ConHAN: Contextualized Hierarchical Attention Networks for Authorship Identification

Jean Bouteiller
Massachusetts Institute of Technology
jeanbout@mit.edu

Victor Jouault
Massachusetts Institute of Technology
vjouault@mit.edu

Jack Schooley
Massachusetts Institute of Technology
jschool@mit.edu

Abstract

Authorship identification is a common and important task in the field of Natural Language Processing. Defined as the task of predicting the author of a given text, it enables us to accomplish a broad range of tasks, from identifying ghost writers to detecting plagiarism. We show the advantages of using neural network architectures over classic machine learning approaches in terms of accuracy. We first reproduce literature approaches which we benchmark on the Reuters_50_50 dataset, and then improve on them by introducing ConHAN, a new deep learning architecture combining pre-trained contextualized embeddings (DistilBERT) and a hierarchical attention architecture. Among all models we tested, the best model achieves an out-of-sample accuracy of 78%, compared to the literature benchmark of 69%. Despite a small training size, ConHAN shows promising results on this particular task and we firmly believe it would benefit from a bigger training set. We further analyze our model and discover that ConHAN performs best when predicting authors with simple vocabularies, but it is agnostic to many sentences' grammatical differences.

1 Introduction

Authorship identification is the task of predicting the author of an article, book, or other piece of writing based solely on the text. This task has a variety of applications, including identifying the original author of a widely circulated text, examining documents for plagiarism, and recommending similar authors.

In this paper, we use the Reuters 50-50 (C50) dataset, a well-known dataset for authorship identification. We first construct a baseline random forest model using features like the sentence lengths and the parts of speech present in the text. Then, we implement Recurrent Neural Network (RNN) architectures in the form of Gated Recurrence Units (GRUs) at both the sentence and article levels and achieve comparable results to those previously seen in the literature.

After these RNN implementations, we leverage pre-trained transformers in the form of DistilBERT, from which we can obtain contextualized word embeddings. We first use DistilBERT's beginning of sentence token alone, denoted [CLS] (short for "classification"), as our document embedding, and even by itself this token produces powerful results. However, this simple approach does not use most of the contextualized representations, so our next model uses all of the sentence's contextualized word embeddings as the input to an attention head, which learns a weighted average of the embeddings that can be used for classification.

Our last set of models implement hierarchical attention networks on top of DistilBERT. This additional structure leverages the word attention layer generated previously and adds another attention head on top of it at the sentence level. After generating the sentence representations through the first attention layer, the second attention layer learns a weighted average of each sentence in the document which is used to create the final article embedding. This article embedding is then used for classification.

One other consideration for this architecture deals with how the sentence representations are encoded. Because we use pre-trained DistilBERT representations at the word level, the generated embeddings are already contextualized. This is not the case for the representations at the sentence level, so we also implement a mechanism that accounts for sentence order and context within the article via a bidirectional GRU to encode the sentences. We implement hierarchical attention architectures both with and without this contextualization GRU, and we coin the term ConHAN (Contextualized

Hierarchical Attention Network) to describe the latter.

2 Related work

Authorship identification is a common task in Natural Language Processing (NLP) and a large number of publications focus on this topic. For example, (Stamatatos et al., 2014) present an overview of different software submitted to the PAN-2014 benchmark, which consists of an authorship identification task.

Different document characteristics have traditionally been used to extract the author of a text. (Zheng et al., 2006) develop a framework for authorship identification in which lexical, syntactic, structural and content-specific features are extracted from texts. Others have tried to use the network structure of texts (Akimushkin et al., 2018).

On short documents of a few sentences, (Houvardas and Stamatatos, 2006) leverage n-gram features, under the assumption that character-level n-grams are able to represent text for stylistic purposes, and achieve a 74% accuracy on the Reuters_50_50 dataset. (Qian et al., 2017) introduced an RNN deep learning approach which we attempt to reproduce, and achieve 69% accuracy on the same dataset. These approaches can capture semantic and stylistic components but fail to reproduce the structure of the documents.

In the Natural Language Processing community, (Yang et al., 2016) introduced the Hierarchical Attention Networks (HAN) for document classification, which leverage a hierarchical structure that mirrors the structure of a document along with 2 levels of attention mechanisms.

Therefore, we aim at combining the 2 former approaches under one common framework, ConHAN. Since structure is an important characteristic of a document when it comes to identifying the author, we believe this novel architecture should be able to better capture this aspect of the texts.

3 Theoretical approach and models

This section focuses on the models we designed to solve our specific task, from word embeddings to the downstream classification task. We first describe a baseline machine learning model based on a Random Forest model. Then, we elaborate on Recurrent Neural Networks achieving near-state of the art deep learning results in the literature. Finally, we detail our own contribution through the use of pre-trained contextualized embeddings and the implementation of hierarchical networks fine-tuned for the author identification task.

