A Slightly-modified GI-based Author-verifier with Lots of Features (ASGALF)

Notebook for PAN at CLEF 2014

Mahmoud Khonji and Youssef Iraqi

Khalifa University
{mahmoud.khonji, youssef.iraqi}@ku.ac.ae

Abstract This paper presents the performance evaluation of an authorship verification technique that is based on a modified version of General Impostors (GI) [2]. The novelties of this implementation are: 1. a modified way of combining the `min-max` similarity measure and, 2. a relatively large set of diverse features that spans letter-level, word-level, function word-level, word shape-level, and word tag-level features. The technique ranked high in overall in the author identification task of PAN 2014.

1 Introduction

This paper presents the implementation and evaluation of our classifier for the author identification task in PAN at CLEF 2014.

PAN organizers provided us with a collection of problems $\mathcal{P} = \{P_i : \forall\ i \in I\}$ where $I = \{1, 2, \ldots, n\}$ is the index set of $\mathcal{P}$. Each problem $P_i \subset F$, for all $i \in I$, where $F = \{f_j : \forall\ j \in J\}$ is a set of Unicode text files and $J = \{1, 2, \ldots m\}$ is the index set of F. Additionally, each $P_i \in \mathcal{P}$ has the following properties:

- All $f_j \in P_i$ are written in only one of the following language-genre combinations: Dutch essays, Dutch reviews, English essays, English novels, Greek articles and Spanish articles.
- Only one text file $f_u \in P_i$, for some $u \in J$, is written by an author whose identity is *unknown*, and the rest of the files $P_i - \{f_u\}$ are written by authors whose identities are *known*.
- The cardinality $|P_i - \{f_u\}|$ is allowed to vary from 1 to 5 inclusive as i varies, but (obviously) fixed for a given i.

Thus the goal of the authorship identification problem is to find a mapping function $h : \mathcal{P} \rightarrow Y$ that solves the optimization problem (1), where $Y = [0, 1]$ with the semantics of (2).

$$\underset{h \in H}{\textbf{arg max}}\ \texttt{AUC}(\{h(P_i) : \forall\ P_i \in \mathcal{P}\}) \times \texttt{C@1}(\{h(P_i) : \forall\ P_i \in \mathcal{P}\}) \tag{1}$$

, where H is the set of all possible mapping functions, `AUC` is the area under the Receiver Operating Characteristic (ROC) curve, and `C@1` is essentially a modified accuracy measure that rewards models $h \in H$ that know when not to decide as presented in (3).

$$h(P_i) = \begin{cases} y \in [0, 0.5) & \text{if } h \text{ predicts authors of } f_u \text{ and } P_i - \{f_u\} \text{ are different} \\ y = 0.5 & \text{if } h \text{ does not predict} \\ y \in (0.5, 1] & \text{if } h \text{ predicts authors of } f_u \text{ and } P_i - \{f_u\} \text{ are same} \end{cases} \quad (2)$$

$$\texttt{C@1} = \frac{N_c \frac{N_c N_u}{|\mathcal{P}|}}{|\mathcal{P}|} \quad (3)$$

, where $|\mathcal{P}|$ is the total number of problems, N_c is the total number of problems that the model h has solved correctly, and N_u is the total number of problems that the model h has decided to not solve.

Additionally, and in order to enhance the quality of the found models $h \in H$, the organizers of PAN have also provided us with another problems collection $\mathcal{L}$ that has an identical structure to the collection $\mathcal{P}$. $\mathcal{L}$ is intended to be used as a training set to find better $h \in H$ models and thus PAN has also provided its ground truth information as a form of function $G : \mathcal{L} \rightarrow \{\text{same authors}, \text{different authors}\}$.

2 Method description

Our classifier, namely A Slightly-modified GI-based Author-verifier with Lots of Features (ASGALF), is (as the name implies) based on a modified version of the General Impostors (GI) framework [2]. It differs compared to other GI implementations in a couple of ways. First, we discuss why we have chosen to use GI as the starting point for ASGALF, then we discuss the differences between the GI implementation [3] and ASGALF.

2.1 Reasons for adopting the GI framework

Based on our preliminary tests, previous PAN results, and our intuitive reasoning, it is very clear to us that the set of impostors (or distractors) does contain some information that helps in answering the question of how similar two vectors are, and considering such information in the decision making process is indeed helpful to solve (1) in a better way than otherwise.

For example, consider two feature vectors $\mathbf{v}_1 = k(f_1)$ and $\mathbf{v}_2 = k(f_2)$, where $k : P_i \rightarrow \mathbb{R}^d$ is a function that extracts the features from any input text file $f_j \in P_i$, for any $j \in J$, and returns a features vector $\mathbf{v}_j \in \mathbb{R}^d$ that corresponds to f_j.

Simply knowing the value of $q(\mathbf{v}_1, \mathbf{v}_2)$ is not enough in reality as far as we know, where $q : \mathbb{R}^d \times \mathbb{R}^d \rightarrow Y$ is a function that returns the similarity of its input vectors.

