GLAD: Groningen Lightweight Authorship Detection

Notebook for PAN at CLEF 2015

Manuela Hürlimann, Benno Weck, Esther van den Berg,
Simon Šuster, and Malvina Nissim

Center for Language and Cognition Groningen, University of Groningen
{m.f.hurlimann, b.f.o.weck, e.m.van.den.berg.4}@student.rug.nl,
{s.suster, m.nissim}@rug.nl

Abstract We present a simple and effective approach to authorship verification for Dutch, English, Spanish and Greek, which can be easily ported to yet other languages. We train a binary linear classifier on both the features describing known and unknown documents individually, and the joint features comparing these two types of documents. The list of feature types includes, among others, character n-grams, the lexical overlap, visual text properties and a compression measure. We obtain competitive results that outperform the baseline and position our system among the top PAN competition participants.

1 Introduction

In the authorship verification task as set in the `PAN` competition[1], a system is given a collection of problem sets containing one or more known documents written by author A_k and a new, unknown document written by author A_u, and is then required to determine whether $A_k = A_u$ without access to a closed set of alternatives. In this form, this task is generally interpreted as a one-class classification problem [11], in the sense that the negative class is not homogeneously represented and systems are based on *recognition* of a given class rather than *discrimination* among classes [1]. This is akin to outlier or novelty detection [5,7] and different from standard authorship attribution problems where a system must choose among a set of candidate authors in a more standard, multi-class text categorisation fashion.

Since multi-class classification is a more natural and efficient way of performing classification, researchers have experimented with the introduction of negative instances to each problem set by selecting random external documents [18] and with the addition of positive examples by splitting long known documents into chunks [11]. This way, each problem set is turned into a true binary classification task as both positive and negative instances are represented, the latter mimed by the external documents. Approaches using external documents are referred to as *extrinsic* approaches, while methods that do not introduce any external documents are called *intrinsic* approaches [21]. Building on the *impostor method* used by [18] within the `pan13` competition, which indeed used sets of external documents to produce negative instances, [10] also use an extrinsic approach and achieve optimal results at `pan14`. This result is interesting as most systems

[1] `http://pan.webis.de`

participating at `pan14` are instead *intrinsic* in nature. Additionally, only three out of 13 approaches exploit actual trained models [21], while the rest pursue a "lazy" strategy.

In the light of the above discussion, we decided to cast the problem as a binary classification task where class values are `Y` ($A_k = A_u$) and `N` ($A_k \neq A_u$). To maximise speed and simplicity, we do not introduce any negative examples by means of external documents, thus adhering to an *intrinsic* approach. To fit a binary classification setting we train a model across the whole dataset, which contains both positive ($A_k = A_u$) and negative ($A_k \neq A_u$) problem sets in equal number. In other words, we treat each problem set as a training vector, exploiting both positive and negative instances in learning. Each instance is represented as a feature vector which contains feature values representing the known document, values representing the unknown document and values comparing the known and unknown document, and is associated to the class value of either Y(es) or N(o). Furthermore, all features that we include in our final models are simple and fast to obtain from any text without requiring any complex processing. An average train or test run of our system on one of the `pan` datasets takes only minutes to complete. Finally, to ensure portability, we do not develop any language-specific feature. Rather, we tune the system towards a specific language by means of selecting and combining features in (possibly) different ways.

2 Approach and Data Representation

There are two reasons for casting an authorship verification problem as a binary classification task: (i) because in the `pan15` training data the problem sets are equally distributed for positive and negative evidence and (ii) because of the success of extrinsic methods, which are more similar to multi-class classification (see discussion in Section 1 above). However, theoretically, the problem still resembles one-class classification more than a true binary classification task, especially because the negative class is not inherently homogeneous (our approach is more alike to telling apples from other fruit than telling apples from pears). It also needs to be noted that, in a realistic setting, evidence would not necessarily be balanced, with negative examples outnumbering positive ones (there are many more fruits that aren't apples than apples).

For one-class classification, Support Vector Machines (SVM) have been shown to perform very well [13,23], and this is especially true when the data are highly unbalanced, as demonstrated by [1]. They also show that, in presence of a balanced set up to a ratio of 1:3.5, binary classification outperforms one-class classification. Since we are dealing with balanced datasets, we use a binary class SVM. In a different setting, a one-class SVM could be used, without major variations in the algorithm.

