]Instituto de Investigaciones en Matemáticas aplicadas y en Sistemas (IIMAS)
Universidad Nacional Autónoma de México (UNAM)
http://www.iimas.unam.mx
paola@turing.iimas.unam.mx, gibranfp@turing.iimas.unam.mx,
gabrielajassolopez@gmail.mx,
angeltc@ciencias.unam.mx, ivanvladmir@turing.iimas.unam.mx

Abstract This paper presents a distance metric learning method for the 'PAN 2013 Author Identification' challenge. Our approach extracts multiple distance metrics of different document representations from which our system learns to tune each one of these distances and representations to form an combined distance metric. We reach this learning distances by means of linear programming, Support Vector Regression and Neuronal Networks models. As specified by the description of the task our system can be applied to English, Greek and Spanish, and can be configured to be language independent. We achieved moderately successful results on this PAN competition, with better results on the development test set. We present results with the official test set and our own corpus based on web documents.

1 Introduction

Authorship attribution [4,5,8] is an important task for several areas including information retrieval and computational linguistics and has an impact in fields such as law and journalism. A common scenario for this task is the author verification problem. In this setting we have access to a set of documents by a single author and a questioned document, the problem is to determine if the questioned document was written by that particular author or not.

This year in the *PAN 2013 Author Identification* task was defined as follow[1]:

> *Given a small set (no more than* 10*, possibly as few as one) of "known" documents by a single person and a "questioned" document, the task is to determine whether the questioned document was written by the same person who wrote the known document set.*

There are multiple features which could characterise an author writing: lexical, syntactic and stylistic. In our approach, we focus into capture these features in different

[1] As described in the official website of the competition http://pan.webis.de/

vector space representations of the *known* documents, for instance *bag of words*, n-*grams*. A reasonable hypothesis is that two documents by the same author should be “closer” while documents by different authors should be “distant”. We aim to capture this intuition by calculating a metric distance between the vector space representations of the documents. Documents by the same author would share similar characteristics and therefore they would have a distance close to zero while documents by different authors would have a distance close to one.

We frame the author verification problem as a problem of learning a distance[9] from multiple distances over multiple vector space representations. In order to achieve this we tried a *linear programming* [6] (LP), a *Support Vector Regression* [2] (SVR) and *Neural Network* [3] (NN) machine learning frameworks. The intuition is to identify the contribution of each one of the distances to form a new distance by means of a linear combination of distances, support vectors or weights. This setting allows us to verify the author in two documents. In order to infer the author for a set of documents, as stated in the task, we calculate the distance between the *questioned* document and each of the *known* documents. We used a voting scheme in which we count how many times the system decided a *yes* or *no* for the *questioned* document.

The outline of this works is the following. Section 2 reviews the vector space representation of documents and lists the representations used in our approach 2. Section 3 presents the metrics used to calculate the distance between documents. Section 4 introduces the different machine learning methods we experimented with. Section 6.2 lists some specification of our final system. Section 5 shows the characteristics of the corpora used during development. Section 6 shows our development and final results using this approach and our system. Finally, 7 discuss about future work and presents some conclusions.

2 Vector Space Representation

A vector space model[7] represents a document as a vector space:

$$d = (w_1, w_2, ..., w_m)$$

In which w_i is a frequency or weight of a *word* of the vocabulary of size m for the word i. A common representation of a vector space model is the bag of words model, in which the words represent actual words of the document and the frequencies counts of occurrence of such words in the represented document.

In our approach we use the following documents representations:

Bag of words Frequencies of words in the document.
Bigram Frequencies of two consecutive words.
Trigram Frequencies of three consecutive words.
Comma Frequencies of commas signs.
Dots Frequencies of period signs.
Numbers Frequencies of whole numbers.
Capitals Frequencies of capitalized words.
Words per paragraph Frequencies of words per paragraphs
Sentences per paragraph Frequencies of sentences per paragraphs.
Square brackets Frequencies of square brackets.

3 Distance metrics

A distance function calculates the dis-similarity between two elements. We can calculate distances between the vector space representations of our documents presented in the previous section. Formally, a distance function D and vector space d form metric spaces when the following holds:

$$\begin{aligned}
&D(d_i, d_j) \geq 0 \\
&D(d_i, d_i) = 0 \\
&D(d_i, d_j) = d(d_i, d_j), i \neq j \\
&D(d_i, d_k) \leq D(d_i, d_j) + D(d_j, d_k), i \neq j, j \neq k
\end{aligned}$$

In particular we focus on bounded distances, these are distances that also hold:

$$D(d_i, d_j) \leq << 1.0, i \neq j$$

This means that the distances are bounded by one: they range from zero, when they are the same element, to one, when they are completely dis-similar elements.

