Authorship Attribution: Identifying the authors of Yelp Reviews

Gareth Dwyer
University of Groningen
Groningen
The Netherlands
gareth@dwyer.co.za

Long Hoang Nguyen
Czech Technical University
Prague
Czechia
nguyeho7@fel.cvut.cz

Abstract

We present our experiments relating to authorship attribution on the Yelp dataset, attempting both identification and verification tasks. We show that generative character-based neural network language models can be used for authorship identification, and that in spite of recent advancements in neural network approaches, Support Vector Machines are still highly competitive in terms of accuracy, while being far more computationally efficient than many neural approaches. We achieve an accuracy of 0.90 for a 50-author identification problem. For a binary verification task, we achieve an accuracy score of 0.95.

1 Introduction

Identifying an author by the style of their writing is a problem that has fascinated researchers for centuries. It has practical applications in detecting forgeries (did Shakespeare really write all of the plays that we associate with him?), forensic linguistics (was this suicide note really written by the deceased, or was it by his murderer?) and more recently in detecting "fake" news and reviews.

Using a large dataset of Yelp reviews, we investigate how well a classifier can automatically discriminate between different authors. Being able to reliably identify an author by their writing style alone makes it easier to weed out fake reviews. For example, if an author leaves fake positive reviews for a specific business, he or she might do so from many different user accounts, each with its own unique username. If these reveiws share a similar writing style, we can use this to identify that these reviews are not authentic. On the other hand, different authors may share the same account (for example, a married couple). If the reviews were used, for example, to personalize adverts that the reviewer left, it would be interesting to take into account that adverts should be personalized to two people instead of only one.

Using the methods that we present here, a system could be built to identify these cases of one author having many accounts or many authors sharing a single account. More ambitiously, the methods could be used to identify a specific user who used other internet services, even if they used these other services under a different username.

In the Authorship Attribution (AA) literature, there are two main substasks. These are Authorship Identification, in which we attempt to predict which of a closed set of N candidate authors wrote a specific text, and Authorship Verification, in which we attempt to predict whether or not two different texts are written by the same author, independently of his or her identity. The first can be modelled as a multi-class classification problem, where each class is represented by a candidate author. The second can be modelled as a binary classification problem, where the two possible classes are *same-author* (meaning that the texts were written by the same author) and *different-author* (meaning that they were not).

Here we present experiments on both Authorship Identification and Authorship Verification tasks. Our contributions are:

- A simple method which uses TF-IDF vectors and Support Vector Machine classifiers for the Authorship Identification task
- Evidence that character-based neural language modelling can be used for the Authorship Identification task
- A novel algorithm for the Authorship Verification task.

2 Background

Although Authorship Identification is the better-known task and has been the focus of more research, Authorship Verification is arguably the more interesting problem, both practically and theoretically. For example, Luyckx and Daelemans (2008) explain how a majority of prior work has used datasets consisting of very few authors (often only two), and with a lot of data per author (often 10 000 words). In real-world tasks, we often have much less data per author and have a much larger set of candidate authors. Therefore, the Authorship Verification task, which assumes many, possibly infinite, candidate authors, and which usually assumes only a single short text per author, has more practical applications. Koppel has also made this argument (Koppel et al., 2011), and has gone as far as to state that the verification task is the "fundamental" AA task, and that solving it would "serve as a building block for solving almost any conceivable authorship attribution problem" (Koppel et al., 2012).

Some successful approaches to Authorship Verification have modelled the verification problem as an Authorship Identification problem. For example, Koppel and Winter (2014) invented the "Imposter's" method, in which the authors add more texts to the verification problem and then attempt to decide if the unknown text is more likely to be written by the author of the known text than by any of the "imposter" authors. A variation of this algorithm, which also relied on external texts, was the most successful at the PAN 2014 Authorship Verification shared task (Stamatatos et al., 2014). Similarly, Bagnall (2015) achieved the best results in the PAN 2015 Authorship Verification shared task by paritally modelling it as an identification problem.

Because the verification task is more fundamental than the identification task, it is desireable to model it directly, instead of through the identification problem. Therefore, we present a novel approach, described later, which treats the verification task as a binary classification problem and does not rely on external texts. This is not only a more intuitively pleasing way of solving the task, but it is also more efficient (we look only at the relation between the two presented texts), and easier to implement (we do not need to find related texts to use as imposters). Further, our method does not require the use of a manual threshold value to decide when two texts are "similar enough" to be given a *same author* prediction. Such a threshold is often needed in the cases discussed above, and is usually arbitrarily assigned.