Our system is implemented using Python's scikit-learn [15] machine learning library as well as the Natural Language Toolkit (NLTK) [2]. We used SVM with default parameter settings in all final models, with an implementation based on libsvm. The features we experimented with and the tools used to extract them are described below.

Building on the existing literature and on observations that came from preliminary exploration of the training data (both from `pan14` and from `pan15`), we developed a set of 29 features. Not all of them were used in the final configuration, as ablation experiments showed little or no contribution of some groups of features. Nevertheless, we describe them all in this section, and present results from ablation experiments in the next.

There are two major ways to divide up the features we experimented with. The first one has to do with the kind of information they represent, and we clustered them in seven different groups (see Sections 2.1–2.7). The second one has to do with how the information is encoded regarding the known and unknown documents. Specifically, features can describe the known and unknown documents *separately*, in which case we talk about *individual* features, or they can describe them *together*, in which case we talk about *joint* features. For example, when comparing the average sentence length of the known and the unknown portion of a training instance, one can represent the average sentence lengths as two *individual* values or compute the difference of the averages to get a single *joint* feature.

2.1 N-gram features

Information on correspondence of character sequences in known and unknown documents has been shown to be a successful feature for this task [20,21]. All n-gram features in our model are *joint* features. We included n-grams with n ranging from 1 to 5, and calculated n-gram similarity in two different ways. According to [9], an author profile is defined as the set of the k most frequent n-grams with their normalised frequencies, as collected from training data. In order to deal with sparseness when $n > 2$, the (dis)similarity measure that they use, and that we adopt, takes into account the difference between the relative frequency of a given n-gram averaged over all known-unknown document pairs of one problem instance. We call this feature group `n-gram norm`. We also use a simple n-gram overlap measure called SPI (simplified profile intersection [20, p. 548]). This measure is based on the number of common n-grams in the most frequent k n-grams for each document. We calculate the SPI score separately for each n between 1 and 5.

Profile size k is another parameter of the `n-gram norm` and `SPI` measures. For our system we fixed k at 100, thus taking into account the 100 most frequent n-grams of each document.

2.2 Token features

A common way to measure the similarity of two given texts is to compute the cosine similarity of their vector representation. We hypothesise that texts written by different authors are lexically less similar than texts written by the same author. We measure the similarity in each training instance by averaging over the L2-normalised dot product of the raw term frequency vectors of a single known document and the unknown document. The token feature is considered a *joint* feature.

2.3 Sentence features

We consider the number of tokens per sentence to be a simple yet effective feature for detecting authorship [12,17]. The values for average sentence length are obtained and represented both as *joint* and *individual* features. Sentence boundaries are determined using language-specific models of the NLTK Punkt Tokeniser[2].

2.4 Entropy features

The notion of the entropy of a text was first introduced by Shannon [19]. We hypothesise that authors have distinct entropy profiles due to the varying lexical and morphosyntactic patterns they use. We use the average entropy of each known and unknown documents (as an *individual* feature type), as well as two *joint* measures: (i) the average entropy of each known concatenated with the unknown document, (ii) and the absolute difference between entropies of known and unknown documents.

2.5 Visual features

Upon inspecting the training data, it became apparent that while Dutch documents contained no newline characters, and Greek and Spanish texts were visually very uniform, many of the English training instances had been drawn from plays and poems. These documents are characterised by marked usage of white space and punctuation. For instance, the sample text in Figure 1 includes apostrophes to reflect the speakers' accents, many mid-sentence line breaks and the segmentation of the text into turns of dialogue. Compared to a piece of prose, or to a play, we predict that this text has a higher number of apostrophes and newlines and that this information is crucial for distinguishing same-author from different-author instances.

```
 Aw, goin ' to school didn ' t do me no good. The teachers was all
down on me. I couldn ' t learn nothin ' there.

Nor any other place , I 'm thinkin ', you 're that
thick , Whisht! It 's
the doctor comin ' down from Eileen. What ' ll he say , I wonder? Aw, Doctor , ↙
     ↳ and how ' s Eileen now? Have you got her
cured of the weakness?

 Here are two prescriptions
that ' ll have to be filled immediately.

You take them, Billy , and run round to the drug
store.

 Give me the money, then.
```

Figure 1. Text sample from English training data.

