University of the Aegean

School of Engineering

Department of Information and
Communication Systems Engineering

MSc in Research in Information and
Communication Systems

Cross-domain Authorship Attribution Using Pre-trained Language Models

Δια-τομεακή Αναγνώριση Συγγραφέα με Χρήση Προ-εκπαιδευμένων Γλωσσικών Μοντέλων

Georgios Barlas

supervised by
Dr. Efstathios Stamatatos

June 2020

Abstract

Cross-domain authorship attribution is a category of realistic authorship attribution problems on terms of applications mainly in forensics. In a cross-domain scenario the texts of known authorship (training set) are on different domain (cross-domain) than the texts of unknown authorship (test set). The use of pre-trained language models in various natural language processing tasks inspired us to explore their potentialities in authorship attribution problem. In this paper, we experiment with four architectural different pre-trained language models (BERT, ELMo, GPT-2 and ULMFiT). The proposed method is a modification of a successful authorship verification approach based on a multi-headed neural network language model to combine with pre-trained language models. We evaluated the proposed method on two corpora (CMCC, PAN18) on three cross-domain scenarios (cross-topic, cross-genre and cross-fandom). The achieved results are very promising and they demonstrate the crucial effect of the normalization corpus in cross-domain attribution.

Περίληψη

Η δια-τομεακή αναγνώριση συγγραφέα είναι μια κατηγορία ρεαλιστικών προβλημάτων αναγνώρισης συγγραφέων με όρους εφαρμογών κυρίως στην εγκληματολογία. Στα δια-τομεακά σενάρια τα κείμενα με γνωστό συγγραφέα (σετ εκπαίδευσης) είναι σε διαφορετικό τομέα (δια-τομεακή) από τα κείμενα αγνώστου συγγραφέα (σετ δοκιμών). Η χρήση προ-εκπαιδευμένων γλωσσικών μοντέλων σε διάφορα καθήκοντα επεξεργασίας φυσικής γλώσσας μας ενέπνευσε να διερευνήσουμε τις δυνατότητές τους στο πρόβλημα της αναγνώρισης του συγγραφέα. Σε αυτή την εργασία, πειραματιζόμαστε με τέσσερα διαφορετικής αρχιτεκτονικής προ-εκπαιδευμένα γλωσσικά μοντέλα (BERT, ELMo, GPT-2 και ULMFiT). Η προτεινόμενη μέθοδος είναι μια τροποποίηση μιας επιτυχούς προσέγγισης επαλήθευσης συγγραφέα, που βασίζεται σε ένα μοντέλο γλώσσας νευρωνικών δικτύων πολλαπλών κεφαλών για να συνδυαστεί με τα προ-εκπαιδευμένα γλωσσικά μοντέλα. Αξιολογήσαμε την προτεινόμενη μέθοδο σε δύο συλλογές κειμένων (CMCC, PAN18) σε τρία δια-τομεακά σενάρια. Τα επιτευχθέντα αποτελέσματα είναι πολύ ελπιδοφόρα και καταδεικνύουν την κρίσιμη επίδραση του σετ κανονικοποίησης στην δια-τομεακή αναγνώριση συγγραφέα.

Contents

Introduction

Authorship Attribution (AA) is a very active area of research dealing with the identification of persons who wrote specific texts [21, 33]. Typically, there is a list of suspects and a number of texts of known authorship by each suspect and the task is to assign texts of disputed authorship to one of the suspects. The basic forms of AA are closed-set attribution (where the list of suspects necessarily includes the true author), open-set attribution (where the true author could be excluded from the list of suspects), and author verification (where there is only one candidate author). The main applications of this technology are in digital forensics, cyber-security, digital humanities, and social media analytics [13, 26].

In real life scenarios the known and the unknown texts may not share the same properties. The topic of the texts may differ but also the genre (e.g., essay, email, chat). Cross-domain AA examines those cases where the texts of known authorship (training set) differ with respect to the texts of unknown authorship (test set) in topic (cross-topic AA) or in genre (cross-genre AA) [30, 36]. The main challenge here is to avoid the use of information related to topic or genre of documents and focus only on stylistic properties of texts related to the personal style of authors.

Another cross-domain and more challenging than cross-topic and cross-genre scenarios is the cross-fandom scenario, where the documents of unknown authorship are fics from the same fandom (target fandom) and the document of known authorship are fics of several fandoms other than the target fandom. Because of the explicit intertextuality (i.e. borrowings from the original canon), it can be anticipated that the style and content of the original canon shave a strong influence on the fanfics, because these often aim to imitate the style of the canon's original authors.

Recently, the use of pre-trained language models (e.g., BERT, ELMo, ULMFiT, GPT-2) has been demonstrated to obtain significant gains in several text classification tasks including sentiment analysis, emotion classification, and topic classification [4, 11, 22, 24]. However, it is not yet clear whether they can be equally useful for style-based text categorization tasks.

Especially, in cross-topic AA, information about the topic of texts can be misleading.

An approach based on neural network language models achieved top performance in recent shared tasks on authorship verification and authorship clustering (i.e., grouping documents by authorship) [27, 37]. This method is based on a character-level recurrent (RNN) neural network language model and a multi-headed classifier (MHC) [1]. So far, this model has not been tested in closed-set attribution which is the most popular scenario in relevant literature. In this paper, we adopt this approach for the task of closed-set AA and more specifically the challenging and realistic cross-domain AA scenarios.

We examine the use of pre-trained language models (e.g., BERT, ELMo, ULMFiT, GPT-2) in AA and the potentials of MHC. We also demonstrate that in cross-domain AA conditions, the effect of an appropriate normalization corpus is crucial.

Previous Work

The vast majority of previous work in AA focus on the closed-set attribution scenario. The main issues is to define appropriate stylometric measures to quantify the personal style of authors and the use of effective classification methods [21, 33].

A relatively small number of previous studies examine the case of cross-topic AA. In early approaches, features like function words or part-of-speech n-grams have been suggested as less likely to correlate with topic of documents [16, 17]. However, one main finding of several studies is that low-level features, like character n-grams, can be quite effective in this challenging task [30, 34]. Typed character n-grams provide a means for focusing on specific aspects of texts [28]. Interestingly, character n-grams associated with word affixes and punctuation marks seem to be the most useful ones in cross-topic AA. Another interesting idea is to apply structural correspondence learning using punctuation-based character n-gram as pivot features [29]. Recently, a text distortion method has been proposed as a pre-processing step to mask topic-related information in documents while keeping the text structure (i.e., use of function words and punctuation marks) intact [36].

There have been attempts to use language modeling for AA including traditional n-gram based models as well as neural network-based models [1, 7, 8]. The latter is closely related to representation learning approaches that use deep learning methods to generate distributed text representations [5, 15]. In all these cases, the language models are extracted from the texts of known authorship. As a result, they heavily depend on the size of the training set per candidate author.

A series of digital text forensics tasks named PAN started in 2009 [23], in 2018 included also a closed-set cross-domain AA task [38], where the selected scenario was a cross-fandom scenario. There were eleven submissions in total from several countries. Most of the approaches based on character and word n-grams, while TF and TF-IDF where the most popular weighting/normalization schemes. The winning method of Custódio and Paraboni [3] was an ensemble of three simple authorship attribution approaches based

on character and word n-gram features and a distorted version of texts [35]. The second-best method of Murauer et al. [20] used dynamic adaptation of parameter values for each attribution problem separately and the and third-best method of Halvani and Graner [10] based on text compression. In overall, the results of the task indicate that simple and language-independent features are more effective in comparison to more sophisticated approaches based on linguistic analysis and deep learning.

