Authorship Verification with Prediction by Partial Matching and Context-free Grammar

Notebook for PAN at CLEF 2020

Łukasz Gągała

University of Göttingen
lukasz.gagala@stud.uni-goettingen.de

Abstract In our approach to authorship verification (AV) we proposed a data compression method based on the widespread *Prediction by Partial Matching* (PPM) algorithm extended with *Context-free Grammar* character preprocessing. The most frequent in-word bigrams in each text are replaced by special symbols and accepted by a modified version of PPM algorithm. For similarity measure between text samples we chose *Compression-Based Cosine* (CBC) that in previous research had been proven to perform slightly better than alternative measures.

1 Introduction

Authorship verification is an active research area of computational linguistics that can be expressed as a fundamental question of stylometry, namely whether or not two texts are written by one and the same author [13]. It has a wide range of applications in forensic linguistics and fraud and plagiarism detection. Among notable examples, we can name blackmail messages, false insurance claims or online reviews and opinion statements, where authorship analysis may turn out to be instrumental in answering forensic questions.

For 2020 edition of PAN challenge a large dataset in twofold version was prepared [2]. The dataset consists of pairs of text fragments of English language fanfiction being written either by one or two authors. For PAN 2020, similarly as in 2018 and 2019 editions, fanfiction literature was chosen as a source of the task corpus. Fanfiction literature is written by members of particular fandom, i.e. a subculture of fans concentrated around specific films, TV series, books, like Harry Potter and Star Wars, who re-enact a so-called universe of their interest by organising fan events, drawing comic books, creating multimedia content or writing new novels. The particular literary genre of literary fanfiction aims to emulate the original world of a respective fiction, but they very often enhance its plot and add new figures, places and events [3] [9]. In this regard fanfiction of different fandoms written by one person may display a lot of stylistic divergence coming from attempts to stick to a particular style that is regarded as typical for a given fandom. Authorship verification for that type of texts can, therefore, be seen as cross-domain verification similarly to cross-domain attribution [12].

2 Related work

For PAN 2020 Authorship Verification challenge we tried to enrich data-compression-based methods for authorship attribution/verification (AA/AV) with text pre-processing. Our starting point were publications on the subject by Halvani et al. [8] [7] exploring different types of compression algorithms and metrics. Furthermore, we applied character-based context-free grammar rules as a feature engineering step as described by Aljehane [1]. The final verification decision was made by as proposed in [8]. We describe carefully our approach in the subsequent section Method.

3 Method

3.1 Data Compression Methods

Data compression algorithms are used in a variety of domains (bioinformatics, social sciences, image processing) for classification and clustering tasks. The theoretical foundation of those methods lies in information theory, more precisely, in Kolmogorov complexity [14]. The length of the shortest code (description) that can produce an object is called the Kolmogorov complexity of that object.

$$K(s) = |d(s)| \tag{1}$$

Kolmogorov complexity can be used to define an information distance between two objects and has the following formula:

$$NID(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}}, \tag{2}$$

where $K(x|y)$ is a conditional complexity of x given y. *Normalised Information Distance* (NID) has been proven to satisfy the requirements for metric distance measure, however it remains incomputable [14, pp. 660]. For practical applications we can approximate NID by compression algorithms, such like *Huffman coding*, *Lempel-Ziv compression* (LZ77 and LZ78) or *Prediction by Partial Matching* (PPM) that are available in popular data compression libraries and programs, for example WINZIP or 7-ZIP [15]. Such an approximation of NID is called *Normalised Compression Distance* (NCD). In this case C value is the length of the compressed objects expressed in the number of bits.

$$NCD(x,y) = \frac{max\{C(x|y), C(y|x)\}}{max\{C(x), C(y)\}} \tag{3}$$

A couple of alternative variations of that metric measure were proposed for the task of authorship verification:

- *Compression-based Cosine* (CBC) introduced by Sculley and Brodley [16].

$$CBC(x,y) = 1 - \frac{C(x) + C(y) - C(xy)}{\sqrt{C(x)C(y)}} \tag{4}$$

- *Chen-Li Metric* (CLM) credited by Sculley and Brodley [16] to Li et al. [4].

$$CLM(x,y) = 1 - \frac{C(x) - C(x|y)}{C(xy)} \tag{5}$$

- *Compression-based Dissimilarity Measure* (CDM) proposed by Keogh et al. [10].

$$CDM(x,y) = \frac{C(xy)}{C(x) + Cy} \tag{6}$$

Drawing on the research by Halvani et al.[8], we decided to proceed with CBC and our own Python implementation of PPM in PPMC variant [6].