An important part of supervised learning tasks is feature extraction. Prior work in AA (for both identification and verification tasks) has shown that it is possible to fingerprint an author by using a number of different features, including shallow lexical features (for example, word and character n-grams), structural features (for example, sentence length) and deeper syntactic features (for example, part-of-speech tags). Features such as vocabulary richness have also been used (Koppel et al., 2009; Stamatatos, 2009). Although it is sometimes worthwhile to put extensive effort into manually crafting task-specific features for text classification tasks, often this only ends up replicating what a supervised classifier is able to learn on its own. The models we describe below make use only of simple lexical features. This is beneficial as they can be easily reused on other tasks, even across languages, without needing to find domain-specific tools such as taggers.

More recent work has shown that neural networks can be effective at authorship attribution tasks. For example, neural network based approaches have achieved the highest results in the PAN shared tasks relating to authorship attribution for the last two years running (Stamatatos et al., 2015; Rosso et al., 2016). However, neural networks have so far failed to become well established in AA tasks (in spite of their centrepiece role in most other NLP tasks), and are still being explored to reach their full potential in this context. Our research therefore includes experiments which use neural network models, similar to those used by the aforementioned winner of the two previous PAN shared tasks (Bagnall, 2015, 2016).

3 Method

We experimented with various tasks related to Authorship Identification and Authorship Verification. In this section, we describe how we used the Yelp dataset for each task, and what features and classification methods we used.

We present three models showing different approaches to Authorship Identification and one model for Authorship Verification.

3.1 Authorship Identification

First we experiment with Support Vector Machines (SVMs), using Term Frequency–Inverse Document Frequency (TF-IDF) vectors as features. Second, we use a Recurrent Neural Network with Gated Recurrent Units (GRU-RNN), using word embeddings as features. Third, we experiment with a more unusual approach in which we train a different generative character-based language model for each author, and predict unseen texts by finding the author-specific language model with the lowest cross entropy score on that text.

The first model is fairly generic and can be used for many different text classification tasks. It uses well-established text classification techniques that are highly computationally efficient. The third model uses techniques that are less common for text classification, and it is also less computationally efficient, but due to the large amount of research currently happening around neural networks, we believe it has more potential for future improvement.

3.1.1 Model 1: SVMs with TF-IDF

A Support Vector Machine is a linear classifier. It tries to find a separating hyperplane that maximizes the distance between two classes. In case of unseparable data, SVMs employ a slack variable allowing misclassifications after paying a penalty. Support Vector Machines have been widely used over the last two decades since Joachims (1998) showed that they provide a robust and well-performing method for many text classification tasks. They have proven to work well in tasks ranging from Named Entity Recognition (Kravalová and Žabokrtský, 2009), Authorship Profiling, such as personality and gender prediction (Verhoeven et al., 2016; Busger op Vollenbroek et al., 2016), and authorship attribution tasks (Hürlimann et al., 2015; Diederich et al., 2003; Koppel and Schler, 2003)

For multiclass classification using binary classifiers, one can opt for a one-versus-rest (we train n classifiers, one for each class) or a one-versus-one scheme, (in which we would need to train $\frac{n*(n-1)}{2}$ classifiers and have each classifier vote on the resulting class). Here we opt for the former approach, training one SVM per candidate author.

Dataset and preparation To see if we could discriminate between different review authors, we took the 50 most-prolific reviewers in the Yelp Reviews file. Each of these authors had left more than 100 000 characters of review text in total (across multiple reviews). For each author, we concatenated all of his or her reviews into a single text. We took the first 50 000 characters as training data and the next 50 000 characters to create ten test texts for that author, with each text being 5 000 characters in length, resulting in 500 test texts in total.

We vectorized the texts using TF-IDF (Robertson, 2004), a scheme in which all terms are represented by their frequency, but lower weights are given to terms that appear in many different documents. The resulting vectors consist of word and character n-grams, with unigrams and bigrams for words and bigrams and trigrams for characters. This resulted in a feature set consisting of 806 236 distinct terms (fitted only on the training texts; terms in the test texts that were not found in the training texts were ignored).