The Proposed Method

The Model

An AA task can be expressed as a tuple (A, K, U) where A is the set of candidate authors (suspects), K is the set of known authorship documents (for each $a \in A$ there is a $K_a \subset K$) and U is the set of unknown authorship documents. In closed-set AA, each $d \in U$ should be attributed to exactly one $a \in A$. In cross-domain (e.g. topic, genre, fandom) AA, the domain of documents in U is distinct with respect to the domains found in K.

Bagnall introduced an AA method[1] [1] and obtained top positions in shared tasks in authorship verification and authorship clustering [27, 37]. The main idea is that a character-level RNN is produced using all available texts by the candidate authors while a separate output is built for each author (MHC). Thus, the recurrent layer models the language as a whole while each output of MHC focuses on the texts of a particular candidate author. To reduce the vocabulary size, a simple pre-processing step is performed (i.e., uppercase letters are transformed to lowercase plus a symbol, punctuation marks and digits are replaced by specific symbols) [1].

The model, as presented in Figure 1, consists of three parts, the language model, a Filter which filters the representations of the LM and the multi-head classifier (MHC). The LM consists of a tokenization layer and the pre-trained language model, its role is to generate a representation for each token in the given text, Algorithm 1. The Filter allows to pass only the representations that are referring to tokens whose next token belonging to vocabulary, Algorithm 2. That way for each representation given to MHC, the token to be predicted will belong to vocabulary, which contains the N most frequent tokens in K set. We filter the representations and not the tokens, in that way the information of all tokens (even those that are not belonging to the vocabulary) is encapsulated in the representations. The calculations made for the rejected representations are affecting the representations through the

[1] https://github.com/pan-webis-de/caravel

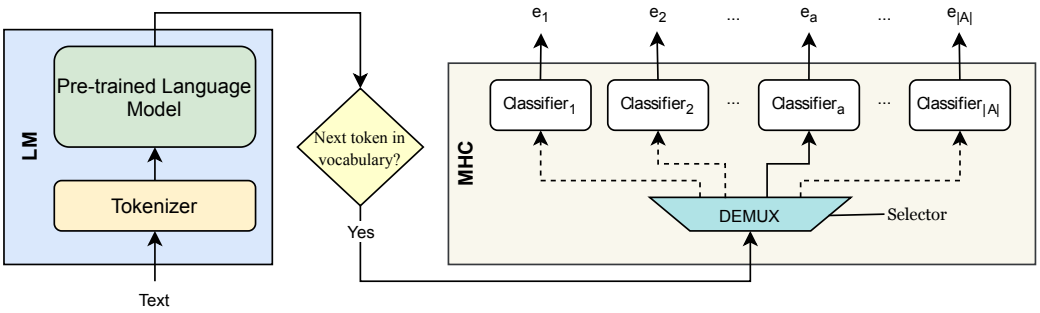


Figure 1: The proposed model consists of two parts, the language model (LM) and the multi-headed classifier (MHC). The DEMUX layer in MHC part functions as a demultiplexer, its state is defined by the selector. During training phase the selector is defined by the author of the input text and during calculation of normalization vector or test phase the input of DEMUX is connected to all its outputs.

hidden states of the LM.

MHC comprises a set of $|A|$ classifiers, where $|A|$ is the number of candidate authors and a demultiplexer which helps to select the desirable classifier, Algorithm 2. Each classifier has N inputs, where N is the dimensionality of the LM's representation, and V outputs, where V is the size of the vocabulary. During training only one classifier is connected to the LM, the one that corresponds to the author of the given text, Algorithm 4. But during evaluation or test phase all classifiers are connecting to the LM in order to obtain the winner author by comparing the outputs, Algorithm 5. Moreover, only the classifiers in MHC are trained during training phase and not the LM.

Each classifier in MHC is trained to predict the text flow on documents of the corresponding author. Given a document, as output we obtain the cross-entropy error of each classifier and at this points we could obtain the winner author for each document in U by simply comparing the outputs of MHC.

Summarizing the above, the idea of the approach is that since the documents are on the same language, most of their content is similar. So, the use of a common LM for each classifier will force the classifiers to respond similar to the common content and each classifier will perform better than the other classifiers on the idioms of the author it has been trained on.

However, the scores obtained for each candidate author (classifier) are not directly comparable due to different bias at each classifier of MHC. To handle this drawback, the outputs are normalized with the help of normalization

vector n [1], Equation 1.

$$n = \frac{1}{|C|} \sum_{d \in C} H(d) \tag{1}$$

where C is the normalization corpus and H a vector that denotes the cross-entropy errors produced by the MHC for the given document. In other words, vector n is the average cross-entropy error of each author on all documents in C. As you may already noticed in Equation 1, the normalization corpus C is an unlabeled corpus, there is no need to be known the authors of its documents. So, C may be an arbitrary set of documents but it may even be the documents of known authorship, set K or even the documents of unknown authorship, set U, since only the text and not the author of each document is required to estimate normalization vector n, Algorithm 6.

Finally, Equation 2 indicates the criterion of how the winner author a for a document $d \in U$ is determined and the way the normalization vector is applied, Algorithm 5.

$$\arg\min_{a} \left(H(d) - n\right) \tag{2}$$

In this paper, we extended Bagnall's model in order to accept tokens as input and we propose the use of a pre-trained language model to replace RNN in the aforementioned AA method. The RNN proposed by Bagnall [1] is trained using a small set of documents (K for closed-set AA). In contrast, pre-trained language models have been trained using millions of documents in the same language. Moreover, RNN is a character-level model while the pre-trained models used in this study are token-level approaches.

The Pre-trained Language Models

In this study we selected to experiment with four different pre-trained LMs BERT, GPT-2, ELMo and ULMFiT. The selection of the LMs based on the directionality, depth and the acceptance from Natural Language Processing community.

BERT (Bidirectional Encoder Representations from Transformer) is a LM that creates contextualized word representations. Its architecture is a multi-layer bidirectional Transformer encoder as introduced in [39]. Initially, two variations of the same model tested the *base* model and the *large* model. The *base*/*large* model consist of 12/24 layers of Transformer blocks, the hidden size is 768/1024, the number of self-attention heads are 12/16 and contains 110/340 million parameters in total. The input format of BERT's model

allows the use of one or two sentences at once. In case of two sentences the format is "[CLS] A [SEP] B [SEP]", where A and B are the two sentences, and "[CLS]" and "[SEP]" are special tokens indicating the beginning of an input and the end of a tokens sequence, respectively. In case of using just one sentence the format is "[CLS] A [SEP]". In both cases the maximum length in tokens is 512 including "[CLS]" and "[SEP]" tokens.