Taking into consideration our aim of designing a lightweight system, we opted for straight-forward measures of layout properties: (i) punctuation, to capture differences

[2] http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.punkt

in use of typographical signs, (ii) line endings, to measure preferred ways of closing lines, (iii) letter case, to capture e.g. the occurrence of names and the capitalisation of the start of sentences, and finally (iv) line length measures and (v) text block size, to capture the amount and distribution of blank space in a document.
In detail, the visual features consist of:

(i) Punctuation:
frequencies of exclamation marks, question marks, semi-colons, colons, commas, full stops, hyphens and quotation marks

(ii) Line endings:
frequencies of full stops, commas, question marks, exclamation marks, spaces, hyphens, and semi-colons at the end of a line

(iii) Letter case
- ratio of uppercase characters to lowercase characters
- proportion of uppercase characters

(iv) Line length:
- sentences per line
- words per line
- proportion of blank lines

(v) Block size:
- number of lines per text block
- number of characters per text block

After obtaining vectors for documents, counts were averaged for groups of known documents, generating a simple author profile for the known documents to be compared to the questioned document. We use the cosine similarity for the property vectors *punctuation*, *line endings* and *line length*, and simple subtraction for *letter case* and *text block*. All visual features are *joint*.

2.6 Compression feature

Compression features have been successfully used for authorship identification, and can yield performance similar to n-gram features [12]. We used the “Compression Dissimilarity Measure (CDM)" [12, p.19], which, for two documents `x` and `y` is defined as the sum of the compressed lengths of `x` and `y` divided by the compressed length of the concatenation of the two documents:

$$CDM(x,y) = \frac{C(x) + C(y)}{C(xy)}.$$

Our implementation of CDM normalises the compressed lengths by the number of characters in each document and uses the zlib[3] algorithm for compression. By definition this is a *joint* feature.

[3] `http://zlib.net/`

2.7 (Morpho)syntactic features

We also investigate the role of more elaborate feature types, namely part-of-speech (POS) tags and syntactic functions obtained from dependency trees. Previous attempts have mostly dealt with shallow information obtained from POS tags (see [20] for an overview), whereas methods using syntactic features are less common, especially those using dependency- rather than phrase-based syntax [6,16]. There are many possible ways to include (morpho)syntactic features, however the underlying motivation for including them is typically that the frequency and the patterns of (morpho)syntactic categories are not under conscious control by authors. We run these experiments for English only.

We use the MST dependency parser [14] for English trained on sections 2–21 of the Penn Treebank WSJ (PTB) prepared using the standard procedures of [22] and [8]. The parser achieves a reasonable labeled accuracy of around 0.85 on the testing part of the PTB. POS tags are obtained with the Citar tagger [4], which achieves an accuracy of around 96% on the PTB test sections. We take a simple approach of comparing POS and dependency label distributions of known and unknown documents, and use two measures for comparing the distributions: cosine similarity and entropy. For the former, we calculate the L2-normalised cosine similarity between two frequency distributions, producing a *joint* feature. For entropy, we calculate the Shannon entropy (see Section 2.4) of the known and unknown texts separately (averaged in case of multiple known documents), yielding two *individual* features. We also use the absolute difference between the two as a *joint* feature. Entropy features are only used for POS tags.

3 Feature selection

In order to assess the contribution of the feature types discussed in Section 2, we ran a series of feature ablation and single-feature experiments, where "feature" actually stands for each of the groups defined in Sections 2.1–2.7. To these, we added four more categories that just group selected feature types:

- all *individual* features
- all *joint* features
- visual and compression features (`vis+comp`)
- visual features, compression features, and n-gram features (`vis+n+tok`)

All tests for evaluating the contribution of the various features were conducted in [4], using the `LibSVM` classifier, which has the same default parameters as the scikit-learn implementation (cf. Section 2). We ran the experiments on the `pan15` training data, which consists of 100 problem sets per language, with a balanced distribution of positive and negative instances. In Figures 2 and 3 we report results using 10-fold stratified cross-validation, averaged over five runs.