Classification We then trained Support Vector Machines, as described above, on the 50 known texts. We used the scikit-learn (Pedregosa et al., 2011) "LinearSVC" implementation with default parameters. Predictions were generated for each of the test texts based on the confidence scores assigned by the SVMs, assigning each test text to its most likely author.

3.1.2 Model 2: Word-level GRU-RNNs

Recurrent Neural Networks with word embeddings have been used successfully in other text classification tasks, such as sentiment analysis (Tang et al., 2015). Instead of using frequency-based features, as we discussed above, a Neural Network can "read" a text sequentially, similarly to how a human might. The network can then learn the most relevant words and relations, and discriminate between classes.

We use an encoder architecture, which reads each review one word at a time using a recurrent neural network. The resulting document vector is fed to a fully connected layer for classification. We tried both one- and bidirectional encoding. In the latter case, we would have two GRU encoders run through the text in both backward and forward direction and concatenate the result into one vector. We present results only on the one directional model, as the bidirectional model showed no improvement.

Dataset and preparation We used the same dataset as described above in 3.1.1, but instead of TF-IDF vectors we employed untrained word embeddings. We built the vocabulary out of the whole training set. Unknown words were replaced with an ¡UNK¿ token. We experimented with several vocabulary sizes by taking the most frequently occuring words. For each author, we split the 50000 character chunks into ten 5000-character samples for training. We tokenized each text chunk into words and transformed them into a matrix of one-hot-encoded vectors. We padded the length of each sequence to 900 words.

Classification The one-hot-encoded reviews extract the actual word embedding vector from an 128-dimensional word embedding matrix. The resulting sequence of vectors is fed into the Gated Recurrent Unit layer with a dimension of 256. The final output of the GRU layer is then passed to a fully connected output layer with softmax activation to directly identify the author.

We employed Batch Normalization in the recurrent layer to improve training speed and Dropout with a keep probability of 0.9 as a form of regularization. During training, we used 10% of the training data as a validation set to measure generalization. We used the Adam optimizer and cross-entropy as the loss function. We trained the network for 10 epochs.

3.1.3 Model 3: Language modeling

In an oft-cited blog post titled *The Unreasonable Effectiveness of Recurrent Neural Networks*[1], Karpathy showed how well character-level recurrent neural networks were able to generate text in a specific style. For example, his network was able to automatically generate text that closely resembles that written by Shakespeare, an excerpt of which is given below:

> KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

If we are able to generate text in the style of a specific author, then we should be able to use the same method to discriminate between different authorship styles. Neural models can be generative or discriminative, and using generative models for classification is an active topic of research (Yogatama et al., 2017). To discriminate between writing styles, we create one generative language model per known author. To predict the author of an unknown text, we find the author that best matches that text, by finding the model that gives the lowest cross entropy score for that text. That is, we find:

$$i^* = argmin_i(CE(LM_i(t))$$

where CE is cross-entropy, LM_i is one of language models, trained exclusively on the relevant authors text, and t is the unknown text. This is similar to the models we previously discussed by Bagnall (2015, 2016).

Dataset and preparation We used the same dataset as described above in 3.1.1. Instead of using TF-IDF vectors, we used character embeddings. First, we preprocess each text, converting any sequence of whitespace tokens, including newlines, to a single space. We worked with an alphabet consisting of the uppercase and lowercase characters of the English alphabet, the standard punctuation symbols

```
!"#&'()*+,-./:;<=>?@[]^_`{|}~%$\
```

and a space character, which we hereon indicate as SPACE. We then transform each text into a set of partially overlapping sequences of 100 characters each, with each sequence of 100 being mapped to the 101st character. To reduce the number of sequences, we use a step-size of three characters, effectively skipping some of the overlapping sequences. For example, using a sequence of 10 characters the sentence "A quick brown fox jumps over the lazy dog." would be represented by the sequences shown in Table 1. Finally, each character is converted to an integer, based on its index in our alphabet.

Classification For each author, we train a GRU-RNN to predict the next character, given the previous 100 characters. The model consists of an Embedding layer that learns 300 dimensional embeddings for each character, a Gated Recurrent Unit layer with dimension 250, and a fully connected output layer, which uses a softmax activation function to choose one of the 85 characters in

[1] http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sequence	Next
A SPACE q u i c k SPACE b r u i c k SPACE b r o w n k SPACE b r o w n SPACE f o (etc.)	o SPACE x.