Our justification to this is that the output of all q functions that are known to-date are somewhat relevant to the topic/context of the evaluated vectors. For example, in some topic/context authors in general tend to be similar, while in some other topic/context authors tend to be dissimilar.

E.g. in some context $q(\mathbf{v}_1, \mathbf{v}_2) = y$ indicates that $\mathbf{v}_1$ and $\mathbf{v}_2$ represent documents authored by the same individual only if y is greater than (say) 0.9. This can be the case if

the topic/context is a restrictive one that causes authors to be very similar to each other (e.g. reports). On the other hand, some other topic/context may allow high variability among authors, thus vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ can be assumed to represent documents that are authored by the same individual even if $q < 0.9$, such as $q = 0.4$ can possibly indicate that the authors are the same.

Thus fetching a set of impostor text files $M = \{m_w : \forall\ w \in W\}$, where $W = \{1, 2, \ldots, n\}$ is the index set of M, is — in our view — a solid step towards a better optimization of the problem (1).

We believe that measuring the distance against impostor vectors $\{\mathbf{x}_w = k(m_w) : \forall w \in W\}$, allows the model to see how close $\mathbf{v}_1$ and $\mathbf{v}_2$ are to each other *relative* to other impostor vectors in the same topic.

Additionally, the GI framework is one that is based upon ensembling randomized models. Although it might not be very obvious at the first glance, we believe that GI essentially creates a set of randomized models in every run (by choosing different features and impostors subsets in every run). Relying on ensembles of models is another strength of the GI framework that is appreciated by the Machine Learning community as well as other competitions such as the Netflix Prize[1] where most of the top techniques were composed of some form of ensembles.

2.2 Differences between ASGALF and [3]

– Instead of using the original Impostors $score$ measure (4), we adopted a modified one as presented in (5). The advantages of (5) is that it allows us to measure *how* similar input vectors are as opposed to *whether* they are similar enough. On the other hand, a possible disadvantage could be over fitting the collection of training problems $\mathcal{L}$.

$$score \leftarrow score + 1 \quad \textbf{if} \quad \begin{array}{l} \texttt{min-max}(\mathbf{v}_1, \mathbf{v}_2)^2 > \\ \texttt{min-max}(\mathbf{v}_1, \mathbf{x}_1) \times \texttt{min-max}(\mathbf{v}_2, \mathbf{x}_2) \end{array} \tag{4}$$

, where $\mathbf{x}_1$ is the most similar impostor to $\mathbf{v}_1$ and $\mathbf{x}_2$ is the most similar impostor to $\mathbf{v}_2$.

$$score \leftarrow score + \frac{\texttt{min-max}(\mathbf{v}_1, \mathbf{v}_2)^2}{\texttt{min-max}(\mathbf{v}_1, \mathbf{x}_1) \times \texttt{min-max}(\mathbf{v}_2, \mathbf{x}_2)} \tag{5}$$

– Using a large set of diverse features. Essentially, we have extracted letter-level, word-level, function word-level, word shape-level, part-of-speech tag-level features as follows:
  - n-grams with various combinations of n values and gram types — n values are $\{1, 2, .., 10\}$, gram types are {letters, words, function words, word shapes[2],

[1] http://en.wikipedia.org/wiki/Netflix_Prize

[2] The word shapes are based on three properties: characters case (e.g. lower/upper case), characters type (whether it is a letter or a number), and words length. For example, the word "School" is represented as the gram "Cccccc", "2014" is represented as the gram "NNNN", "x86" is represented as the gram "cNN", etc.

POS tags[3], POS-words[4]} and the resultant features were based on the combinatorics of all n values and gram types. This resulted in a large number of features, most of which were too infrequent to be reliable, thus we have only considered features that occurred for at least 5 times in any single document. However, the number of features remained large in general even after removing the infrequent features. The total number of extracted features after removing the infrequent ones varied depending on language-genre combinations as shown in Table 1.

Table 1. Total number of extracted features on per language-genre basis.

Language	Genre	Features extracted
Dutch	Essays	19,883
Dutch	Reviews	3,312
English	Essays	70,738
English	Novels	43,931
Greek	Articles	72,944
Spanish	Articles	39,099

- Body richness — the total number of unique words in a given text file f_j, for any $j \in J$, normalized by total number of words in the same file f_j.

Other details of ASGALF that are not mentioned in this notebook are the same as suggested in [3]. For example, similar to [3] we fetch the set of impostors of a given document from a search engine by submitting search queries of 5 randomly chosen words from the subject document, download the 10-highest ranked HTML pages, strip them from any HTML markup, and use their first 1500 words. In ASGALF, we fetched the first 1500 words of 10 impostors for each document in the training set, and then grouped them on per language-genre basis. The output of this process is a set of impostors for each language-genre combination.

3 Parameter tuning

The parameters were tuned based on our preliminary tests against problems in $\mathcal{L}$ as follows:

– Score correction offsets are: -0.4585, -0.62950, -0.43850, -0.24850, -0.478, and -0.56600 for English essays, English novels, Dutch essays, Dutch reviews, Spanish

[3] Part of speech tags such as NN, NNS, NNP, VB, VBD, VBG, etc. For example, if the word "school" existed in a text as a noun, then it would be represented as the gram "NN". A comprehensive list of such tags can be found in `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`.