The deep learning approaches all share the same goal: to produce the most accurate document embedding possible. This representation is then fed to a classification head with *softmax* activation to output a probability distribution over the authors in our dataset. Negative log likelihood of the correct label is used as a training loss across all models. Therefore, neural network approaches differ solely in the way this document-level embedding is produced.

3.1 Baseline model

Before running any deep neural models, we used a classical Machine Learning approach, with features that can be easily interpreted. Using recommendations from linguistic experts, we created features that could allow us to distinguish authors' writing styles, using open source packages:

- **Sentence Length**: Average number of words per sentence
- **Part of Speech Tagging**: Using the *nltk* part of speech tagging package, we were able to count the average number of nouns, pronouns, verbs, adjectives and adverbs per sentence.
- **Part of Speech Tagging N-grams**: we have computed the most frequent type of word sequences (e.g: Noun-Verb-Adverb for a 3-gram), for bigrams, 3-grams and 4-grams. For each article, we computed the average number of each n-gram per sentence.

With a **Random Forest model** with 200 estimators and trees of max depth 5, we achieved a cross-validated accuracy out-of-sample of 11.8% (36.6% in-sample), and an F1-score of 10.1% out-of-sample.

3.2 Recurrent Neural Networks

We build two models using Recurrent Neural Networks architectures, at the sentence and at the article level.

For the preprocessing, we take the text from each article and parse it, removing all characters that are not letters or numbers. We then separate each article into sentence blocks separated by line breaks; each sentence block is actually 3 sentences approximately, but henceforth we will refer to each block as the sentence itself. Once we separate

the sentences, we then separate each sentence into individual words. During preprocessing, we also obtain the sentence lengths for each article, which are the number of words in each sentence. We use these lengths later when packing the inputs to the model.

For each RNN model in this section, we initialize our word representations using the GloVe.6B embeddings of size 50. The initial input to each model is a tensor of word tokens that gets mapped to their respective GloVe embeddings in the embedding layer. We utilize a padding index to account for uneven sentence lengths in terms of words and article lengths in terms of sentences. We backpropagate the gradients to the embedding layer with the exception of the padding index, which stays constant throughout training.

We use the GRU architecture as a basis of these models, which updates the hidden state h_t according to the following equations:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}) \quad (1)$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1} + b^{(r)}) \quad (2)$$

$$\tilde{h_t} = \tanh(r_t \circ U^{(h)}h_{t-1} + W^{(h)}x_t + b^{(h)}) \quad (3)$$

$$h_t = (1 - z_t) \circ \tilde{h_t} + z_t \circ h_{t-1} \quad (4)$$

We train each RNN with stochastic gradient descent as our optimization algorithm, using a learning rate of 0.1 each time. We use 15 epochs for both RNNs, along with a 90/10 training/testing split for these RNNs.

3.2.1 Sentence Level GRU

For the sentence level RNN model, after getting the word embeddings, we pack the inputs to account for padding. We input these packed embeddings into the GRU, and extract the hidden states as our output. We use a hidden size of 300 for the GRU. Finally, we input the computed hidden states into the final linear layer, which produces a normalized probability vector after softmax activation, as seen in Figure 1.

Each batch is actually a single whole article. The batch input tensor is padded to ensure that the sentence sequence length is constant across each batch. When it comes time to pack the padded inputs, we use the sentence lengths for each article that were obtained earlier. The sequence length for each batch is equal to the maximum sentence length, which can vary among batches. In addition, because each batch is a single article, the batch size

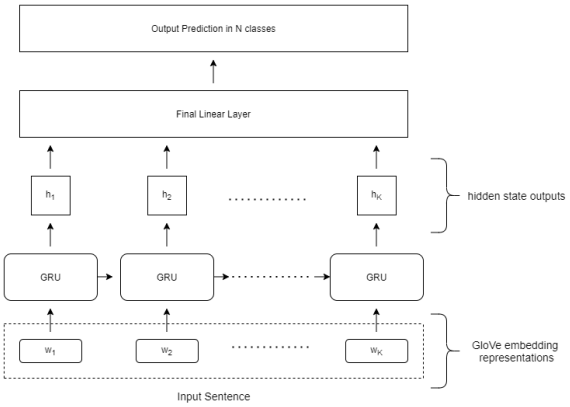

Figure 1: Sentence Level GRU

is equal to the length of each article in terms of sentences, which is also variable.

While this approach means that our input tensors are not uniform in size across batches, the advantage is that we can minimize the effects of padding on our model. This means that the model will not learn the padding index at all, thus it will only focus on learning the actual words in the text. Another advantage here is that we do not need to select an arbitrary sequence length to adhere to, meaning that we do not need to truncate inputs for this model. This approach provides fairly powerful results at the sentence level.