[4] `http://github.com/danieldk/citar`

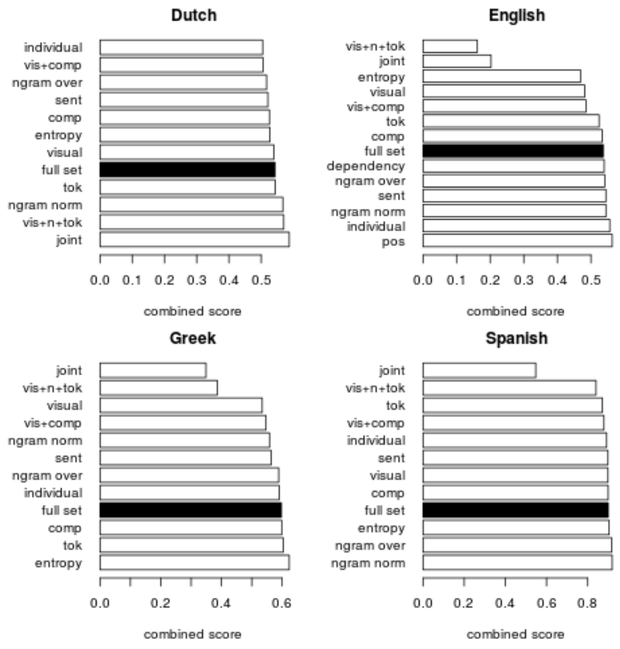

Figure 2. Combined scores when removing certain feature groups.

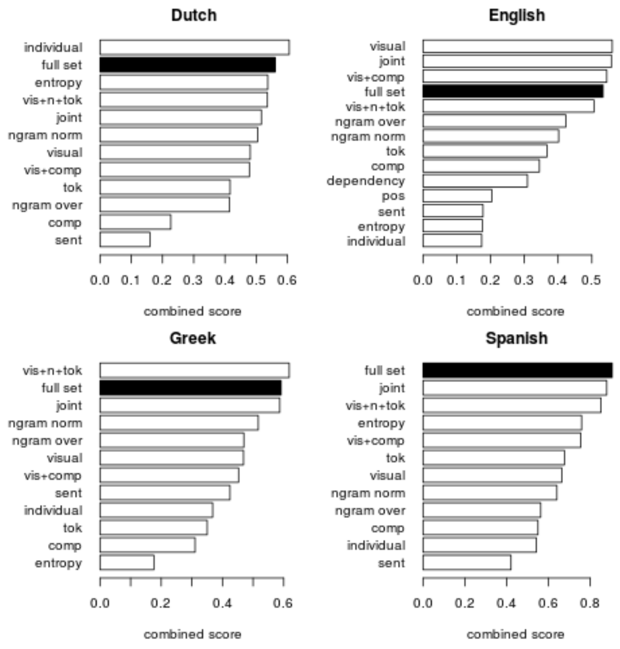

Figure 3. Combined scores when using only certain feature groups.

3.1 Ablation results

Figure 2 reports on our ablation study, in which we leave out a single feature group at a time. If removing a feature group causes lower scores, it is beneficial, while removing harmful feature groups increases performance. We can see that for Dutch, removing the *individual* features leads to the greatest drop in performance. These are therefore the most useful features. The *joint* features, on the other hand, appear to be harmful to the performance of the system. Most other feature groups do not greatly affect the results.

For English, the most useful combination includes visual features, compression features, and n-gram features. *Joint* features perform well as well, while *individual* features are harmful. Also harmful appear to be part-of-speech features. More generally, the ablation tests show that morphosyntactic features do not seem to be effective, leading us to not pursue these further for the other three languages. A possible explanation is that the quality of the dependency parser is not sufficient, however this is less likely to be the reason for the POS tagger. We leave the design of more complex (morpho)syntactic feature types for future work.

Greek and Spanish both experience the greatest drop in performance when *joint* features are removed, further underlining the usefulness of this feature group. It is interesting that only Dutch shows a preference for *individual* over *joint* features, though we could not identify a specific reason as to why this is the case, and it will require further investigation.

3.2 Single-feature results

Figure 3 shows the effects of using single groups of features only. Consistent with the results in Figure 2, *individual* features are the most useful combination for Dutch, while sentence and compression features score conspicuously low.