Table 1: Example of partially overlapping sequences for 10 characters (note that for the actual model we used sequences of 100 characters).

our alphabet as a prediction. We further add a 10 percent chance of Dropout after the Embeddings layer and a 30 percent chance of Dropout after the GRU layer. We perform Batch Normalization after each Dropout step. We use the Adam optimizer and cross entropy as a loss function.

We chose these hyperparamaters by trying different variations on a single author's model, and attempting to find a configuration that resulted in the lowest cross-entropy loss score when holding out 10 percent of the 50 000 character training data as a validation set.

We predict the author for each of our 500 test texts by evaluating each text under each author's language model. Even though each evaluation procedure is computationally efficient, we need to do this 25 000 times (500 texts * 50 models), meaning that generating the predictions is more computationally expensive than the actual training.

3.2 Verification

Our fourth model concerns Authorship Verification, and we present a simple novel approach in which we take the absolute difference between the TF-IDF vectors of the known and unknown texts and use SVMs to make *same-author* and *different-author* predictions based on this representation.

As discussed above, in the Authorship Verification task, our goal is to predict whether or not two texts are written by the same author. In the previous literature, the Authorship Verification task is often modelled as a similarity task. If the two texts are "similar" by some metric, then this provides evidence that they are written by the same author. Because similarity scores are usually modelled on a scale, some threshold has to be set. If the texts are evaluated as being "more similar" than the threshold, a *same author* prediction is made; otherwise, we predict the *different author* label. While these similarity based approaches have proven effective, we present an approach here where no threshold is needed.

The difficulty in modelling the verification task comes through the fact that we always need to consider two input texts at a time. In normal classification tasks, we assign one label per example, but in Authorship Verification we need to assign one label per *pair* of example texts, looking at the relation between the two texts.

We represent both texts as a single example by calculating the TF-IDF vector of each, and taking the absolute difference between each element of the resulting matrix. This results in a matrix that represents how similar the texts are on a feature-by-feature level. For example, if two texts use the bigram "my favourite" with similar frequency, these will effectively cancel out, leaving a small or zero element in the matrix. If there are many semicolons (;) in one text, and none in the other, there will be a large value for the TF-IDF element that represents the semicolon. With the pair of texts represented in this fashion, the intuitive ideas is that the classifier is able to work out which features are representative of authorship style (probably features with low TF-IDF counts in same-author pairs, as the author uses these features consistently over different texts thus having them effectively "cancel out") and which features to ignore (probably features with high TF-IDF counts in same-author pairs, as these features vary in frequency per text irrespective of authorship).

Dataset and preparation As with the identification task, we concatenated all of the reviews of each author, and divided them into fixed chunks. For authorship verification, we need a small amount of text from a large number of authors. This is different from the identification task, where we needed a large amount of text from each of a small number of authors. In the Yelp review dataset, there are 38 985 unique authors who have produced at least 10 000 characters in reviews. First, we took 10 000 character texts from 38 900 unique authors (dropping 85 texts to work with a round number). We divided each text into two subtexts of 5 000 characters each, one for the *known* text and one for the *unknown*, and we put these texts into $known$ and $unknown$ arrays respectively. Each text in the unknown array is paired with a text by the same author in the known array. We then shifted the texts in the second half of the $unknown$ array by one, so that each text in

the second half of the $unknown$ array was paired with a *different-author* example in the $known$ array. Assuming eight texts, this would look as seen in Table 2.

$k = 1, 2, 3, 4, 5, 6, 7, 8$
$u = 1, 2, 3, 4, 5, 6, 7, 8$
$k = 1, 2, 3, 4, 5, 6, 7, 8$
$u = 1, 2, 3, 4, 8, 5, 6, 7$

Table 2: Example of pairing same-author and different-author verification examples. All pairs are *same-author* in the first two arrays (before shift), while half are *different author* in the second two (after shift).

To gain some insight into whether it is easier to classify shorter texts when more training examples are present, or longer texts with fewer examples, we also created a similar dataset using 6 000 characters per text pair. We set up this dataset in the same way as above, resulting in a second dataset consisting of 71 300 text pairs.