3.2.2 Article Level GRU

For the article level RNN model, we first need to do some additional preprocessing on top of what was needed for the sentence level model. While we were able to avoid setting a fixed sequence length at the sentence level, our article level architecture demands that we make the sentence representation lengths uniform this time. We choose this fixed sequence length to be 25 after some experimentation; this is a relatively short length that goes against intuition, but our sense is that this value strikes a balance between getting enough words in each sentence to be meaningful as well as minimizing the amount of padding present.

After getting the word embeddings, we create our sentence representations using an average pooling layer. Thus, each sentence representation is just a simple average of all the word representations, including any padding present. After we obtain sentence representations, we then pack the inputs before feeding them into the GRU, once again using a hidden size of 300. We then extract the hidden states from the GRU and again feed them into

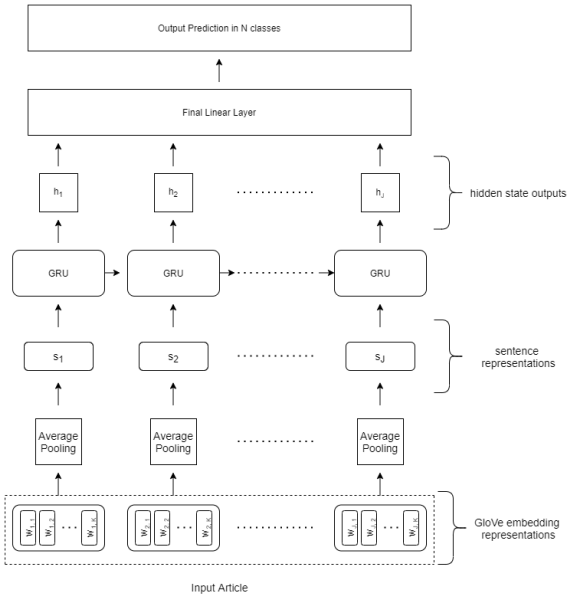

Figure 2: Article Level GRU

the final linear layer, where we apply the softmax operator to obtain normalized probabilities. The architecture can be seen in Figure 2.

We use a batch size of 10 articles. When packing the inputs, we compute the article lengths (in terms of sentences) beforehand and use them in the packing process. We pad articles to ensure that the sequence length in terms of article length is constant across each batch.

3.3 Pre-trained contextualized embeddings with DistilBERT

Transfer learning from large-scale pre-trained models has become more prevalent in NLP in the last few years with the introduction of mega-transformer models such as BERT (Devlin et al., 2018). These transformer-based models make it possible to obtain state-of-the-art contextualized embeddings that can then be fine-tuned on specific tasks. However, these are enormous models that are challenging to operate under constrained computational training. To cope with this problem, we use DistilBERT (Sanh et al., 2019), which combines an easy implementation with lower computational requirements, to produce contextualized embeddings.

In this subsection, the whole article is processed as one long input that is passed through the transformer model. The two models we present have only a single "level". This is a significantly different approach than the hierarchical models we highlight afterward, which offer 2 different levels to embed the structure of the article.

3.3.1 Sentence-level classification token

As stated previously and shown on figure 3, the model takes as input the whole tokenized article and feeds it to DistilBERT's transformer. If an article has K word tokens, then the outputs of the transformer are the K contextualized embeddings of the article's words.

DistilBERT's beginning of sentence token, *[CLS]*, stands for "classification" and plays an important role in sentence level understanding. During the pre-training phase, this token is trained on a *Next Sentence Prediction* task. Therefore, it encompasses the full meaning of the sentence and is the equivalent of the last hidden state of a Recurrent Neural Network. In this first model, we consider this as our document's embedding. We append a linear classification head with *softmax* activation on top of the [CLS] token, as described on figure 3.

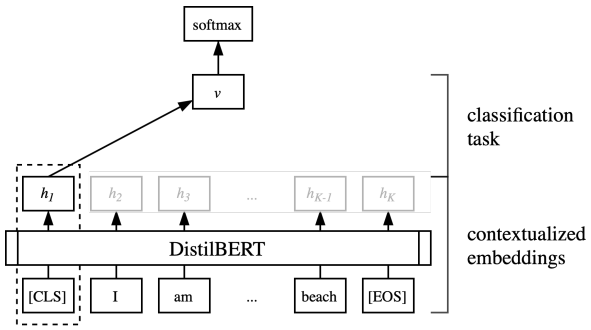

Figure 3: Sentence-level classification token architecture

3.3.2 Word-level attention architecture

While this first model is efficient and makes a good use of the *[CLS]* token, it is rather simple and a lot of the information is not used. Even though the whole model is fine-tuned on the training set, the specific words chosen by the author, as well as the article's structure, might be lost in the process. We add an attention layer on top of our contextualized embeddings with the hope that the model will be able to learn which words to attend to. The document's embedding is then a weighted average of all the words' contextualized embeddings, as described on figure 4.