Visual features score highly for English (with this single feature group outperforming the full feature set), again underlining the importance of capturing layout properties for this data set.

For Greek, we find character n-grams, visual features and token features to be most beneficial. This cannot be attributed to any of the single component groups but seems to arise out of the specific combination and interaction of these feature groups.

Finally, scores on Spanish are maximised by the full feature set, which is marginally better than only the *joint* features. The combination of all available information therefore has a positive effect on Spanish, while the other languages benefit from restricting analysis to a subset of features.

3.3 Final feature sets

Based on the observations gleaned from the above tests, we grouped the features into four combinations with which we experimented in order to define the final feature set for each language. Table 1 presents the cross-validated results for the combinations across the four languages.

– `combo1`: n-grams, visual

– `combo2`: n-grams, visual, token
– `combo3`: n-grams, visual, token, entropy (*joint*)
– `combo4`: full set

Table 1. Cross-validated performance of feature combinations for the four languages.

	`combo1`			`combo2`			`combo3`			`combo4`		
Language	avg	min	max	avg	min	max	avg	min	max	avg	min	max
Dutch	.548	.542	.553	**.552**	.544	.559	.539	.532	.546	.549	.538	.557
English	.487	.476	.499	.499	.486	.507	.531	.498	.549	**.556**	.536	.572
Greek	.479	.465	.489	**.537**	.526	.550	.474	.467	.481	.510	.506	.517
Spanish	.815	.801	.827	.859	.849	.864	.900	.896	.907	**.905**	.902	.911

4 Runs and results

We submitted the runs for all four languages. The models were trained on the corresponding four training sets of `pan15`. Based on the experiments reported in Section 3, we ran our system with the full feature set for English, Dutch, and Spanish `combo4`, but with a different configuration for Greek. Indeed, we observed consistent gains in the overall score for this language when using a subset of features (`combo2`), namely n-grams, visual, and token features. For English, Dutch and Spanish, only marginal gains were observed using different combinations of features; therefore, we did not use a reduced feature set for these languages.

Table 2. Combined scores for `pan15` on the training set (cross-validation) and on the test set (single runs).

Language	Training	Test
Dutch (`full set`)	.55	.62
English (`full set`)	.56	.41
Greek (`combo2`)	.54	.60
Spanish (`full set`)	.90	.54

We report the combined AUC-c@1 scores for all languages in Table 2. On a balanced testset, these results outperform a baseline system which assigns Y (or N) throughout and thus achieves a combined score of 0.25 (c@1=0.5 and AUC=0.5). The cross-validation results on the training data show similar performance for all systems except for Spanish, where the score is much higher. This can be explained by the fact that some instances (documents) were repeated in the Spanish training set. On the test set, we obtain a combined score of around 0.6 for both Dutch and Greek, with which we achieve the third- and fourth-best result out of all pan15 participants. For English, the score of 0.41 is lower than for other languages, and contrasts somewhat with the cross-validation results. A possible explanation is that there is a considerable domain shift between the training and test sets for English.

Our method also achieves fast testing times – with some variance per language – with on average one minute per run.

5 Conclusion

We have presented a simple, yet effective approach to the authorship verification task of the PAN competition. The task was to determine whether $A_k = A_u$ for one or more known documents written by author A_k and a "questioned" document written by author A_u. Our solution to this problem was to treat it as a binary classification task, training a model across the whole dataset. Based on the prediction that texts written by different authors are less similar than texts written by the same author due to author-specific patterns in writing not under the conscious control of the author themselves, we developed an initial set of 29 features to model similarity or lack thereof.

Ablation and single-feature experiments were used to asses the contribution of the various features we originally selected, and led to the configuration of the final model for each language. Indeed, to ensure portability, we did not develop any language-specific feature (apart from NLTK language-specific sentence-splitters), and instead tuned the system by feature selection, as we noticed some differences in performance between features combinations when applied to different languages. One interesting observation was that only Dutch showed a preference for *individual* over *joint* features, but no aspect of the Dutch training data could be found which might cause this result. This raises the question for further research of whether there are theoretical grounds for preferring joint similarity measures over separate, individual measures for author verification tasks. Also, results for Greek showed that a smaller set of specific features was consistently outperforming the full set, but further investigation is required to understand exactly why.