There are multiple distance functions to used with the vector space representation. We divided the metrics into three categories: binary, weighted and euclidian based distances. Next we describe each one of these categories.

3.1 Binary distances

This type of distances are based on point wise logical operations on the vector spaces. Example of these distances are Jaccard, Massi and Sorensen. However in our approach we only used:

Jaccard This metric measures the common type vocabulary between two sets:

$$D(d_1, d_2) = \frac{\sum d_1 \wedge d_2}{\sum d_1 \vee d_2} \tag{1}$$

3.2 Weighted distances

This type of distances are based on binary distances but they are modified to take into account the frequencies or weights of the vector space representation. In our approach we use the following weighted distances:

Weighted Jaccard This metric measures the common token vocabulary between two sets:

$$D(d_1, d_2) = \frac{\sum min(d_1, d_2)}{\sum max(d_1, d_2)} \tag{2}$$

Weighted Sorensen This metric measures the twice the common token vocabulary between two sets:

$$D(d_1, d_2) = \frac{\sum d_1 \oplus d_2}{\sum d_1 + d_2} \tag{3}$$

where $\oplus$ is the sum of the vectors components with values grater than zero.

Weighted Massi This metric measures the common token vocabulary compared with the vector with more mass frequency:

$$D(d_1, d_2) = \frac{\sum min(d_1, d_2)}{max(\sum d_1, \sum d_2)} \tag{4}$$

Weighted h_0 [1] This metric measures the minimum token vocabulary in common versus the maximum vocabulary in common:

$$D(d_1, d_2) = \frac{\sum min(d_1, d_2)}{max(\sum max(d_1, d_2)} \tag{5}$$

3.3 Euclidean distances

This type of distances are based on the Euclidean representation of the vector space. In order to bound them to 1 we normalize them.

Euclidean This metrics measures the Euclidean distance between two vectors :

$$D(d_1, d_2) = \frac{\sqrt{\sum_{i=0}^{m} (d_{2i} - d_{1i})^2}}{\sum_{i=0}^{m} (d_{2i} - d_{1i})^2} \tag{6}$$

Cosine This metric measures cosine angle between to vectors:

$$D(d_1, d_2) = \frac{d_1 \dot{d_2}}{d_1 \dot{d_1} * d_2 \dot{d_2}} \tag{7}$$

3.4 Ledesma Coefficient

Additionally to the standard distances presented in the previous sections, we found the following coefficient quite helpful during the learning stage:

$$c(d_1, d_2) = \frac{d_1 + d_2}{(\sum d_1)^2 * (\sum d_2)^2} \tag{8}$$

We found this coefficient to be sensitive to documents that are very different among themselves, which helps to signal when two documents did not belong to the same author.

4 Linear programming model

Let (V, D) be a metric space bounded by 1 in which V is a vector space and D a distance function, then for all $d_1, d_2, d_3 \in V$, $D(d_n, d_n) = 0$, $D(d_1, d_2) = d(d_2, d_1)$, $D(d_1, d_3) \leq D(d_1, d_2) + D(d_2, d_3)$ and $D(d_1, d_2) \leq 1$ holds. With these properties in mind we propose to have a linear combination λ of distance functions $\mathcal{D}$ such that for $D_1, D_2, ..., D_n \in \mathcal{D}$, $d_l = \lambda_1 D_1 + \lambda_2 D_2 + ... + \lambda_n D_n$ is a bound distance function bounded by 1. In order to find such linear combination we propose the following linear program:

$$\begin{aligned}
\textbf{minimize} \quad & \lambda_1 + \lambda_2 + ... + \lambda_m \\
\textbf{subject to} \quad & \lambda_1 * d_1 \lambda_2 * d_2 ... + \lambda_m * d_m \geq 0.0, \textit{if} \text{ author} \\
& \lambda_1 * d_1 \lambda_2 * d_2 ... + \lambda_m * d_m \leq 0.5, \textit{if} \text{ author} \\
& \lambda_1 * d_1 \lambda_2 * d_2 ... + \lambda_m * d_m > 0.5, \textit{if} \text{ not author} \\
& \lambda_1 * d_1 \lambda_2 * d_2 ... + \lambda_m * d_m \leq 1.0, \textit{if} \text{ not author} \\
\textbf{and} \quad & \lambda > 0
\end{aligned}$$

This linear program aims to find the weights λ such that it can split the distances which makes two document be assigned to the same author or otherwise. During the training stage the inequality constrains are expanded with actual examples of values for distances and subject to the inequality depending if thats an example of documents from the same author or not. Then the linear program is run and the solution represents the weights λ.

5 Corpora

Additionally to the official training corpus provided by PAN organization we collected an extra corpus based on documents from the web. Table 3 summarises the main characteristics of both corpora.