Specifically, attention mechanisms, introduced in (Bahdanau et al., 2014), help extract words that are most important to the author identification task and aggregate the representation of those informative words to form a document embedding. We use

the following equations:

$$u_i = \tanh(W_w h_i + b_w) \quad (5)$$

$$\alpha_i = \frac{\exp(u_i^T w_{att})}{\sum_j exp(u_j^T w_{att})} \quad (6)$$

$$v = \sum_i \alpha_i h_i \quad (7)$$

In other words, we first feed the contextualized word embeddings h_i into a linear layer to obtain u_i, the hidden representation of the word embeddings. Then, we compare them to a constant context vector w_{att}, and we scale the results through a softmax function to obtain the vector α. This enables us to get normalized weights which we can then use to produce the document embedding vector v as a weighted average of the contextualized article word embeddings. Again, we then feed this document's embedding to a classification head to output probabilities over the authors.

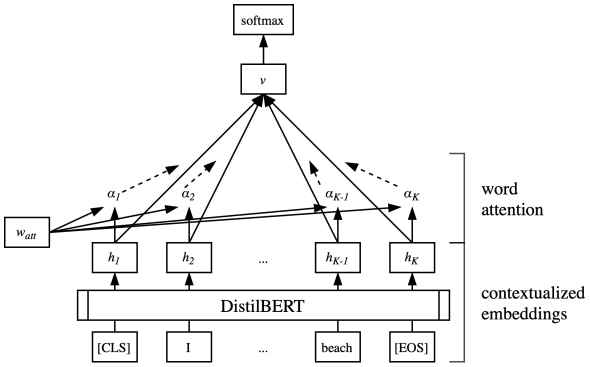

Figure 4: Word-level attention architecture

3.4 Hierarchical attention networks

Hierarchical attention networks have been introduced (Yang et al., 2016) to help with document classification. The main contribution of these models is to leverage the hierarchical structure of documents to obtain a better representation. (Yang et al., 2016) showcases the benefits of such models in a wide range of large-scale classification tasks. We have reasons to believe that incorporating this structure could help us to identify authors, since a document's layout is purely subjective based on the author.

This subsection leverages this specific architecture for the author identification task. Each article is therefore divided into sentences or smaller parts. The unit input of the first level is a word, while the unit input of the second level is a sentence.

3.4.1 Simple Hierarchical Attention Network

The model's architecture is shown in figure 5. Assume that an article has L sentences s_i and that each sentence contains T_i words, represented as w_{it} with $t \in [1, T]$.

Our proposed model first takes the raw article and divides it into words and sentences. Each word is projected into a contextualized vector representation thanks to the use of a *DistilBERT* transformer.

Then, we use the word attention mechanism described above to obtain a sentence-level embedding for each sentence in our article.

Finally, to reward sentences that are the most effective at correctly identifying an author, we again use an attention mechanism at the sentence level. The document level embedding is therefore the weighted average of the article's sentence level embeddings.

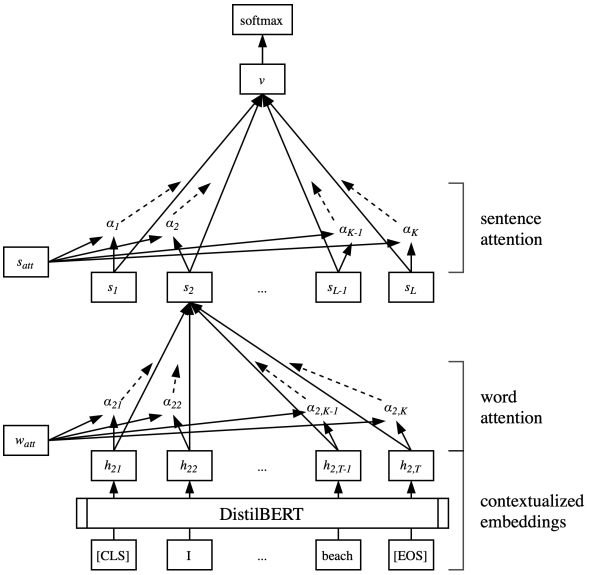

Figure 5: Simple Hierarchical Attention Network architecture

3.4.2 ConHAN: Contextualized Hierarchical Attention Network

While the Simple Hierarchical Attention Network we just presented allows us to leverage the article's structure, the simple sentence-level attention mechanism does not include any positional information about the sentences. In comparison, thanks to the use of contextualized embeddings, word order is already accounted for in the model and justifies the direct use of an attention mechanism.

We add a sentence encoder between the word attention layer and the sentence attention layer to

account for the sentence order. This sentence encoder is composed of a GRU network, which has been described in subsection 3.2.