BERT was trained on two different tasks Masked LM (MLM) and Next Sentence Prediction (NSP). On MLM, some percentage of the input tokens are masked at random, and then the model trying to predict only those masked tokens. On NSP, two sentences (sequences of tokens) A and B are given to the model in order to decide whether the sentence B is the actual next sentence that follows A or not. The corpus used were BooksCorpus (800 million words) and English Wikipedia (2500 million words) for a vocabulary of $30k$ tokens. BERT achieved state-of-the-art on 11 Natural Language Process (NLP) tasks, General Language Understanding Evaluation (GLUE) task set consisting of 9 tasks, Stanford Question Answering Dataset (SQuAD) v1.1 and v2.0 and Situations With Adversarial Generations (SWAG). On October 25, 2019, Google Search announced that they applying BERT on search queries.

GPT-2 (Generative Pre-trained Transformer 2) model is the successor of the OpenAI GPT [25] model with modified initialization procedure, weight scaling and normalization layers. Additionally, the vocabulary expanded to 50257 tokens and the input sequence from 512 to 1024 tokens. GPT-2 is a left-to-right unidirectional model and instead of being a word-level or a byte-level LM, it makes use of Byte Pair Encoding (BPE) [31] which allows to combine the empirical benefits of word-level LM with the generality of byte-level approaches. The model's architecture is based on Transformers and four different variations of model's size were used with 117, 345, 762 and 1542 millions of parameters each and 12, 24, 36 and 48 layers, respectively. GPT-2 was trained on WebText created by OpenAI which consists of scraped web pages which have been curated/filtered by human and test on zero-shot tasks achieving state-of-the-art accuracy and perplexity on 7 out of 8.

ELMo (Embeddings from Language Models) is a feature-based approach that uses the representations of all hidden layers of a bidirectional LM that consists of two identical unidirectional LMs (a left-to-right and a right-to-left), to generate a contextualized word representation that encapsulate the information of each layer, including the output of the Softmax layer. In other words ELMo is a linear combination of the LM's representations. While training the weights of the LM are freezed and only the coefficients of the linear combination are updated. The bidirectional LM adopter for their experiments based on the architectures used in [12, 14] with modifications

to support joint training of both directions and the addition of a residual connection between LSTM layers. The final model provides three layers of representation for input token, while the size of the ELMo representation is 1024 ELMo representations evaluated and achieved state-of-the-art on six NLP tasks SQuAD, SNLI, SRL, Coref, NER and SST-5.

ULMFiT (Universal Language Model Fine-Tuning) is based on AWD-LSTM [18] LM and its contribution is on the training techniques that applies to enhance the performance of the LM. Initially, the model trained on Wikitext-103 [19] and then during fine-tuning they apply *discriminative fine-tuning* and *slanted triangular learning rates*. Instead of using the same learning rate for all layers of the model, *discriminative fine-tuning* allows to tune each layer with different learning rates. *Slanted triangular learning* modifies triangular learning rates [32] with a short increase and a long decay period. Additionally, *concat pooling* and *gradual unfreezing* where used. The *concat pooling* representation is a concatenation of three representations (i) the representation of the last token, (ii) the *maxpool* of the representations of the N previous tokens and (iii) the *meanpool* of the representations of the N previous tokens. *Gradual unfreezing* is a technique similar to *chain-thaw* [6] that gradually unfreezes the model's layers during training starting from the last layer. ULMFiT evaluated on six widely-studied datasets on three common text classification tasks sentiment analysis, question classification and topic classification, achieving state-of-the-art results.

Experiments

Corpus

In this study, we experiment on two corpuses the CMCC corpus and the PAN2018 corpus. The former applies on cross-topic and cross-genre scenarios, and the later on cross-fandom scenario.

CMCC

The CMCC corpus introduced in [9] and also used in previous cross-domain AA works [30, 36]. CMCC is a controlled corpus in terms of genre, topic and demographics of subjects. It includes samples by 21 undergraduate students as candidates authors (A), covering six genres (blog, email, essay, chat, discussion, and interview) and six topics (catholic church, gay marriage, privacy rights, legalization of marijuana, war in Iraq, gender discrimination) in English. To ensure that the same specific aspect of the topic is followed, a short question was given to subjects (e.g., Do you think the Catholic Church needs to change its ways to adapt to life in the 21th Century?). In two genres (discussion and interview) the samples were audio recordings and they have been transcribed into text as accurately as possible maintaining information about pauses, laughs etc. For each subject, there is exactly one sample for each combination of genre and topic. More details about the construction of this corpus are provided in [9].

PAN18

The PAN18 corpus created for Cross-Domain Authorship Attribution 2018 task of PAN 2018 [38]. The corpus consists of a development set with 10 problems (2 for each language[2]) and an evaluation set with 20 problems (4 for each language). Each problem has 7 documents for each candidate author (documents of known authorship), but the number of documents of unknown

[2]English, French, Italian, Polish and Spanish

authorship is different for each author. In this study we use the 4 problems on English language of the evaluation set, which differ only on the number of candidate authors (5, 10, 15 and 20). All documents of unknown authorship are fics of the same fandom (target-fandom) while the documents of known authorship by the candidate authors are fics of several fandoms (other than the target-fandom). The documents are selected texts from *Archive of Our Own*, a project of the Organization for Transformative Works[3]. The selected texts have at least 500 tokens and in case of having more than 1000 tokens were restricted to middle 1000 tokens. All texts were encoded as plain text (UTF8) and tokenization was done using NLTK's "WordPunctTokenizer" [2].

Experimental Setup

In this study, our focus is on cross-domain AA and more specifically on cross-topic, cross-genre and cross-fandom scenarios. Based on previous works for comparison reasons, we use CMCC corpus on cross-topic and cross-genre scenarios and PAN2018 on cross-fandom scenario. Following are the details of the setup on each scenario:

- *Cross-topic* We assume that the topic of training texts (K) is different from the topic of test texts (U) while all texts (both K and U) belong in the same genre. Similar to [36] and [30], we perform leave-one-topic-out cross-validation where all texts on a specific topic (within a certain genre) are included in the test corpus and all remaining texts on the remaining topics (in that genre) are included in the training corpus. This is repeated six times so that all available topics to serve exactly once as the test topic. Mean classification accuracy over all topics is reported.

- *Cross-genre* Similar to cross-topic, we perform leave-one-genre-out cross-validation as in [36], where all texts on a specific genre (within a certain topic) are included in the test corpus and all remaining texts on the remaining genres (in that topic) are included in the training corpus. The number of available genres is also six like topics, and though we repeat the leave-one-genre-out cross-validation six times and report the mean classification accuracy. In both scenarios, cross-topic and cross-genre, the candidates authors set A consists of 21 undergraduate students as mentioned in section .

[3]https://archiveofourown.org

- *Cross-fandom* We use the four problems in English from the evaluation set of PAN18 dataset, where the training and test document sets are predefined by the creator of the dataset. On each problem, the training texts (K) are on different fandoms than the test texts (U) fandom and the number of candidate authors is different (5, 10, 15, and 20). As on [38] where the same dataset were used, the F1 score on each problem and the macro-averaged F1 score over all four problems are reported.

All the examined models use a MHC on top of a language modeling method. First, we study the original Bagnall's approach where a character-level RNN is trained over K. Then, each one of the pre-trained language models described in previous section, with minor differences on the setup, in case of:

- *BERT* we used only the one sentence format, as presented on Section , to extract the representations and due to restriction of 512 tokens per input, we split each text to segments of 510 tokens maximum (e.g. a text with 1200 tokens splitted in three segments of 510, 510 and 180 tokens, respectively).

