Author Masking Directed by Author's Style

Notebook for PAN at CLEF 2018

Mostafa Rahgouy[1], Hamed Babaei Giglou[1], Taher Rahgooy[2], Hasan Zeynali[1], and Salar Khayat Mirza Rasouli[1]

[1] University of Mohaghegh Ardabili, Computer Science Department, Ardabil, Iran
{MostafaRahgouy, hamedbabaeigiglou, hasanzeynalimc, salarkhayatrasooli}@gmail.com
[2]Tulane University, Computer Science Department, New Orleans, LA, USA
trahgooy@tulane.edu

Abstract Author verification algorithms mainly rely on learning statistical finger-prints of authors. In the other hand, most of the previous algorithms in author masking try to apply changes to the original texts blindly without considering those finger-prints. In this paper, we propose an approach that learns author's finger-prints and uses them to apply directed transformations and distortions to the original text. We represent author finger-prints with different statistics such as word choice distribution, sentence length preference, etc. obtained from author's known texts. Automatic and manual evaluations of the obfuscated texts are very promising and show the effectiveness of our approach.

1 Introduction

The author masking task of PAN 2018 is to paraphrase a given document so that author-ship verification algorithms fail to detect it's author . The problem consist of a set of documents written by a specific author and another document of the same author named "original". The challenge is to use the information in the provided set of documents in order to obfuscate the "original" document.

Statistical properties of words, phrases, and sentences in documents play a pivotal role in identifying the author's style and finger-prints. For example average sentence length, text entropy, and word usage frequency alongside many other statistics is used by [8,1,2] to learn the author finger-prints either directly or indirectly. However, they apply their changes to the original texts irrespective of the author's style. As a result, some of those changes enforces the finger-prints of the author rather than diminish them.

Therefore, inspired by [10] work, we propose an approach that learns the author's finger-prints statistics(e.g. sentence length, word usage, etc.) from author's known documents and tries to distort those statistics in the target document using word replacement, combining/splitting sentences, and contraction transformation. In the rest of this paper, we explain the proposed approach and it's steps in Section 2, automatic and manual evaluations are covered in Section 3, and finally we summarize the paper findings in Section 4.

2 Proposed Approach

First, we learn the author's finger-prints by calculating several global statistics from documents written by the author (same-author documents). Next, we obfuscate the original text sentence by sentence considering the global statistics in the whole document and author's finger-prints. After splitting the text to sentences, we try to distort the text using word/phrase transformations, splitting and combining sentences, and word/phrase replacements in such a way that the resulting document has different statistics compared to the statistics obtained from other documents of the same author.

2.1 Preprocessing

The first step in the proposed algorithm is to prepare the examples for obfuscation. In this step we split the original text to sentences using NLTK 3.0 Toolkit [3]. Next, using a pre-defined lexicon we split each sentence to phrase chunks and tokens. In other words, if we encounter a phrase from the lexicon we consider it as a chunk otherwise we use the tokens obtained from the tokenizer. The phrase lexicon is manually crafted from important phrases used in the same-author documents in the train-set.

2.2 Author word usage distribution

Word choice and usage is one of the important components of every author's finger-prints. In order to capture this important component we use a maximum likelihood estimate of the word lemma frequencies in the *same-author* documents excluding stop words and auxiliary verbs. We denote the probability of choosing word w by $P(w)$.

2.3 Word/Phrase Replacement

In this section we explain the process we used to replace words in the original document. Let d_{orig}, d_{obf}, w, and $used$ be the original document, obfuscated document, current word, and list of words that already used for replacement respectively. We initialize the obfuscated document by the original document. Next, we iteratively choose words from the obfuscated document and perform replacements using Algorithm 1. As illustrated in Algorithm 1, we only consider adjectives, adverbs and verbs(excluding auxiliary verbs) for replacement. The replacement candidates are obtained from either *WordNet* [5] or *Word2Vec* [11]. Next, we choose the best scoring replacement from those that are less likely to be chosen by the author(have smaller P) and also not used before. We score the appropriateness of a candidate replacement by averaging over three sub-scores:

1. **Similarity to the original word:** this score helps us to find meaningful replacements. This score obtained from *WordNet* or *Word2Vec* depending on the model that we evaluate.
2. **Inverse of author's word usage probability:** we want to use replacement that are less likely to be selected by the author.