3.2 Prediction by Partial Matching

Compression Method algorithms (CM) are widely used across many fields of scientific research (not only in computer sciences but also in genetics or social sciences). They have gained momentum in like manner in the domain of computational linguistics. Particularly in the text classification task, the *Prediction by Partial Matching* (PPM) algorithm seems to be a preferred CM, however it is not the most compelling approach for data compression itself [8]. PPM is used in different variances (therefore we can rather speak of a family of algorithms) in many compression programs, where it is very often combined with other computational techniques [15, pp.292]. The fundamental principle of PPM is a context-dependent prediction of each subsequent character in a text string. The context is given by a window of preceding characters with a predefined length (there exists also a PPM version with a so-called unbound context [5]). The PPM algorithm with context size $= n$ can be also seen as a *Hidden Markov Model* of order n. The context size can be freely defined, yet most applications choose a range between 3 and 12, since longer context do not appear to be of practical use. Figure 1 shows the way, in which the PPM algorithm proceeds through a text taking the context of a given length and each subsequent character.

sequence	context	symbol-to-predict
Eve**r**y word has a meaning.	eve	r
Ever**y** word has a meaning.	ver	y
Every word has a meaning.	ery	_
Every **w**ord has a meaning.	ry_	w
Every w**o**rd has a meaning.	y_w	o

Figure 1: The PPM algorithm running through a text consists primary of two components: a context and a symbol-to-predict (*context* : *symbol*). Here the context size is 3 and an encoded white space is represented by underscore sign _.

3.3 Feature Engineering

The bulk of development research into PPM has been done in the domain of data compression for obvious reasons. Much less seems to be done in regard to CM for different text classification tasks – the main objective of PPM is data compression as such, and merely by the fact that similar objects (texts) tend to let themselves be compressed together more efficiently than dissimilar objects do. A superior compression rate of a particular approach does not necessarily lead to better classification results, so that CM may be an attractive starting point for AA/AV but they require some dedicated amendments. One of the recently proposed improvements is a context-free grammar preprocessing for PPM (*Grammar-based preprocessing for PPM*, GB-PPM), which focuses on a more dense representation of character strings [1]. The GB-PPM algorithm preprocessing replaces in subsequent runs n most frequent bigrams or trigrams of characters with special symbols reducing the overall length of a text and simplifying the distribution of characters for different contexts. First, GB-PPM identifies frequent n-grams (in our case bigrams, in this toy example the status of being frequent is assigned arbitrarily):

(7) $\underline{\text{Ev}}\text{ery }\underline{\text{wo}}\text{rd has a m}\underline{\text{ea}}\text{n}\underline{\text{in}}\text{g.}$

Then the frequent n-grams are replaced with special symbols (e.g. $\mathcal{ABC}$...), which are not a part of the set of characters in the text from Eq. 7:

(8) $\mathcal{A}\text{ery }\mathcal{B}\text{rd has a m}\mathcal{C}\text{n}\mathcal{D}\text{g.}$

The algorithm can be run once again, identifying the next set of frequent n-grams, this time together with special symbols standing for frequent n-grams from the last run in Eq. 8:

(9) $\mathcal{A}\underline{\text{er}}\text{y }\underline{\mathcal{B}\text{r}}\text{d h}\underline{\text{as}}\text{ a }\underline{\text{m}\mathcal{C}}\text{n}\mathcal{D}\text{g.}$

Also, those frequent n-grams can be replaced further with special symbols covering both original characters and other special symbols. After n runs (in our example $n = 2$) the algorithm can yield a new version of the text string in Eq. 10 and a corresponding character context-free grammar in Figure 2:

(10) $\mathcal{A}\mathcal{E}\text{y }\mathcal{F}\text{d h}\mathcal{G}\text{ a }\mathcal{H}\text{n}\mathcal{D}\text{g.}$

$$\begin{aligned}
\mathcal{A} &\rightarrow \mathcal{E}\text{v}\\
\mathcal{B} &\rightarrow \text{wo}\\
\mathcal{C} &\rightarrow \text{ea}\\
\mathcal{D} &\rightarrow \text{in}\\
\mathcal{E} &\rightarrow \text{er}\\
\mathcal{F} &\rightarrow \mathcal{B}\text{r}\\
\mathcal{G} &\rightarrow \text{as}\\
\mathcal{H} &\rightarrow \text{m}\mathcal{C}
\end{aligned}$$

Figure 2: A character free-context grammar resulting from transformation in Eq. 7 - 10.