A summary of the two datasets and how we split them into train and test sets is given in Table 3.

	Long	**Short**
Author Length	10 000	6 000
Text Length	5 000	3 000
Number of Authors	38 900	71 300
Train Examples	30 000	60 000
Test Examples	8 900	11 300
Feature Size	24979	17662

Table 3: Description of our Authorship Verification datasets. Author Length refers to the total number of characters of text we used to represent one author. Text Length is half of this to account for creating *known* and *unknown* texts for each author.

We converted each text into a TF-IDF vectorization in the same way as described in 3.1.1. One difference was that we used only the most frequent features, dropping terms that were used by fewer than one percent of authors in each dataset. Finally, we converted each respective *known* and *unknown* pair of texts into a single sample by taking the absolute value of the difference between them.

Classification We used the same classification method described in 3.1.1. Because this is a binary classification task, only two SVMs are required, as opposed to the 50 required for the identification task.

4 Results and discussion

We evaluated each of our models against the datasets described above. The models relating to the Authorship Identification task are compared against each other, as we used the same subset of the Yelp review dataset for each of them. Then we compare the single Authorship Verification model by evaluating its performance on the two verification datasets we described above, one with longer texts but fewer examples, and the other with shorter texts and more examples.

4.1 Authorship Identification

For the identification experiments, we achieved equal results using Support Vector Machines with TF-IDF vectors and the generative character-based language models. However, the Support Vector Machine approach was substantially more efficient for both training and for generating predictions, as can be seen in Table 4. The word-based discriminative neural network did not perform well, and we discuss potential reasons for this below.

	SVM	**Char-NN**	**Word-NN**
Accuracy	0.90	0.90	0.03
Train time	00:00:27	02:35:50	00:05:32
Test time	00:00:00	12:29:40	00:00:00
Total time	00:00:27	15:05:30	00:05:32

Table 4: Results for the Identification task. Train time includes vectorization and preprocessing time.

For the language modelling experiment we used an AWS p2.xlarge machine with four CPU-cores, 61 GiB RAM and an NVIDIA K80 GPU, with 2 496 parallel processing cores and 12 GiB of GPU memory. The other experiments were run on a 2015 MacBook Pro with a 2.7 GHz i5 processor and 8 GB RAM.

Note that although the generative language modelling predictions took an excessively long time to generate, these are easily parallelizable. Because each language model needs to evaluate each text, it would be trivial to share this workload amongs multiple cores or machines by deploying one model and all test texts per machine.

A random baseline approach would have a $1/50$ chance of guessing the correct author, so we would expect an accuracy of 0.02. Given that the test texts are relatively short (5 000 characters is approximately 900 words), the task is not easy and it is evident that both the SVM with TF-IDF model and the character-based language model are proficient at discriminating between authorship style. We were surprised by the fact that the two models achieved identical results, but pleased with the indication that the less-common neural approach proved suitable for the task.

It is surprising that our word-based discriminative RNN achieved such poor results on this task, when similar models have worked well for other text classification tasks. We believe that this is partly due to the fact that while word embeddings are good at discriminating between the *meaning* of a text (for example, in sentiment analysis tasks), they are not good at discriminating between text *style*. This is because the idea behind word embeddings is to represent words that appear in similar contexts as being close in vector space. When we discriminate between authorship styles, we want to separate different ways of saying similar things. While our model could easily learn the classes in the training set, it did so in a way that did not generalize well to held-out test data.

4.2 Authorship Verification

For the Authorship Verification task, we achieved the results shown in Table 5.

	Long	**Short**
Accuracy	0.95	0.92
Train time	00:00:27	00:24:02
Test time	00:00:00	00:00:00
Total time	00:00:27	00:24:02

Table 5: Results for the Verification task. Train time includes vectorization and preprocessing time.

We can see that shorter texts are more difficult to accurately classify, even though we had more examples to train on (71 300 and 38 900 respectively). However, in both cases our novel approach achieved satisfactory results. As discussed before, these results are especially interesting as the authors in the test set are disjoint from those in the training set (in fact, no author is represented more than once). It is impressive that even using a simple approach, a supervised classifier is able to make meaningful decisions about authors that it has not seen examples of during the training phase.