- *GPT-2*, we split each text to segments of 1024 tokens maximum (e.g. a text with 1200 tokens splitted in two segments of 1024 and 176 tokens respectively).

- *ELMo*, we adopted the implementation from Google[4], which has no limitation on the size of the input sequence.

- *ULMFiT*, we used the representation of *concat pooling* produced by the left-to-right AWD-LSTM. The size of the input sequence in ULMFiT is not restricted.

All of the pre-trained LMs was fine-tuned for the specific AA task with MHC as classifier without further training the language model, since our goal is to explore the potential of pre-trained models obtained from general domain corpora.

In MHC, each author corresponds to a separate classifier with N inputs and M outputs, where N is the dimensionality of text representation, Table 1, and M is equal to vocabulary size V. During training, each classification layer is trained only with the documents of the corresponding author. The vocabulary is defined as the most frequent tokens in the corpus. These are less likely to be affected by topic shifts and the reduced input size increases

[4]https://tfhub.dev/google/elmo/2

the efficiency of our approach. The selected values of V are 100, 500, $1k$, $2k$ and $5k$. Each model used its own tokenization stage except from ELMo (where ULMFiT's tokenization was used). Note that RNN is a character-level model while all pre-trained models are token-based.

Table 1: Dimensionality of representation (N) for each language model in this study.

Model	RNN	BERT	ELMo	GPT-2	ULMFiT
N	149	768	1024	768	400

Since RNN is trained from scratch for a corpus of small size, it is considerably affected by initialization. As a result, there is significant variance when it is applied several times to the same corpus. To compensate this, we report average performance results for 10 repetitions. Regarding the training phase of each method, we use 100 epochs for RNN and examine four cases for the pre-trained models: the minimal training of 1 epoch and the cases of 5, 10 and 20 epochs of training. As concerns the normalization phase three different approaches were examined (1) without normalization ($C = \varnothing$), (2) using the (unlabeled) training texts as normalization corpus ($C = K$) and (3) using the test texts as normalization corpus ($C = U$), as noted in Section in all cases the texts are unlabeled.

Results on Cross-topic AA

Table 2 presents the leave-one-topic-out cross-validation accuracy results for each one of the six available genres as well as the average performance over all genres for each method. Two cases of normalization corpus are presented: one using the (unlabeled) training texts as normalization corpus ($C = K$) and another where the (unlabeled) test texts are used as normalization corpus ($C = U$). The former means that C includes documents with distinct topics with respect to the document of unknown authorship while the latter ensures that there is perfect thematic similarity. As can be seen, the use of a suitable normalization corpus is crucial to enhance the performance of the examined methods. The third case where no normalization is applied ($C = \varnothing$) is not worth mentioning on this scenario.

As concerns individual pre-trained language models, BERT and ELMo are better able to surpass the RNN baseline while ULMFiT and GPT-2 are not that competitive. In addition, BERT and ELMo methods need small number of training epochs while ULMFiT and GPT-2 improve with increased number of epochs.

Table 2: Accuracy results (%) on Cross-Topic AA. The reported performance of baseline models is taken from the corresponding publications.

‡	LM	V	epochs	Blog	Email	Essay	Chat	Disc.	Interv.	Avg.
$C = K$	RNN	-	100	47.94	44.37	41.00	73.41	75.71	72.54	59.16
	BERT	$2k$	1	57.14	49.21	60.32	84.92	79.37	80.16	68.52
	BERT	$5k$	1	53.97	52.38	58.73	86.51	77.78	78.57	67.99
	ELMo	$2k$	1	56.35	55.56	56.35	80.95	72.22	76.98	66.40
	ELMo	$5k$	1	55.56	53.17	57.14	82.54	70.64	76.19	65.87
	GPT-2	$2k$	20	60.32	57.94	54.76	76.98	63.49	79.37	65.48
	GPT-2	$5k$	20	58.73	59.52	61.11	84.13	63.49	76.98	67.33
	ULMFiT	$2k$	10	50.00	43.65	52.38	79.37	72.22	71.43	61.51
	ULMFiT	$5k$	20	46.83	40.48	50.79	80.16	69.84	70.64	59.79
$C = U$	RNN	-	100	61.67	56.43	68.36	81.27	**86.90**	84.52	73.19
	BERT	$2k$	5	72.22	64.29	76.98	90.48	84.13	90.48	79.76
	BERT	$5k$	5	**73.81**	61.11	**77.78**	**92.86**	84.13	90.48	**80.03**
	ELMo	$2k$	10	72.22	65.08	75.40	89.68	76.19	**91.27**	78.31
	ELMo	$5k$	10	72.22	**67.46**	76.98	88.10	76.98	89.68	78.57
	GPT-2	$2k$	20	72.22	64.29	73.02	80.16	67.46	82.54	73.28
	GPT-2	$5k$	20	69.84	65.87	69.84	84.13	73.81	85.71	74.87
	ULMFiT	$1k$	10	64.29	57.94	73.02	87.30	80.16	88.89	75.26
	ULMFiT	$2k$	10	64.29	54.76	73.81	88.89	78.57	88.10	74.74
	ULMFiT	$5k$	20	58.73	54.76	75.40	88.89	75.40	84.13	72.88
	C3G-SVM [30]	-	-	33.41	36.53	36.66	57.46	49.91	56.35	45.05
	PPM5 [36]	-	-	52.38	39.68	50.00	57.94	36.51	47.62	47.35
	DV-MA [36]	-	-	43.65	65.87	60.32	71.43	80.16	67.46	64.81

Table 2 also shows the corresponding results from previous studies on cross-topic AA using exactly the same experimental setup. These baselines are based on character 3-grams features and a SVM classifier (C3G-SVM) [30], a compression-based method (PPM5) [36], and a method using text distortion to mask thematic information (DV-MA) [36]. As can be seen, when $C = U$ all of the examined methods surpass the best baseline in average performance and the improvement is high in all genres. It is remarkable that all models except ULMFiT achieve to surpass the baselines (in average performance) even when $C = K$.

Figure 2 presents the mean classification accuracy with respect to vocabulary size on cross-topic AA. Each sub-figure correspond to different LM. The type of the line indicates the normalization corpus, a dashed line indicates the use of training texts ($C = K$) as normalization corpus, while a continuous line indicates the use of test texts ($C = U$), as noted in section in both cases the texts are unlabeled. The shape of each point correspond to epochs of training, 1, 5, 10 and 20 for circle, triangle, square and x-mark respectively.