[4] Combinations of words and their respective POS tags. For example, if the word "saw" was a noun then it would be represented as the gram "saw-NN", and if the word was a verb then it would be represented as the gram "saw-VBD".

articles, and Greek articles respectively. This allowed the final score to be centered around 0.5 in order to satisfy the semantics in (2).
- Total number of Impostor rounds: 50, which also matches the optimal value for the Spanish set in [1].
- Total number of impostor documents: 8,614, 1,257, 2,728, 2,073, 5,347 and 3,104 for English essays, English novels, Dutch essays, Dutch reviews, Spanish articles and Greek articles respectively.
- Total number of randomly chosen impostors per round: 20.
- Total number of randomly chosen features per round: 40% of the total features, which also matches the optimal value for the English and Spanish sets in [3].

4 Evaluation results

Evaluation results are presented in Table 2.

Table 2. PAN14 evaluation results. Note that Testing 1 is the early-bird evaluation dataset, and Testing 2 is the one that is used by PAN's rankings

Language	Genre	Eval. set	AUC	C@1	AUC × C@1	Runtime
Dutch	Essays	Training	0.94076	0.875	0.82316	00:51:38
		Testing 1				00:30:22
		Testing 2	0.91254	0.84375	0.76996	00:58:21
Dutch	Reviews	Training	0.7258	0.67	0.48629	00:12:40
		Testing 1				00:06:05
		Testing 2	0.7364	0.65	0.47866	00:12:24
English	Essays	Training	0.5285	0.545	0.28803	09:29:28
		Testing 1				04:46:23
		Testing 2	0.59875	0.5829	0.34901	09:10:01
English	Novels	Training	0.9716	0.87	0.84529	01:02:03
		Testing 1				01:09:46
		Testing 2	0.75	0.61	0.4575	02:06:16
Greek	Articles	Training	0.5596	0.5959	0.33347	03:43:16
		Testing 1				01:47:47
		Testing 2	0.8892	0.81	0.72025	03:41:48
Spanish	Articles	Training	0.8992	0.87	0.7823	04:54:45
		Testing 1				02:41:57
		Testing 2	0.898	0.7777	0.69837	04:50:49

5 Notable limitations

We believe that the score of the model can improve if the following limitations are addressed:

– Our classifier is implemented such that it always attempts to predict an answer. I.e. it never outputs a score $y = 0.5$, thus limiting its ability in taking advantage of the metric `C@1`.
– The parameters were tuned in a preliminary testing phase against the training datasets. The outcome was that all language-genre combinations had similar optimal parameter values. However, a more rigorous optimization process could have revealed better language-genre-specific parameter values.
– At the core of Impostors as described in [3] is the min-max similarity measure, which is also the measure that we have used as an implementation of the q function. It is possible that a more sophisticated model could have found a better use of our diverse set of features.
– No feature subset selection was performed other than removing features that did not occur frequently enough according to a simplistic criteria as described in previous sections. Using more sound feature subset selection methods (e.g. IG, Wraper, etc) can possibly reduce CPU time requirements as well as enhance the accuracy.

6 Conclusions

This paper describes an authorship verification classifier that is based on the General Impostors (GI) framework with the exception of using a modified method of combining the scores, as well as a diverse set of features. The features were of various types, namely: letter-level, word-level, function word-level, word shape-level, and part-of-speech tag-level.

The evaluation on the testing set shown high classification accuracy in general. This also confirms that impostor/distractor-based methodologies are indeed a step forward. Although the selection of the impostors/distractors set is a known limitation, it seems that it is practically not a major issue and that such set of impostors can be obtained with relative ease (e.g. such as by using search engines as in [3]).

On the downside, our technique generally had a slow runtime. However, we believe that this issue can be minimized in a number of ways, such as:

– Our code made excessive use of automatic execution of external commands via the shell. Some of such external commands were themselves very slow in general, such as the software used to extract the part-of-speech tags. If such external dependency is replaced by some faster one (or completely removed), the speed of the feature extraction phase can improve dramatically.
– We believe that techniques that are based on the GI framework can be easily distributed should multiple cores or machines be available. This is due to the fact that all GI rounds are independent from each other, thus allowing the runs to be distributed across multiple cores or even different machines, ultimately leading in a much reduced runtime.

Acknowledgement

Thanks to Shachar Seidman for answering our questions about General Impostors (GI).

References

1. Joula, P., Stamatatos, E.: Overfiew of the Author Identification Task at PAN 2013. In: Conference and Labs of the Evaluation Forum (2013)
2. Koppel, M., Seidman, S.: Automatically Identifying Pseudepigraphic Texts. In: EMNLP. pp. 1449–1454. ACL (2013)
3. Seidman, S.: Authorship Verification Using the Impostors Methods. In: Conference and Labs of the Evaluation Forum (2013)