3. **Word mover's distance(WMD)[9]:** The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. In other words the most efficient way to "move" the distribution of a document to the distribution of another document. Using WMD we can progressively measure the distance between the obfuscated document and the original document and choose a replacement that increases this measure. The *ReplaceAll* method used in the algorithm replaces all occurrences of word w in document d by replacement word r. This method returns a document with replacements and doesn't change the input document.

Algorithm 1: ReplaceWord $(d_{obf}, d_{orig}, w, used)$

```
1  REPLACEWORD (d_obf, d_orig, w, used)
2    if POS(w) in {ADJ, ADV, VERB} then
3      (syns, similarities) ← GETSYNONYMS (w)
4      (r, r_score) ← (null, 0)
5      for i ← 1 to len(syns) do
6        if syns[i] in used or P (w) ≤ P (syns[i]) then
7          continue
           /* replace all w with syns[i] and put it in d_temp
              without changing d_obf                          */
8        d_temp ← REPLACEALL(d_obf, w, syns[i])
9        s ← (similarities[i] + 1/(1+P(syns[i])) + WMD(d_temp, d_orig))/3
10       if s > r_score then
11         r_score ← s
12         r ← syns[i]
13     if r_score > 0 then
         // apply the best scoring replacement to d_obf
14       d_obf ← REPLACEALL(d_obf, w, r)
15       used ← used ∪ {r}
16   return d_obf
```

2.4 Contraction Transformation

We follow the method used in [4] to transform contractions to expanded form and vise versa. In order to perform these transformations we use the average preference of the author for using contractions in the same-author documents. If author prefers contractions then we try to convert them to expanded form in the obfuscated document and vise versa.

2.5 Sentence Length Alteration

According to [8], sentence length is a deciding factor in authorship verification. As a result we consider it as a component of author's style and change the average sentence length of the obfuscated text based on preference of the author for sentence length in the same-author documents. If the author prefers short sentences, we try to combine consecutive sentences using *and* connector to increase the average length in the obfuscated text. In contrary, when author prefers long sentences, we search for connectors (e.g. words with CC POS-tag) in each long sentence and split it to two sentences.

3 Experimental Evaluation

Author obfuscation task is evaluated by three performance dimensions. Accordingly, we call an obfuscation software [6]:

1. **safe:** if its obfuscated texts cannot be attributed to their original authors anymore
2. **sound:** if its obfuscated texts are textually entailed by their originals
3. **sensible:** if its obfuscated texts are well-formed and inconspicuous

From these criteria only *safety* criterion is evaluated automatically and the other two are evaluated by human judgments. As discussed in Section 2.3 we rely on either *WordNet* or *Word2Vec* to find best replacements for words. Thus, we submitted two version of our algorithm for the task, one with *Word-Net* and another with *Word2Vec*. At the time of writing this paper the test results were not published. Therefore, in the following, we provide some examples and performance statistics of our proposed approach on training data using *Word2Vec*. The proposed algorithm and automatic evaluation are implemented in *Python 3.6* and they are published on *GitHub*[*].

Example	Original text	Obfuscated text
1. Word replacement	For some people this is **very** devastating.	For some people this is **extremely** devastating.
2. Contraction	Whenever he speaks **it is** to **say** things like "Give me this" or "Ben **wants** that".	Whenever he speaks **it's** to **believe** things like "Give me this" or "Ben **deserves** that"
3. Expansion	**Don't** patronise them, sir–pray, **don't** patronise them.	**Do not** patronise them , sir–pray , **do not** patronise them.
4. Sentence split	Actually, the difficulty to draw lies rather in learning how to observe **or** being able to switch over to a particular way of seeing...	Actually, the difficulty to draw lies rather in learning how to observe**.** being able to switch over to a particular way of seeing...