From the aspect of vocabulary size, in contradiction to the state of the art [36], where the best results achieved for vocabularies that consisted of less than $1k$ words (most frequent), in our set up the most appropriate value seems to be above $2k$. Despite the gap between $2k$ and $5k$ words in vocabulary size BERT and ELMo have minor difference in accuracy indicating that above $2k$ words the affect of vocabulary size is minor. GPT-2 continues to increment the accuracy and ULMFiT started to decrement for values above $1k$ words, Table 2. Experiments with values over $5k$ were prohibitive due to runtime

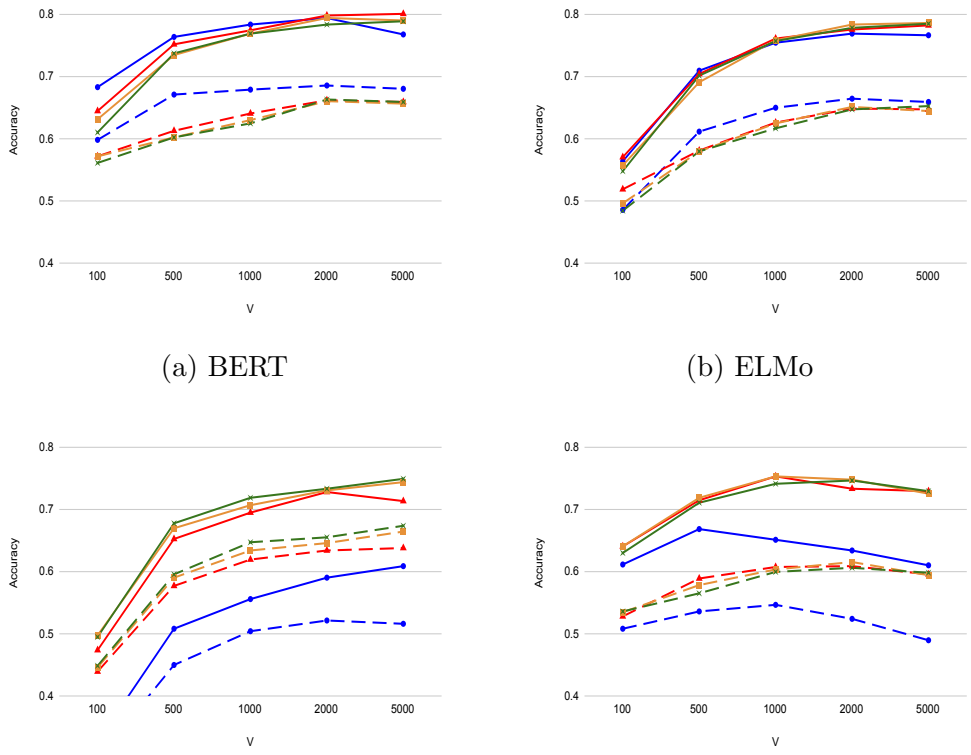
of training, with $5k$ words the runtime was approximately 4 days for each model running on GPU.

From the aspect of training epochs, BERT and ELMo achieved their best performance in $C = K$ cace with minimal training. In $C = U$ case their performance is slightly affected by the number of training epochs. This behavior raises the question of over-fitting. As mentioned in section the selection criterion Equation 2, is based on the cross-entropy of each text. MHC is trained on predicting the text flow and thus the cross-entropy decreases after each epoch of training. Having in mind the cross-entropy, if we have a second look on Figure 2, the case of over-fitting is rejected since the behavior of accuracy in relevance with the number of training epochs (indicated by the shape of point) do not have the characteristics of over-fitting (increment of training epochs decrements the accuracy).

Results on Cross-Genre AA

The experiments on cross-genre performed on the same set up as in cross-topic. Table 3 presents the accuracy results on leave-one-genre-out cross-validation for each one of the six available topics and the average performance over all topics, similar to Table 2. Based on the results of Section the most reasonable value of V in order to check the performance of each method is $V = 2k$. The case of $V = 5k$ is very time consuming without offering valuable gain and below $1k$ the performance is not remarkable. For the experiments on cross-genre the values of $1k$ and $2k$ were selected for V. Comparing the two cases, the results with $V = 2k$ surpass in all experiments the results with $V = 1k$ and thus we selected to present only the case of $V = 2k$ on Table 3.

BERT and ELMo achieved high results as expected from their performance on cross-topic, with ELMo achieving the highest accuracy result. Unexpectedly, ULMFiT which had the worst performance in cross-topic achieved the second best performance. GPT-2 performed lower than RNN baseline in both cases of $C = K$ and $C = U$. Comparing Table 2 and Table 3 is noticeable that ELMo and BERT are more stable in performance than GPT-2 and ULMFiT. The main diference between the former and the latter is the directionality, the former two are bidirectional while the latter are unidirectiocal, we suspect that this is the main reason that affects the stability in performance.



(a) BERT

(b) ELMo

(c) GPT-2

(d) ULMFiT

Figure 2: Accuracy results for vocabulary sizes (V) 100, 500, $1k$, $2k$ and $5k$ for each pre-trained model. Colored symbols blue circle, red triangle, yellow square and green x-mark correspond to 1, 5, 10 and 20 training epochs, respectively. The type of the line indicates the normalization corpus, a dashed line indicates the use of training texts ($C = K$) as normalization corpus, while a continuous line indicates the use of test texts ($C = U$).

Table 3: Accuracy results (%) on cross-genre AA for vocabulary size $2k$ ($V = 2k$) and each topic (Church (C), Gay Marriage (G), War in Iraq (I), Legalization of Marijuana (M), Privacy Rights (P), Gender Discrimination (S)). The reported performance of the baseline models (only available in average across all topics) is taken from the corresponding publications.

‡	LM	epochs	C	G	I	M	P	S	Avg.
$C = K$	RNN	100	58.89	68.33	71.59	60.24	50.40	62.22	61.94
	BERT	10	70.63	77.78	83.33	73.81	62.70	76.98	74.21
	ELMo	10	68.25	78.57	78.57	71.43	55.56	65.08	69.58
	GPT-2	20	52.38	67.46	61.11	57.94	50.79	53.17	57.14
	ULMFiT	20	72.22	77.78	79.37	70.63	61.11	68.25	71.56
$C = U$	RNN	100	75.32	75.95	86.11	79.52	69.37	74.21	76.75
	BERT	5	84.13	87.30	88.10	82.54	**77.78**	78.57	83.07
	ELMo	20	87.30	88.89	**88.89**	**83.33**	76.98	**81.75**	**84.52**
	GPT-2	20	69.84	76.98	74.60	67.46	61.11	72.22	70.37
	ULMFiT	10	**88.10**	**89.68**	85.71	82.54	**77.78**	79.37	83.86
	C3G-SVM [30]	-							
	PPM5 [36]	-							60.00
	DV-MA [36]	-							33.00