The generic code for GB-PPM is given in Algorithm 1. The procedure is being applied directly before the compression of each text sample, but one can easily imagine an alternative implementation, where a single run is performed over the whole text corpus $\mathcal{D}$, so that the most frequent n-grams $\mathcal{B}$ are derived rather from $\mathcal{D}$, $\mathcal{B} \leftarrow \mathcal{D}$, than a particular text t, $t \in \mathcal{D}$.

Algorithm 1: Non-finite symbol preprocessing with the GR-PPM algorithm

```
Input:
$\mathcal{T}$ (sequence),
$i$ (number of passes),
$n$ (number of most frequent bigrams in the current sequence),
Output: sequence $\mathcal{T}$ with non-terminal symbols
$C \leftarrow$ PREDICTION BY PARTIAL MATCHING
for $k \leftarrow 0$ to $i$ do
    $\mathcal{B} \leftarrow$ Find the $n$ most frequent in-word bigraphs in the current text $\mathcal{T}$;
    foreach frequent bigram $b \in \mathcal{B}$ do
        foreach frequent bigram $b \in \mathcal{T}$ do
            non-terminal symbol $s \leftarrow$ frequent bigram $b$;
return $C(\mathcal{T})$
```

3.4 Verification

To verify the authorship of a pair of texts $\rho = (T_x, T_y)$, we need to find a decision threshold θ, for which our data compression dissimilarity measure $\mathcal{M}$ gives us the score $s_\rho = \mathcal{M}(T_x, T_y)$, so that:

$$\text{decision}(\rho) = \begin{cases} \texttt{Y} \;\; (Yes), & \text{if } s_\rho < \theta \\ \texttt{N} \;\; (No), & \text{otherwise} \end{cases}$$

Identically as in [8], we find θ by using the *Equal Error Rate* (EER) algorithm on n text pairs with the equal number of positive (Y) and negative (N) authorship cases. In this way we can determine such θ, that the rates of false positives and false negatives are equal.

Algorithm 2: Determine θ threshold by EER

```
Input: Y (the dissimilarity scores of the Y problems), N (the dissimilarity
       scores of the N problems)
Output: $\theta$ threshold
if length($Y$) $\neq$ length($N$) then
    Exception $\leftarrow$ "Number of Y and N problems mismatch!";
    throw Exception;
Sort Y and N in ascending order;
$\mathscr{l} \leftarrow$ length($Y$);
$\mathscr{i} \leftarrow 0$;
$\mathscr{j} \leftarrow \mathscr{l} - 1$ ;
for $\mathscr{k} \leftarrow 0$ to $\mathscr{l} - 1$ do
    if $Y_i < N_j$ then
        $\mathscr{i} \leftarrow \mathscr{i} + 1$;
        $\mathscr{j} \leftarrow \mathscr{j} - 1$;
        continue;
    if $Y_i = N_j$ then
        $\theta \leftarrow Y_i$;
        break;
    if $Y_i > N_j$ then
        if $\mathscr{i} = 0$ then
            $\theta \leftarrow \frac{1}{2}(Y_i + N_j)$ ;
        else
            $\theta \leftarrow \frac{1}{2}(min(Y_i, N_j) + min(Y_{i-1}, N_j)$ ;
        break;
if $\mathscr{i} = \mathscr{l}$ then
    $\theta \leftarrow \frac{1}{2}(Y_{i-1} + N_{j+1})$ ;
return $\theta$ ;
```

4 Implementation Details and Submission

The development dataset provided by the PAN 2020 organisers consists of two versions: a smaller one that is supposed for data-sparse machine learning methods (52,601 text snippet pairs) and a bigger one that should be more applicable for data-hungry approaches (275,565 text snippet pairs) [11]. Both subcorpora were drawn from even a bigger text collection consisting of 5.8 million stories written by 1.4 million authors in 44 different languages and in 10,328 different topical domains. The data is a set of problems being pairs of English language text fragments by the same or two different authors. Moreover, the texts in the pair come from different fanfics, however there are no fanfiction crossovers, i.e. literary texts that encompass multiple fiction universes, e.g.

Batman and Star Wars, Harry Potter and Spider-Man. The overall distribution of authors across all the problems resembles the "long-tail" distribution of the original text corpus, from which the development set was composed.

To establish the verification decision threshold we used 2000 text pairs. Larger amounts of text pairs did not improve our calibration parameter. For GB-PPM algorithm we decided to process only bigrams in a single run, since additional executions of that preprocessing step gave no improvement. As a dissimilarity measure we chose CBC as described in Section Method.