With parsimony and speed in mind, all selected features in our final models were straight-forward to implement and easy to obtain for any text. Indeed, our resulting system is easy to run, adaptable to new languages through feature selection, and fast, with runs taking one minute on average.

The pan15 test results position our system in the third- and fourth-best place out of all participants. We obtain a somewhat lower score for English, which contrasts with promising cross-validation results. While inspection of the actual test data, once available, might shed light on the causes of such drop, this result hints at the variability which is inherent to binary classification when applied to moderate-size training and test sets. Further exploration of features, feature parameters and feature combinations for individual languages is left as a challenging avenue for future work. We make our system publicly available at https://github.com/pan-webis-de.

References

1. Bellinger, C., Sharma, S., Japkowicz, N.: One-class versus binary classification: Which and when? In: Proceedings of the 2012 11th International Conference on Machine Learning and Applications - Volume 02. pp. 102–106. ICMLA '12, IEEE Computer Society, Washington, DC, USA (2012)
2. Bird, S., Klein, E., Loper, E.: Natural language processing with Python. " O'Reilly Media, Inc." (2009)
3. Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.): Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014, CEUR Workshop Proceedings, vol. 1180. CEUR-WS.org (2014), `http://ceur-ws.org/Vol-1180`
4. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explorations Volume 11(1) (2009)
5. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review 22(2), 85–126 (2004)
6. Hollingsworth, C.: Using dependency-based annotations for authorship identification. In: Sojka, P., Horák, A., Kopecek, I., Pala, K. (eds.) TSD. Lecture Notes in Computer Science, vol. 7499, pp. 314–319. Springer (2012)
7. Japkowicz, N., Myers, C., Gluck, M., et al.: A novelty detection approach to classification. In: IJCAI. pp. 518–523 (1995)
8. Johansson, R., Nugues, P.: Extended constituent-to-dependency conversion for English. In: NODALIDA. pp. 105–112. Tartu, Estonia (2007)
9. Keselj, V., Peng, F., Cercone, N., Thomas, C.: N-gram-based author profiles for authorship attribution. In: Proceedings of the Conference of the Pacific Association for Computational Linguistics, PACLING. vol. 3, pp. 255–264 (2003)
10. Khonji, M., Iraqi, Y.: A slightly-modified gi-based author-verifier with lots of features (ASGALF). In: Cappellato et al. [3], pp. 977–983, `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-KonijEt2014.pdf`
11. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: Proceedings of the Twenty-first International Conference on Machine Learning. pp. 62–. ICML '04, ACM, New York, NY, USA (2004), `http://doi.acm.org/10.1145/1015330.1015448`
12. Li, Z.: An Exploratory Study on Authorship Verification Models for Forensic Purpose. Master's thesis, Delft University of Technology (2013)
13. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. Journal of Machine Learning Research 2, 139–154 (2001)
14. McDonald, R., Pereira, F.: Online learning of approximate dependency parsing algorithms. In: EACL (2006)
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
16. Rygl, J., Zemková, K., Kovár, V.: Authorship verification based on syntax features. In: RASLAN (2012)
17. Sapkota, U., Solorio, T., Montes-y GÃşmez, M., Rosso, P.: The use of orthogonal similarity relations in the prediction of authorship. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science, vol. 7817, pp. 463–475. Springer Berlin Heidelberg (2013)
18. Seidman, S.: Authorship verification using the impostors method – notebook for pan at clef 2013. In: Forner, P., Navigli, R., Tufis, D. (eds.) CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers (2013)

19. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review 5(1), 3–55 (2001)
20. Stamatatos, E.: A survey of modern authorship attribution methods. J. Am. Soc. Inf. Sci. Technol. 60(3), 538–556 (2009)
21. Stamatatos, E., Daelemans, W., Verhoeven, B., Stein, B., Potthast, M., Juola, P., Sánchez-Pérez, M.A., Barrón-Cedeño, A.: Overview of the author identification task at PAN 2014. In: Cappellato et al. [3], pp. 877–897, `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf`
22. Vadas, D., Curran, J.R.: Adding Noun Phrase Structure to the Penn Treebank. In: ACL (2007)
23. Yu, H.: Svmc: Single-class classification with support vector machines. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. pp. 567–572. IJCAI'03, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)