Results on Cross-Fandom AA

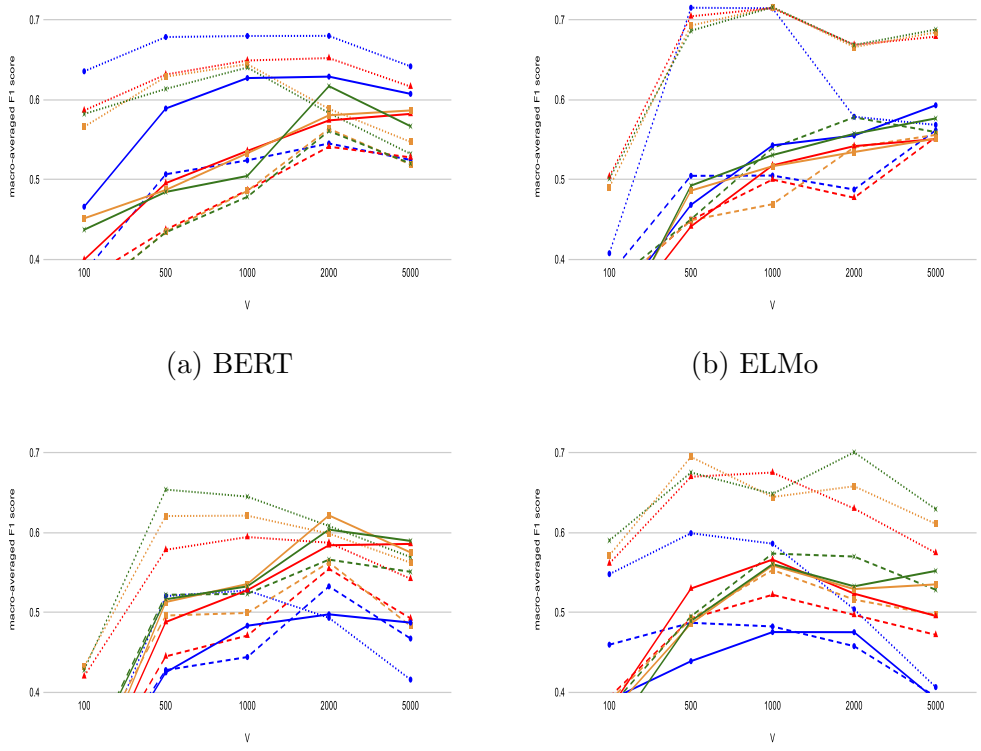
Figure 3 presents the macro-averaged F1 scores with respect to vocabulary size on PAN18 dataset. Each sub-figure correspond to different LM. The type of the line indicates the normalization corpus, a dashed line indicates the use of training texts ($C = K$) as normalization corpus, a continuous line indicates the use of test texts ($C = U$) and a dotted line indicated that no normalization applied ($C = \varnothing$), as noted in section in all cases the texts are unlabeled. The shape of each point correspond to epochs of training, 1, 5, 10 and 20 for circle, triangle, square and x-mark respectively.

Normalization corpus has significant impact on how the vocabulary size affects the results. When no normalization is applied the optimal vocabulary size for all methods in general seems to be somewhere between 500 and $1k$ tokens. If normalization is applied (training set or test set) the optimal vocabulary size transfers around $2k$ tokens. Vocabularies with 100 tokens expected to have better results, since generally the most frequent words are more valuable in AA [33, 36].

In each sub-figure of Figure 3, if we isolate the results according to the normalization set, we may notice that the layout according to epochs of training remains the same. Moreover the layout in almost all cases is stable to vocabulary size. This observations indicates that the optimal number of training epochs is characteristic of the LM. BERT and ELMo perform better with few training epochs, in contradiction to GPT-2 an ULMFiT.

The performance of BERT with minimal training raises the question of

over-fitting. As mentioned in section the selection criterion Equation 2, is based on the cross-entropy of each text. MHC is trained on predicting the text flow and thus the cross-entropy decreases after each epoch of training. Having in mind the cross-entropy, if we have a second look on Figure 3, the case of over-fitting is rejected since the behavior of scores in relevance with the number of training epochs (indicated by the shape of point) do not have the characteristics of over-fitting (increment of training epochs decrements the score).



(a) BERT

(b) ELMo

(c) GPT-2

(d) ULMFiT

Figure 3: Macro-averaged F1 scores for vocabulary sizes (V) 100, 500, $1k$, $2k$ and $5k$ for each pre-trained model. Colored symbols blue circle, red triangle, yellow square and green x-mark correspond to 1, 5, 10 and 20 training epochs, respectively. The type of the line indicates the normalization corpus, a dashed line indicates the use of training texts ($C = K$) as normalization corpus, a continuous line indicates the use of test texts ($C = U$) and a dotted line indicates that no normalization applied ($C = \varnothing$).

Table 4 presents the best runs according to macro-averaged F1 score over

Table 4: Macro-averaged F1 scores on PAN18. The reported performance of baseline models is taken from the corresponding publications. In each column (problem) underline indicates the best performance on our experiments while bold indicates the best performance among all.

‡	LM	V	epochs	Prob.1	Prob.2	Prob.3	Prob.4	Macro-F1
$C = \varnothing$	RNN	-	100	0.433	0.537	0.497	0.526	0.499
	BERT	$2k$	1	0.598	0.644	0.721	0.758	0.680
	GPT-2	500	20	0.490	0.612	**0.844**	0.670	0.654
	ELMo	$1k$	10	0.617	**0.723**	0.767	0.758	0.716
	ULMFiT	$2k$	20	0.609	0.652	0.747	0.795	0.701
$C = K$	RNN	-	100	0.439	0.513	0.639	0.820	0.603
	BERT	$2k$	10	0.444	0.453	0.563	0.795	0.564
	GPT-2	$2k$	20	0.435	0.490	0.540	0.800	0.566
	ELMo	$2k$	20	0.498	0.505	0.509	0.800	0.578
	ULMFiT	$1k$	20	0.427	0.507	0.562	0.800	0.574
$C = U$	RNN	-	100	0.369	0.362	0.535	0.635	0.475
	BERT	$2k$	1	0.522	0.597	0.597	0.800	0.629
	GPT-2	$2k$	10	0.551	0.582	0.598	0.756	0.622
	ELMo	$5k$	1	0.516	0.525	0.591	0.741	0.593
	ULMFiT	$1k$	5	0.492	0.529	0.712	0.532	0.566
	BASELINE [38]	-	-	-	-	-	-	0.697
	Murauer et al. [20]	-	-	**0.730**	0.689	0.800	**0.830**	**0.762**

all four problems, of each method in all three normalization cases: (1) without applying any normalization corpus ($C = \varnothing$), (2) using the (unlabeled) training texts as normalization corpus ($C = K$) and (3) using the (unlabeled) test texts as normalization corpus ($C = U$).

In general all methods performed better if no normalization is used ($C = \varnothing$), exept from RNN. ELMo and ULMFiT surpasses the baseline, with ELMo achieving higher score but still being behind Marauer et al. [20], the winner of Cross-Domain Authorship Attribution 2018 task of PAN 2018 [38].

The F1 scores of for each one of the four available problems as well as the macro-averaged F1 score over all problems for each method. All the three cases of normalization are presented:

Exploring Normalization Corpus

In cross-domain AA, the documents in training set K are on different domain (e.g. topic, genre, fandom) than the documents in test set U. To limit the effect that differentiation of domain has on MCH, we use a vocabulary created by the most frequently appeared tokens which have a greater relation to the author's style than to the domain [38]. A side effect of training a classifier using the most frequent appeared tokens is the bias that develops to the tokens with respect to the their frequency. The role of normalization corpus C is to overcome that side effect by reducing the bias with normalization

vector n, Equation 1.

The results of the experiments with different variations of normalization corpuses, indicates the significance of the normalization corpus and its relation to the characteristics of each dataset. More specifically, comparing the scores between $C = K$ (dashed lines) and $C = U$ (continuous lines) cases on Figure 2, confirms the significance of normalization corpus. In all of four LMs, regardless the values of the other parameters (vocabulary size and epochs of training) the use of $C = K$ results in significant improvement of accuracy. Figure 3 on the other hand reflects the significance of normalization corpus with a different manner. The absence of normalization (dotted lines) results in significant better results, while the cases of $C = K$ and $C = U$ have similar results without being clear which of those two normalization corpuses is better in general. In Figure 2, the results without the use of normalization corpus where omitted since the best accuracy achieved is less than 0.5.