Although we still need a fair amount of text for each known author, this is far less than what is needed to achieve reasonable results in an identification task. As mentioned before, the verification task can be used a stepping-stone to solve many other authorship attribution tasks, including identification. We could, for example, do pairwise comparisons across all reviews in a corpus to find the likelihood of a single author using multiple accounts, or do a pairwise comparison of all the reviews associated with a single username to find out if it's likely that more than one person uses that account.

5 Conclusion and future work

We presented four experiments using different combinations of features and classifiers, all relating to supervised machine learning for Authorship Attribution. We showed that SVMs with TF-IDF vectors can provide a simple, efficient, and highly accurate approach to some Authorship Attribution tasks, but we expect that even better results can still be achieved.

Would results improve if we combined word embeddings with TF-IDF vectors? How would it affect performance if we combined character-based and word-based models? Additional work is also needed to compare our results more closely to existing approaches, both in terms of using our methods on other datasets, and also using other methods on the datasets that we used in our research.

We plan to run more experiments to see how well the classifier trained on Yelp reviews can generalize to other genres of text. For example, can the classifiers we trained in these experiments discriminate between novelists, even without being retrained on novels, or is it the case that the stylistic features that they rely upon to achieve good results on the review data is specific to the writing style usually found in internet reviews.

Finally, while we argued that this research has practical applications, we would like to see industry-specific implementations to prove that our methods can be applied satisfactorily in real-world tasks, such as banning users with multiple accounts, or detecting accounts that are used by more than one user.

References

Douglas Bagnall. 2015. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891* .

Douglas Bagnall. 2016. Authorship clustering using multi-headed recurrent neural networks. *arXiv preprint arXiv:1608.04485* .

Mart Busger op Vollenbroek, Talvany Carlotto, Tim Kreutz, Maria Medvedeva, Chris Pool, Johannes Bjerva, Hessel Haagsma, and Malvina Nissim. 2016. GronUP: Groningen User Profiling—Notebook for PAN at CLEF 2016. In Krisztian Balog, Linda Cappellato, Nicola Ferro, and Craig Macdonald, editors, *CLEF 2016 Evaluation Labs and Workshop – Working Notes Papers, 5-8 September, Évora, Portugal*. CEUR-WS.org. http://ceur-ws.org/Vol-1609/.

Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. 2003. Authorship attribution with support vector machines. *Applied intelligence* 19(1):109–123.

Manuela Hürlimann, Benno Weck, Esther van den Berg, Simon Suster, and Malvina Nissim. 2015. Glad: Groningen lightweight authorship detection. In *CLEF (Working Notes)*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98* pages 137–142.

Moshe Koppel and Jonathan Schler. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*. volume 69, page 72.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the Association for Information Science and Technology* 60(1):9–26.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation* 45(1):83–94.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Yaron Winter. 2012. The fundamental problem of authorship attribution. *English Studies* 93(3):284–291.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology* 65(1):178–187.

Jana Kravalová and Zdeněk Žabokrtský. 2009. Czech named entity corpus and svm-based recognizer. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*. Association for Computational Linguistics, Stroudsburg, PA, USA, NEWS '09, pages 194–201. http://dl.acm.org/citation.cfm?id=1699705.1699748.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 513–520.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation* 60(5):503–520.

Paolo Rosso, Francisco Rangel, Martin Potthast, Efstathios Stamatatos, Michael Tschuggnall, and Benno Stein. 2016. Overview of pan16. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pages 332–350.

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

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. 2014. Overview of the author identification task at pan 2014. In *CLEF (Working Notes)*. pages 877–897.

Efstathios Stamatatos, Martin Potthast, Francisco Rangel, Paolo Rosso, and Benno Stein. 2015. Overview of the PAN/CLEF 2015 Evaluation Lab. In Josiane Mothe, Jacques Savoy, Jaap Kamps, Karen Pinel-Sauvagnat, Gareth J.F. Jones, Eric SanJuan, Linda Cappellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 6th International Conference of the CLEF Initiative (CLEF 15)*. Springer, Berlin Heidelberg New York, pages 518–538. https://doi.org/http://dx.doi.org/10.1007/978-3-319-24027-$5_4$9.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*. pages 1422–1432.

Ben Verhoeven, Walter Daelemans, and Barbara Plank. 2016. Twisty: A multilingual twitter stylometry corpus for gender and personality profiling. In *LREC*.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* .