Table 1. Manually evaluated examples

[*] https://github.com/Rahgooy/author_masking

3.1 Manually Evaluated Examples

In this section we provide some manually evaluated examples. In the first example in Table 1 the adverb *very* is replaced by a very good alternative, *extremely*. As you can see in this example *is* is unchanged because it is an auxiliary verb. Second example shows a successful replacement of an expanded form *it is* with it's contraction *it's* for an author that prefers expanded forms over contractions. In addition, we have replacements for verbs *say and wants* with *believe and deserves* respectively. Conversely, in the third example, a contracted form phrase *don't* is converted to its expanded form *do not* due to the author preference for contractions.

The original text in the example 4 contains a long sentence from an author that prefers long sentences. We split this sentence by replacing conjunction word *or* with punctuation. It is clear that the resulting sentences are not complete and meaningful in this example.

3.2 Automatic Evaluation

To evaluate our approach automatically, we trained a long-short term memory network (LSTM) [7] on the same author documents. We separated the texts in the same-author documents of each problem to train and evaluation sets. For training we randomly

Example	Original score	Obfuscated score
Problem203.txt	0.82	0.33
Problem164.txt	0.46	0.18
Problem044.txt	0.66	0.60
Problem079.txt	0.51	0.44
Problem148.txt	0.97	0.89
Problem185.txt	0.55	0.48
Problem024.txt	0.36	0.19
Problem152.txt	0.70	0.74
Problem175.txt	0.98	0.60
Problem095.txt	0.58	0.54
Problem150.txt	0.96	0.68
Problem080.txt	0.48	0.48
Problem160.txt	0.99	0.57
Problem132.txt	0.50	0.41
Problem015.txt	0.54	0.50
Problem040.txt	041	0.45
Problem195.txt	0.61	0.53
Problem133.txt	0.86	0.85
Problem069.txt	0.85	0.88
Problem183.txt	0.65	0.35

Table 2. Automatic evaluation. Scores above 0.5 are identified as same author.

sampled fixed-length text from the documents and labeled them as positive examples. For negative examples we chose a number of documents from other problems randomly and generated approximately the same number of examples as number of positives.

We tested the model using the original document for each problem. The average verification score obtained by this model for original texts was **75.61%**. The average score dropped dramatically to **51.71%** when we applied the same model on the obfuscated data. Table 2 shows the results for 20 examples from the dataset.

4 Conclusion

In this paper we proposed an algorithm for author masking task in PAN 2018. The proposed algorithms learns the author's finger-prints from known documents of the author. Next, it uses those statistics to apply calculated changes to the text in order to deform author's finger-prints in the obfuscated text effectively. Manual and automatic evaluation shows that our approach is very capable of diminishing author's finger-prints. In future works we try to add more components to the author's finger-prints and find clever ways to distort them in the obfuscated texts.

References

1. Bagnall, D.: Author identification using multi-headed recurrent neural networks. arXiv preprint arXiv:1506.04891 (2015)
2. Bartoli, A., Dagri, A., De Lorenzo, A., Medvet, E., Tarlao, F.: An author verification approach based on differential features. In: CEUR WORKSHOP PROCEEDINGS. vol. 1391. CEUR (2015)
3. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)
4. Castro-Castro, D., Bueno, R.O., Muñoz, R.: Author masking by sentence transformation. In: CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1866. CEUR-WS.org (2017)
5. Fellbaum, C.: WordNet: An Electronic Lexical Database. Bradford Books (1998)
6. Hagen, M., Potthast, M., Stein, B.: Overview of the author obfuscation task at pan 2017: safety evaluation revisited. Working Notes Papers of the CLEF pp. 33–64 (2017)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
8. Hürlimann, M., Weck, B., van den Berg, E., Suster, S., Nissim, M.: Glad: Groningen lightweight authorship detection. In: CLEF (Working Notes) (2015)
9. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: International Conference on Machine Learning. pp. 957–966 (2015)
10. Mansoorizadeh, M., Rahgooy, T., Aminiyan, M., Eskandari, M.: Author obfuscation using wordnet and language models-notebook for pan at clef 2016
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)