In order to explore the relation between the frequency of tokens and normalization corpus, we defined the set M, Equation 3, as set of the average difference in frequency of appearance in common tokens of known and unknown document sets for each candidate author $a \in A$.

$$M = \left\{ \frac{1}{|T_a|} \sum_{token \in T_a} |F_{K_a}(token) - F_{U_a}(token)| \middle| \forall a \in A \right\} \tag{3}$$

where T_a is the set of common tokens in K_a and U_a sets of documents, where K_a and U_a are author's a sets of known and unknown documents, respectively. The function F is defined by

$$F_D(t) = \text{the frequency of appearance of token } t \text{ in document set } D \tag{4}$$

We create an M set for each problem on each scenario (cross-topic, cross-genre and cross-fandom) and for each tokenization method applied in our experiments. There are 36 cross-topic problems in total, 36 cross-genre problems and 4 cross-fandom problems, 76 problems in total. Three tokenization methods, each LM make use of its own tokenization method except from ELMo, where ULMFiT's tokenization method where used. We name each tokenization method by the name of the LM used by. Concluding in $3 \times 76 = 228M$ sets in total. Then we merged those sets according to their scenario and tokenization method into nine sets as Table 5 indicates.

Table 5 presents some descriptive measures of the nine sets created by merging the $228M$ sets according to their scenario and tokenization method. As mentioned above the values $m \in M$ are indicating the mean difference in frequency of appearance of the common tokens between known and unknown

Table 5: Descriptive measures of the m values for each scenario

Corpus	Scenario	Tokenization	Min	Max	Mean ± Std
CMCC	Cross-topic	BERT	0.686	13.918	2.944 ± 1.440
		GPT-2	0.673	12.611	2.625 ± 1.334
		ULMFiT/ELMo	0.779	13.751	3.072 ± 1.490
	Cross-genre	BERT	0.980	14.179	3.401 ± 1.413
		GPT-2	1.151	13.563	3.667 ± 1.541
		ULMFiT/ELMo	1.233	14.001	3.574 ± 1.494
PAN18	Cross-fandom	BERT	0.297	2.180	1.036 ± 0.602
		GPT-2	0.296	1.886	0.942 ± 0.513
		ULMFiT/ELMo	0.324	2.315	1.145 ± 0.625

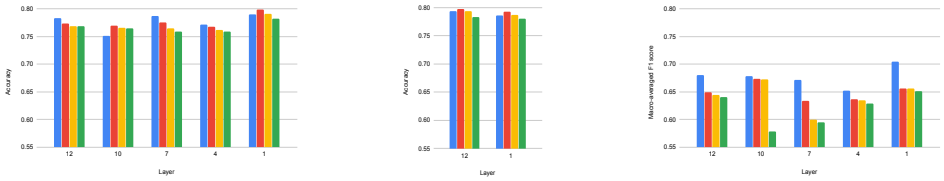
documents of an author. In other words, an m value is a similarity index of know documents (training set) and unknown documents (test set) of an author. The lower the m value the higher the similarity.

Comparing the values between CMCC and PAN18 its is obvious that the similarity of each author's known and unknown documents is significantly higher on PAN18 problems that it is on CMCC problems. That difference in similarity affects the impact of the normalization corpus. Comparing Figure 2 and Figure 3 on the difference that the results have between the use of K as normalization corpus C and the use of U as C, on Figure 2 the gap is clear, but on Figure 3 there is no gap. Moreover, the absence of normalization corpus, Figure 3, achieved the higher scores allowing us to conjecture that in case of high similarity the use of K or U as normalization corpus is like increasing a bit the noise level.

Shallower Layers

Two out of the four LM we examine of this study (BERT and GPT-2) are deep networks with twelve layer each, with each layer capable of producing a representation for each token given as input. On Sections , and the results obtained by the use of the last layer's representation (layer 12), on this Section, we are examining the representations from other layers by experimenting on layers 1, 4, 7, 10 and 12 of BERT's LM. Despite the fact that GPT-2 is also a deep network and we could run similar experiments on its layers too, we selected only BERT, since BERT surpassed GPT-2 on all scenarios (cross-topic, cross-genre and cross-fandom).

Figure 4a shows the accuracy results of various BERT's layers representations on the cross-topic scenario, for vocabulary size of $V = 1k$. Layers



(a) Cross-topic $V = 1k$ and $C = U$

(b) Cross-topic $V = 2k$ and $C = U$

(c) Cross-fandom $V = 1k$ and $C = \varnothing$

Figure 4: Accuracy results on (a) and (b), and macro-averaged F1 scores on (c) for representations from different layers of BERT LM. The color of each bar indicated the number of training epochs blue, red, yellow and green for 1, 5, 10 and 20 epochs, respectively.

12, 10, 7, and 4 are quite close, while layer 1 seems to performed better. In absolute values, the difference in accuracy results is about 0.02 (2%). Examining further the response of layer 1, we test it on the same scenario for vocabulary size of $V = 2k$. As can be seen on Figure 4b, the results are not those that expected, layer 1 performed lower that the last layer.

Figure 4c similar to Figure 4a shows the macro-averaged F1 scores of various BERT's layers representations on the cross-fandom scenario, for vocabulary size of $V = 1k$. On this scenario the we can not say clearly which layer performed better, layer 1 has the best performance when the training epochs are one or 20, while layer 10 has the best performance when the training epochs are 5 of 10.

Unfortunately, the results of our experiments on BERT's representations from layers 1, 4, 7, 10 and 12 are not allowing us to draw any safe conclusion. Further experimentation is needed in order to clarify the potentials or the kind of the information that shallower layers capture. Focusing on those questions is out of the scope of this study, but the results we already obtained are indicating that something is hiding on the first layers.

Conclusions

In this paper, we explore the usefulness of pre-trained language models in cross-domain AA. Based on Bagnall's model [1], originally proposed for authorship verification, we compare the performance when we use either the original character-level RNN trained from scratch in the small-size AA corpus or pre-trained token-based language models obtained from general-domain corpora. We demonstrate that BERT and ELMo pre-trained models achieve the best results in cross-topic and cross-genre scenarios, while being the most stable approaches with respect to the results in both scenarios. In cross-fandom scenario, unexpectedly ELMo and ULMFiT achieve the best results, while BERT do not manage to surpass the baseline.

To explore further the unexpected results on cross-fandom, we present a comparison between PAN18 corpus used in cross-fandom scenario and CMCC corpus used in cross-topic and cross-genre scenarios. The results of the comparison reveal a connection between the properties of the corpus and the normalization corpus. Moreover, experiments with shallower layers of BERT to identify which layer captures the most useful information for an AA task, show that despite the fact that there maybe remarkable differences on the results on specific cases, in general it is not clear if there is a layer which is better than the others.

A crucial factor to enhance performance is the normalization corpus used in the MHC. In cross-domain AA, it is very important for the normalization corpus to have exactly the same properties with the documents of unknown authorship. In our experiments, using a controlled corpus, it is possible to ensure a perfect match in all scenarios. In practice, this is not always feasible. A future work direction is to explore how one can build an appropriate normalization corpus for a given document of unknown authorship. Other interesting extensions of this work is to study the effect of extending fine-tuning to language model layers.

Bibliography

[1] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum*, 2015.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[3] José Eleandro Custódio and Ivandré Paraboni. Each-usp ensemble cross-domain authorship attribution. *Working Notes Papers of the CLEF*, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[5] S.H.H. Ding, B.C.M. Fung, F. Iqbal, and W.K. Cheung. Learning stylometric representations for authorship analysis. *IEEE Transactions on Cybernetics*, 49(1):107–121, 2019.