Specifically, each sentence-level embedding s_i, resulting from a weighted average of the word-level embeddings $h_i t$, is fed to a bidirectional GRU to encode the sentences. We then concatenate both direction's hidden vectors h_i^1 and h_i^2 to get a hidden annotation of sentence i.

We obtain the final model architecture described in figure 6: ConHAN. It is composed of a contextualized embedding layer that takes word tokens and outputs an embedding for each of them. Then, word embeddings are taken through a weighted average thanks to the word attention mechanism to obtain sentence-level embeddings. Sentence representations are then fed to the sentence encoder to get a positional annotation of the sentences. Finally, we compute the document-level embedding thanks to a sentence attention mechanism. A classification head is appended on top to output probabilities over the known authors in the dataset.

To the best of our knowledge and in comparison to the original paper (Yang et al., 2016), we are the first to leverage pre-trained contextualized word embeddings via DistilBERT in a hierarchical framework.

All the implementation characteristics (sentence length, batch size, hidden size, ...) for subsections 3.3 and 3.4 can be found in the source code linked at the end of the article.

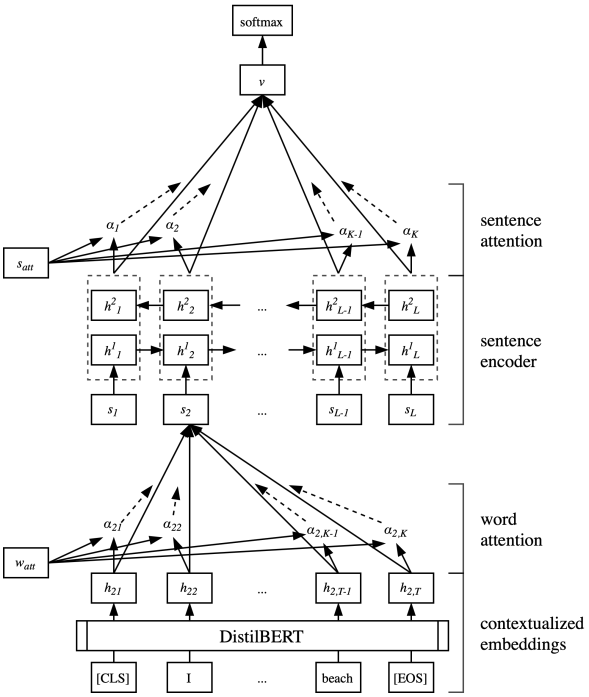

Figure 6: ConHAN architecture

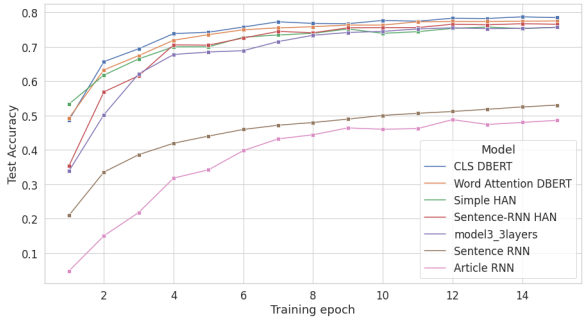

Figure 7: ConHAN Test Accuracy vs epoch

4 Experiments and Results

4.1 Datasets

We use the Reuters_50_50 (C50) dataset to train and evaluate our models. It is a subset of the Reuters Corpus Volume 1 (RCV1) and contains 5000 articles from 50 different authors. In order to prevent any topic dependency, all articles selected have been tagged with at least one subtopic of the class CCAT (corporate/industrial). We split this data in a non-overlapping 75% / 25% train/test set.

On this dataset, (Qian et al., 2017) have achieved an accuracy of 69%, which serves here as a benchmark.

4.2 Model performances Evaluation

We showcase our results in Figure 7.

4.2.1 Performance Metrics

.

In order to compare the performance of our models (baseline, GRU, Hierarchical Attention, etc.), we use several multi-classification metrics mentioned in (Grandini et al., 2020).

We create a measure of the ability to perform equally among all authors, called *Author Standard Deviation,* σ. Let A be the distribution of authors' accuracy and A_i be the accuracy for author i. σ is the standard deviation of the distribution A. We also use a macro F1-score (average of Author-Level F1 score). Table 1 presents the performance metrics for the different models used.

First, please note that all our Deep Learning models achieve an in-sample accuracy of 1, which means we are overfitting the training set. (Cf Interpretation and Discussion). However, testing accu-

racy kept on increasing, which is why we compare our models after 15 epochs.

We also display the confusion matrix for the Hierarchical Attention with RNN model.