As we can see in Table 1 our approach yielded better results as baseline methods suggested by the organisers, but could not break a "glass ceiling" of around 0.8 overall score – only two approaches by Boenninghoff and by Weerasinghe achieved the level of around 0.9 overall score.

Rank	Team	Training dataset	AUC	c@1	F0.5u	F1-score	Overall
1	boenninghoff20	large	0.969	0.928	0.907	0.936	0.935
2	weerasinghe20	large	0.953	0.880	0.882	0.891	0.902
3	boenninghoff20	small	0.940	0.889	0.853	0.906	0.897
4	weerasinghe20	small	0.939	0.833	0.817	0.860	0.862
5	halvani20b	small	0.878	0.796	0.819	0.807	0.825
6	kipnis20	small	0.866	0.801	0.815	0.809	0.823
7	araujo20	small	0.874	0.770	0.762	0.811	0.804
8	niven20	small	0.795	0.786	0.842	0.778	0.800
9	gagala20	small	0.786	0.786	0.809	0.800	0.796
10	araujo20	large	0.859	0.751	0.745	0.800	0.789
11	baseline (naive)	small	0.780	0.723	0.716	0.767	0.747
12	baseline (compression)	small	0.778	0.719	0.703	0.770	0.742
13	ordonez20	large	0.696	0.640	0.655	0.748	0.685
14	faber20	small	0.293	0.331	0.314	0.262	0.300

Table 1: Comparison of performance for different approaches.

Acknowledgements

The research for the author's work has been funded with a PhD bursary for years 2016-2019 by the *Cusanuswerk Academic Foundation*.

References

1. Aljehane, N.O.M.: Grammar-based preprocessing for PPM compression and classification. Ph.D. thesis, Bangor University (2018)
2. Bischoff, S., Deckers, N., Schliebs, M., Thies, B., Hagen, M., Stamatatos, E., Stein, B., Potthast, M.: The Importance of Suppressing Domain Style in Authorship Analysis. CoRR abs/2005.14714 (May 2020), https://arxiv.org/abs/2005.14714
3. Booth, P.: A Companion to Media Fandom and Fan Studies. Wiley Blackwell companions in cultural studies, Wiley (2018), https://books.google.de/books?id=yDJKDwAAQBAJ

4. Chen, X., Kwong, S., Li, M.: A compression algorithm for dna sequences and its applications in genome comparison. pp. 52–61 (1999)
5. Cleary, J.G., Teahan, W.J., Witten, I.H.: Unbounded length contexts for ppm. In: Proceedings DCC '95 Data Compression Conference. pp. 52–61 (1995)
6. Gagala, L.: Code for PAN 2020 Authorship Verification task. `https://github.com/Lukasz-G/PAN2020` (2020)
7. Halvani, O., Graner, L., Vogel, I.: Authorship verification in the absence of explicit features and thresholds. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) Advances in Information Retrieval. pp. 454–465. Springer International Publishing, Cham (2018)
8. Halvani, O., Winter, C., Graner, L.: On the usefulness of compression models for authorship verification. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. ARES '17, Association for Computing Machinery, New York, NY, USA (2017), `https://doi.org/10.1145/3098954.3104050`
9. Hellekson, K., Busse, K.: The Fan Fiction Studies Reader. University of Iowa Press (2014), `http://www.jstor.org/stable/j.ctt20p58d6`
10. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards parameter-free data mining. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 206–215. KDD '04, Association for Computing Machinery, New York, NY, USA (2004), `https://doi.org/10.1145/1014052.1014077`
11. Kestemont, M., Manjavacas, E., Markov, I., Bevendorff, J., Wiegmann, M., Stamatatos, E., Potthast, M., Stein, B.: Overview of the Cross-Domain Authorship Verification Task at PAN 2020. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2020)
12. Kestemont, M., Tschuggnall, M., Stamatatos, E., Daelemans, W., Specht, G., Stein, B., Potthast, M.: Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection. In: Cappellato, L., Ferro, N., Nie, J.Y., Soulier, L. (eds.) Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum (CLEF 2018). Springer (Sep 2018)
13. Koppel, M., Winter, Y.: Determining if two documents are written by the same author. Journal of the Association for Information Science and Technology 65(1), 178–187 (2014), `https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.22954`
14. Li, M., Vitnyi, P.M.: An Introduction to Kolmogorov Complexity and Its Applications. Springer Publishing Company, Incorporated, 3 edn. (2008)
15. Salomon, D., Motta, G.: Handbook of Data Compression. Springer Publishing Company, Incorporated, 5th edn. (2009)
16. Sculley, D., Brodley, C.: Compression and machine learning: A new perspective on feature space vectors. pp. 332– 341 (04 2006)