[6] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, 2017.

[7] Olga Fourkioti, Symeon Symeonidis, and Avi Arampatzis. Language models and fusion for authorship attribution. *Information Processing & Management*, 56(6), 2019.

[8] Zhenhao Ge, Yufang Sun, and Mark J. T. Smith. Authorship attribution using a neural network language model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 4212–4213. AAAI Press, 2016.

[9] Jade Goldstein-Stewart, Ransom Winder, and Roberta Evans Sabin. Person identification from text and speech genre samples. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344. Association for Computational Linguistics, 2009.

[10] Oren Halvani and Lukas Graner. Cross-domain authorship attribution based on compression. *Working Notes of CLEF*, 2018.

[11] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[12] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[13] M. Kestemont, J. Stover, M. Koppel, F. Karsdorp, and W. Daelemans. Authenticating the writings of Julius Caesar. *Expert Systems with Applications*, 63:86–96, 2016.

[14] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[15] Mirco Kocher and Jacques Savoy. Distributed language representation for authorship attribution. *Digital Scholarship in the Humanities*, 33(2):425–441, 2018.

[16] David Madigan, Alexander Genkin, David D Lewis, Shlomo Argamon, Dmitriy Fradkin, and Li Ye. Author identification on the large scale. In *Proceedings of the Meeting of the Classification Society of North America*, 2005.

[17] Rohith Menon and Yejin Choi. Domain independent authorship attribution without domain adaptation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 309–315, 2011.

[18] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[19] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[20] Benjamin Murauer, Michael Tschuggnall, and Günther Specht. Dynamic parameter search for cross-domain authorship attribution. *Working Notes of CLEF 2018*, 2018.

[21] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard. Surveying stylometry techniques and applications. *ACM Computing Surveys*, 50(6), 2018.

[22] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[23] Martin Potthast, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. A decade of shared tasks in digital text forensics at pan. In *European Conference on Information Retrieval*, pages 291–300. Springer, 2019.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[26] Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, 2017.

[27] Paolo Rosso, Francisco Rangel, Martin Potthast, Efstathios Stamatatos, Michael Tschuggnall, and Benno Stein. Overview of pan'16. In Norbert Fuhr, Paulo Quaresma, Teresa Gonçalves, Birger Larsen, Krisztian Balog, Craig Macdonald, Linda Cappellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 332–350. Springer International Publishing, 2016.

[28] Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, 2015.

[29] Upendra Sapkota, Thamar Solorio, Manuel Montes, and Steven Bethard. Domain adaptation for authorship attribution: Improved structural correspondence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2226–2235, 2016.

[30] Upendra Sapkota, Thamar Solorio, Manuel Montes, Steven Bethard, and Paolo Rosso. Cross-topic authorship attribution: Will out-of-topic data help? In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1228–1237, 2014.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[33] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

[34] Efstathios Stamatatos. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21:421–439, 2013.

[35] Efstathios Stamatatos. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the As-*

sociation for Computational Linguistics: Volume 1, Long Papers, pages 1138–1149, 2017.

[36] Efstathios Stamatatos. Masking topic-related information to enhance authorship attribution. *Journal of the Association for Information Science and Technology*, 69(3):461–473, 2018.

[37] Efstathios Stamatatos, Martin Potthast, Francisco Rangel, Paolo Rosso, and Benno Stein. Overview of the pan/clef 2015 evaluation lab. In Josanne Mothe, Jacques Savoy, Jaap Kamps, Karen Pinel-Sauvagnat, Gareth Jones, Eric San Juan, Linda Capellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 518–538. Springer International Publishing, 2015.

[38] Efstathios Stamatatos, Francisco Rangel, Michael Tschuggnall, Benno Stein, Mike Kestemont, Paolo Rosso, and Martin Potthast. Overview of pan 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 267–285. Springer, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Algorithms

Algorithm 1: Pseudocode of LM

```
Function LM(document)
    tokens = Tokenizer(document)
    representations = PreTrainedModel(tokens)
    return tokens, representations
end
```

Algorithm 2: Pseudocode of Filter

```
Function Filter(vocabulary, in_tokens, in_representations)
    initialize out_tokens and out_representations as empty lists
    n_tokens = number of tokens in in_tokens list
    for i = 0 to n_tokens do
        if in_tokens[i] ∈ vocabulary then
            Append in_tokens[i] to out_tokens list
            Append in_representations[i − 1] to out_representations
             list
        end
    end
    return out_tokens, out_representations
end
```

Algorithm 3: Pseudocode of MHC

```
Function MHC(author, input, label)
    /* Classifiers is a list of classifiers (each
       classifier is a linear fully-connected layer), one
       classifier for each author.                          */
    output = Classifiers[author](input)
    cross_entropy_error = CrossEntropy(output, label)
    return cross_entropy_error
end
```

Algorithm 4: Pseudocode of training the proposed method

```
K = a list of tuples (a, d) where a is the author of document d
LM = a pre-trained language model
V = a vocabulary
Classifier a list of classifiers (a linear fully-connected layer), one
 classifier for each author

Method Train
    foreach (author, document) ∈ K do
        tokens, representations = LM(document)
        tokens, representations =
         Filter(V, tokens, representations)
        foreach (token, representation) ∈ (tokens, representations)
         do
            output = Classifier[author](representation)
            cross_entropy_error = CrossEntropy(output, token)
            back-propagate cross_entropy_error and update weights
        end
    end
end
```

Algorithm 5: Pseudocode of predicting authors

```
document = the input document
A = the candidate authors
LM = a pre-trained language model
V = a vocabulary
Classifier a list of classifiers (a linear fully-connected layer), one
 classifier for each author
Method Predict Authors
    tokens, representations = LM(document)
    tokens, representations = Filter(V, tokens, representations)
    initialize score list with zeros
    n_tokens = number of tokens in tokens list
    foreach author ∈ A do
        foreach (token, representation) ∈ (tokens, representations)
         do
            output = Classifier[author](representation)
            cross_entropy_error = CrossEntropy(output, token)
            sums_errors[author] += cross_entropy_error
        end
        score[author] /= n_tokens
        foreach author ∈ A do
            score[author] −= normalization_vector[author]
        end
    end
    predicted_author = arg min(score)
end
```

Algorithm 6: Pseudocode of calculating normalization vectors

```
A = list of authors
C = list of documents (normalization corpus)
LM = a pre-trained language model
V = a vocabulary
Classifier a list of classifiers (a linear fully-connected layer), one
 classifier for each author
Method Calculate Normalization Vector
    initialize normalization_vector with zeros
    foreach document ∈ C do
        tokens, representations = LM(document)
        tokens, representations =
         Filter(V, tokens, representations)
        initialize sums_errors list with zeros
        n_tokens = number of tokens in tokens list
        foreach author ∈ A do
            foreach
             (token, representation) ∈ (tokens, representations) do
                output = Classifier[author](representation)
                cross_entropy_error = CrossEntropy(output, token)
                sums_errors[author] += cross_entropy_error
            end
            normalization_vector[author] /= n_tokens
        end
    end
    nc = number of documents in C
    foreach author ∈ A do
        normalization_vector[author] /= nc
    end
end
```