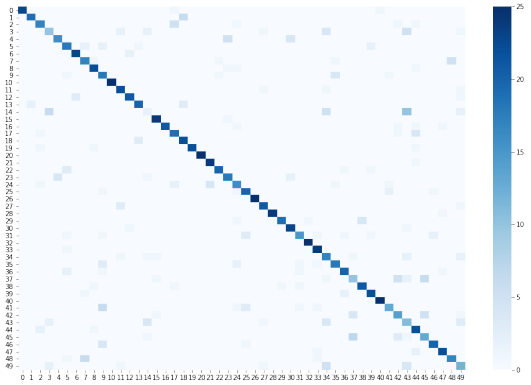

Figure 8: Confusion matrix for HAN RNN model

Comments:

- A naive approach, picking randomly one author among the 50 has an accuracy of 2%, so all these models improve significantly a random selection.
- The Confusion matrix emphasizes that **there are discrepancies among authors**: Our model performs better on some authors than on other.
- A hierarchical approach with attention does not outperform the CLS token of BERT embedding. We can explain that by mentioning the limited size of our dataset (75 articles per author).
- ConHAN has a higher author standard deviation than without the last RNN layer, but it has better accuracy and F1 score. That means that it has an overall better performance, but they are higher discrepancies among different authors. It performs no so well on certain authors, and very well with others.
- **Plagiarism Discussion:** if we want to use our model for plagiarism detection, given an article and an author, we need to be confident when our model predicts negative (the article is not from this author). Indeed, if an article is predicted negative, that may lead to justice condemnation, so we need to have certainty on this output. That means that **we should optimize on the True Negative Rate** if want to use this models on some concrete cases.

5 Model Interpretation

In this section, we focus on the last model (ConHAN) even if it has not the best performance. We believe that this model can outperform other models with a more extensive dataset, and also that its properties may yield greater insights due to its complexity.

5.1 Author Level Interpretation

We can wonder if our model performs equally among authors. The first metric we look at is the Author Standard Deviation σ. As highlighted above, ConHAN has the worst σ, which that means that our model performs really well with some authors but poorly with others. We have therefore decided to investigate this observation by comparing authors together. We remind the reader that A is the distribution of author-level accuracy.

5.1.1 Vocabulary Richness

Here, we analyzed the relationship between vocabulary diversity of an author and our model performance. We use a richness metric K mentioned in (Tanaka-Ishii and Aihara, 2015), which measures the number of times that each word is used in a text. To do that at an author level, we create one unique text per author, as the concatenation of all their articles. We note N the number of distinct words, $V(N, m)$ the number of words appearing N times, and S_1 and S_2 as the first and second moments of the distribution of $V(N, m)$. We have:

$$K_i = 10^4 \frac{S_2 - S_1}{S_1^2}$$

$$K_i = 10^4 [-\frac{1}{N} + \sum_{m=1}^{m_{max}} V(N, m)(\frac{m}{N})^2]$$

We denote K the distribution of author-level richness, and we measure the correlation between A and K. We obtain $corr(A, K) = -40.2\%$, so we can say that our model performs best on authors with a poor vocabulary. Some words may be used only in one article in the test set, so the model is not able to recognize the author. This issue may be solved using a bigger dataset.

5.1.2 Author Words' Specificity

When looking at the data, we see that some authors are always using the same words, and writing about one theme. As an example, *Benjamin Kang Lim* starts 60% of his text with the word "China". We

Model Name	Accuracy	Precision	Recall	F1	Author's StDev
Baseline	11.84%	11.3%	11.9	8.94%	0.179
Article GRU (Qian et al., 2017)	48.6%	47.9%	47.6%	46.7%	0.182
N-Gram (Houvardas and Stamatatos, 2006)	74.0%				
CLS DBert	78.48%	79.1%	78.5%	78.34%	0.173
Word Attention DBert	77.52	78.5%	77.5%	77.47%	0.178
Simple HAN	75.68%	76.0%	75.7%	75.3%	0.187
Sentence-RNN HAN	76.56%	77.0%	76.6%	76.36%	0.190

Table 1: Summary of Models' performance metrics

were then wondering if our model was not simply recognizing the words specific to each author in order to do the prediction. That would be problematic, because our model won't be able to predict correctly another article from the same author but about another topic.

We then performed an *author-level Tf-Idf*: for each unique word in an author's text, we look at this words frequency in other authors' text, and we can come up with a Tf-Idf Score $t(w)$ for each word w in each author's text, as mentioned in (Qaiser and Ali, 2018). For each author, we then denote T_n the n words with the highest $t(w)$. We denote I_i the word importance score for author i:

$$I_i = \sum_{w \in T} t(w)$$

We denote I as the distribution of author-level words importance for n=10, and we measure the correlation between A and I. We have $corr(A, I) = -14.69\%$, so we can say that our model performs best with authors who are not using specific words, less used by other authors. That means that our model is not recognizing some specific words to predict the author, but it is using a more global view of the article's content.