	No. Know. Docs	No. Problems	Vocabulary Type	Vocabulary Token
Official training corpus				
English	32	10	$2,612$	$33,794$
Greek	100	20	$23,773$	$149,373$
Spanish	12	5	$2,913$	$7,920$
Web corpus				
English	75	13	$10,748$	$9,1751$
Greek	7	3	$2,172$	$6,163$
Spanish	37	11	$5,805$	$27,713$

Table 1. Corpora characteristics for known documents

6 Experimental Results

For our development stage we performed a *leave-one-out* cross-validation using the official training corpus, but we tested the model created by our approach using the Web corpus we collected explained in the previous section. Table **??** presents our development and final results for the official training and testing set and the web documents corpus we extracted.

	Training	Web Corpus	Test
English	70.0%	46.2%	46.7%
Greek	95.0%	100.0%	66.7%
Spanish	60.0%	90.9%	72.0%
Total	82.9%	70.4%	61.2%

Table 2. Development results using linear programming

6.1 Linear regression setting

Additionally, to the linear programming approach we tried a linear regression setting with Support Vector Regression and Neural Network. Table **??** shows the reached results, we do not have results for the official test corpus since these configurations were not part of the final system. The distance learning setting was not compatible with the linear regression since it was difficult to constraint the sum of the weights between zero and one. This situation make it hard to reach good results with these settings.

	Training	Web Corpus
Support Vector Regression		
English	70.0%	53.8%
Greek	85.0%	66.7%
Spanish	60.0%	72.7%
Total	77.1%	63.0%
Neural Network		
English	60.0%	46.2%
Greek	40.0%	0.0%
Spanish	60.0%	63.6%
Total	48.6%	48.1%

Table 3. Development results using SVR and NN

6.2 Final System

Additionally to the distance learning setting we incorporating several pre-processing strategies to our finale system[2]. The following explains these strategies and list the used parameters for the final version of the system.

Weighting frequencies For the vector space representation of the documents we tried the following weighting:

– Term frequency tf
– Log term frequency tf_{log}
– Inverse document frequency term frequency $idtf * tf$

From experimentation we settle with log term frequency which is only a change to a logarithm scale for the frequencies of the terms in the vector space.

Stop words Some of the representations were preprocessed by eliminating stop words. For this, we used a standard list of words for each of the languages [3]. The representations which were preprocessed with stop words were: Comma, dots and Capitals.

Cut-off frequencies The vector space representation were also preprocessed by eliminating term with few counts, these are uncommon terms. For each of the representation a *cut-off* value was defined, the following list the values for each of the representations which were different than zero:

Bag of words	5
Numbers	1

7 Conclusions and Future work

In this paper we have presented a learning distance metric method for the 'PAN 2013 Author Identification' challenge. Our approach extracts multiple distance metrics of multiple document representation from which our system learns to tune each one of these distances and representations to form an combined distance metric. Our main approach uses linear programming to find a linear combination of distances, however we also present prelimilary results with a support vector regression and neural networks setting. Our final system achieved moderately successful results on this PAN competition, with better results on the development test set. In particular, English language was harder to verify the author, our system did worst than the baseline system, while for Greek and Spanish we were 4th place in performance. However, the current results point to more work on the side of representations, we are investigating syntactic and semantic feature representations.

[2] `https://github.com/ivanvladimir/authorid`
[3] This can be found here: `https://raw.github.com/ivanvladimir/authorid/4e3bd5b968135380ce839340541892c30d13cf5d/data/stopwords.txt`

We are investigating improvements to our system, in particular the current version verifies an author by a voting scheme from a set of distances from the known document to the unknown document. But our intuition is that even for an author there will be documents which are closer to the unknown and other than no so close, if a cluster are highly closer it should be considered a good evidence of authorship. We are studying mechanism to add this intuition into our inference processing.

References

1. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: Proceedings of BMVC (2008)
2. Drucker, H., Burges, C.J., Kaufman, L., C, C.J., Kaufman, B.L., Smola, A., Vapnik, V.: Support vector regression machines (1996)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edn. (1998)
4. Juola, P.: Authorship attribution. Found. Trends Inf. Retr. 1(3), 233–334 (Dec 2006)
5. Koppel, M., Schler, J., Argamon, S.: Computational methods in authorship attribution. Journal of the American Society for Information Science and Technology 60(1), 9–26 (2009)
6. Mitchell, J.E., Farwell, K., Ramsden, D.: Interior point methods for large-scale linear programming (2004)
7. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (Nov 1975)
8. Stamatatos, E.: A survey of modern authorship attribution methods. Journal of the American Society for Information Science and Technology 60(3), 538–556 (2009)
9. Yang, L.: Distance metric learning: A comprehensive survey (2006)