5.2 Statistical Approach to article level Interpretation

We can now investigate if our model performs well on any type of article, or if some specific types of articles are badly predicted by our model. To do so, we looked at the following elements in each article:

- **Meta-Data:** Average sentence length, Number of words per article, etc.
- **Text Statistics:** Article Vocabulary Richness and Words Specificity.
- **Entity Recognition** (using *spacy*): for each entity (person, money, Geography, etc.), we measure the number of times they are found in an article, normalized by the number of words in this article.
- **Part of Speech Tagging** (using *spacy*): for each tag (noun, verb, adverb, etc.), we measure the number of times they are found in an article, normalized by the number of words.
- **Stop Words**: We measure the frequency of stop words, using *nltk* stop words list.

We then used an approach mentioned in (Gomila, 2020) to analyze where our model performs well and doesn't. Leveraging a Linear Regression, we used the above-mentioned features to predict if an article is correctly predicted by our model. Here, the accuracy of the prediction is not a key element, but we will be focusing on the coefficient of the different parameters. Please refer to this link to know to what each acronym corresponds. We have the following results:

We can see all the coefficients used in the model, and most of them are not statistically significant. We have then decided to run a L1-regularized Linear Regression, selecting a subset of features, and then run another linear regression on this new subset. We have the following results:

Feature	Coef	P Value
Intercept	0.756	$< 10^4$
% Conjunction	-0.032	0.013
% Pronoun	0.042	0.005
% Geographical	-0.057	$< 10^4$
% Person	-0.038	0.002
% Noun	0.032	0.028

Table 2: Features' Coefficients after Lasso Selection

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7656	0.012	65.259	0.000	0.743	0.789
ADJ	0.0163	0.020	0.832	0.406	-0.022	0.055
ADP	-0.0176	0.017	-1.017	0.309	-0.052	0.016
ADV	0.0155	0.015	1.007	0.314	-0.015	0.046
AUX	0.0346	0.018	1.924	0.055	-0.001	0.070
CARDINAL	0.0156	0.023	0.677	0.498	-0.030	0.061
CCONJ	-0.0270	0.014	-1.889	0.059	-0.055	0.001
DATE	-0.0117	0.016	-0.718	0.473	-0.044	0.020
DET	-0.0004	0.021	-0.017	0.986	-0.042	0.041
GPE	-0.0575	0.022	-2.627	0.009	-0.100	-0.015
MONEY	0.0383	0.023	1.646	0.100	-0.007	0.084
NORP	-0.0081	0.015	-0.552	0.581	-0.037	0.021
NOUN	0.0548	0.029	1.863	0.063	-0.003	0.113
NUM	-0.0298	0.040	-0.737	0.461	-0.109	0.049
ORG	-0.0079	0.024	-0.329	0.742	-0.055	0.039
PART	-0.0346	0.015	-2.335	0.020	-0.064	-0.006
PERCENT	0.0165	0.016	1.029	0.304	-0.015	0.048
PERSON	-0.0366	0.016	-2.244	0.025	-0.069	-0.005
PRON	-0.0188	0.019	-0.997	0.319	-0.056	0.018
PROPN	0.0839	0.039	2.174	0.030	0.008	0.160
PUNCT	-0.0032	0.014	-0.225	0.822	-0.031	0.024
Pct_number	0.0364	0.035	1.038	0.300	-0.032	0.105
SCONJ	-0.0078	0.014	-0.562	0.574	-0.035	0.019
Sentence_Len	0.0301	0.016	1.937	0.053	-0.000	0.061
VERB	0.0483	0.021	2.249	0.025	0.006	0.090
Voc_richness	0.0025	0.017	0.146	0.884	-0.031	0.036
Word_Len	-0.0198	0.022	-0.887	0.375	-0.064	0.024
text_len	-0.0113	0.014	-0.828	0.408	-0.038	0.016
weight_max_words	0.0057	0.014	0.413	0.680	-0.021	0.033

Figure 9: Features Coefficients Linear Regression all features

Comments:
We can see that the features selected by the lasso regularizer are statistically significant, and we can draw the following observations:

- Our model is agnostic to all features that are in figure 9 and not in table 2. This means that our model performs almost as well on any type of article, whatever the writing style of the author.
- We still can see some discrepancies: when the coefficient is negative, that means that our model performs poorly on articles with a lot of these elements. For example, we can see that **our model performs best on articles with few geographical information**.
- We can suggest that this is because we have only a small training set (3,750 articles), and we might be over-fitting, which is confirmed by an in-sample accuracy of 1. We may need to use cross-validation to find optimal parameters and to reduce this overfitting.

All these observations will be key elements in order to help us improving our model in the future, as detailed below.

6 Discussion

While we are quite pleased with our results, there are still a number of limitations with our approaches, and therefore many opportunities still for improvement. Although our novel ConHAN architecture is extremely flexible and powerful, it did not perform the best out of all the models we tried. We attribute these results mainly to the relatively small size of our dataset, as we only trained our model on 3,750 short news articles. We firmly believe that given larger amounts of text corpora, our ConHAN model would perform more favorably, as hierarchical attention structures necessitate large amounts of data due to the large complexity (Yang et al., 2016). Nevertheless, this hypothesis needs testing and could be done in future work.

Other potential improvements could be made as a result of the analysis that we have done to interpret model performance. We identify that the model has difficulty predicting authorship for texts that have many conjunctions or geographic terms, so additional parsing could be done to deal with this. More specifically, conjunctions could be removed entirely or replaced with a single common token to minimize their effect on the model. Geographic information could be minimized through training set selection, as one could make an effort to include training texts that talk about many different geographic locations.

Finally, more analysis could be done on our current model results, particularly on the statistical inference side. We only use correlation metrics to determine the impact of training data on our model's performance, but additional statistical analysis could be done to identify more ways in which our model or approach could be improved.

7 Conclusion

In this paper, we took a highly varied approach to authorship identification. We chose to implement our models on the Reuters C50 dataset, a widely-used benchmark for this task. After reviewing relevant literature, we created a baseline random forest model and reproduced results from (Qian et al., 2017) after implementing GRUs at both the sentence and article levels.

Next, we experimented with using the pre-trained transformer DistilBERT and achieved our best results using relatively simple architectures. Using only the DistilBERT [CLS] token as our article representation yielded the best results, achieving an accuracy of 78.5% and an F1 score of 78.34%. We then incorporated all of the DistilBERT word tokens and aggregated them using an attention head to generate our document representations; this model yielded an accuracy of 77.5% and an F1 score of 77.47%.

Our major innovation was implementing hierarchical attention networks in conjunction with DistilBERT. We took contextualized word representations from DistilBERT and fed them into a word attention mechanism to generate sentence representations before feeding these sentence representations into another attention head to generate document representations. With this approach, our model had an accuracy of 75.7% and an F1 score of 75.3%. We iterated upon this model by generating contextualized sentence representations, in the vein of the word representations generated by DistilBERT, using a bidirectional GRU at the sentence level. This final model is the ConHAN model, which had an accuracy of 76.6% and an F1 score of 76.36%.

Finally, after training each of our models and obtaining results, we also did some analysis to determine what our ConHAN model does well with and where it has blind spots. We find that our model performs relatively poorly when predicting authors that have a varied and rich vocabulary, as opposed to those that tend to stick to smaller, more basic terms. Through our analysis, we also find that the models tend not to identify authors via usage of highly specific words, but rather they take a more holistic view of the text content.

We also took a statistical approach to interpreting our models. Using tools to perform named entity recognition, part of speech tagging, and stop word analysis, we produced correlation metrics for certain types of words. The most prominent results of this exercise were that our model tended to perform poorly on texts with many conjunctions as well as geographical terms. These types of words had negative correlations with the ability of our model to make a correct prediction, and these correlations were highly statistically significant.

While our novel ConHAN architecture did not perform the best out of all the models implemented, we think that this architecture has a lot of promise for future work. ConHAN is an extremely flexible and powerful model that estimates many parameters, so we believe that with an augmented training set our model could achieve superior results.

8 Contribution

For this paper, the three authors reviewed the literature and agreed on the overall approach. Jean Bouteiller implemented the first baseline model (Random Forest). Jack Schooley reproduced models and results from (Qian et al., 2017) - namely the Article RNN and Sentence RNN. Victor Jouault designed and implemented the the transfer learning models using DistilBERT, and built on the Hierarchical Attention Networks architecture from (Yang et al., 2016) to create ConHAN, Contextualized Hierarchical Attention Networks for author identification. Finally, Jean Bouteiller evaluated the performances of the models and built the model interpretation framework at the author and article level.

9 Source code

Source code and notebooks to implement this paper can be found in this GitHUB folder.

References

Camilo Akimushkin, Diego R Amancio, and Osvaldo N Oliveira Jr. 2018. On the role of words in the network structure of texts: application to authorship attribution. *Physica A: Statistical Mechanics and its Applications*, 495:49–58.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

C50. Reuters_50_50.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Robin Gomila. 2020. Logistic or linear? estimating causal effects of experimental treatments on binary outcomes using regression analysis. *Journal of Experimental Psychology: General*, pages 1–27.

Margherita Grandini, Enrico Bagli, and Giorgio Visani. 2020. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

John Houvardas and Efstathios Stamatatos. 2006. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*, pages 77–86. Springer.

Shahzad Qaiser and Ramsha Ali. 2018. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181.

Chen Qian, Tianchang He, and Rao Zhang. 2017. Deep learning based authorship identification.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

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. *CLEF (Working Notes)*, 1180:877–897.

Kumiko Tanaka-Ishii and Shunsuke Aihara. 2015. Computational constancy measures of texts—yule's k and rényi's entropy. *Computational Linguistics*, 41(3):481–502.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3):378–393.