Authorship Identification using Recurrent Neural Networks

Shriya TP Gupta
Department of Computer Science
BITS Pilani Goa Campus
Goa, India
shriyatp99@gmail.com

Jajati Keshari Sahoo
Department of Mathematics
BITS Pilani Goa Campus
Goa, India
jksahoo@goa.bits-pilani.ac.in

Rajendra Kumar Roul
Department of Computer Science
Thapar Institute of Technology
Punjab, India
raj.roul@thapar.edu

ABSTRACT

Authorship identification is the process of revealing the hidden identity of authors from a corpus of literary data based on a stylometric analysis of the text. It has essential applications in various fields, such as cyber-forensics, plagiarism detection, and political socialization. This paper aims to use a deep learning approach for the task of authorship identification by defining a suitable characterization of texts to capture the distinctive style of an author. The proposed model uses an index based word embedding for the C50 and the BBC datasets, applied to the input data of article level Long Short Term Memory (LSTM) network and Gated Recurrent Unit (GRU) network models. A comparative study of this new variant of embeddings is done with the standard approach of pre-trained word embeddings.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; *Artificial intelligence*; *Information extraction*;

KEYWORDS

Data Mining, Embeddings, Neural Networks, Text Classification

ACM Reference Format:
Shriya TP Gupta, Jajati Keshari Sahoo, and Rajendra Kumar Roul. 2019. Authorship Identification using Recurrent Neural Networks. In *2019 The 3rd International Conference on Information System and Data Mining (ICISDM 2019), April 6–8, 2019, Houston, TX, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3325917.3325935

1 INTRODUCTION

Deep learning architectures have been a fundamental aspect in creating a paradigm shift in the way we address most classification problems today [6]. Deep learning enables multilevel automatic feature extraction whereas machine learning based Natural Language Processing (NLP) systems depend heavily on hand crafted features that are potentially tedious and often insufficient [14]. Moreover, machine learning models suffer from the curse of dimensionality mainly because linguistic information is represented using high-dimensional and sparse features [4]. Owing to the recent success and popularity of word embeddings, neural networks have obtained very high performance across various text related tasks as compared to the traditional machine learning models.

Deep learning approaches were designed mainly for feed-forward networks such as Convolutional Neural Networks (CNN) which are effective for analyzing data samples of fixed length such as images. However, Recurrent Neural Networks (RNN) showed promising results on variable length data such as natural language text, and the use of these models are gradually increasing [16]. They mainly deal with the syntactic and semantic processing of sequential information and assume all inputs to be independent. RNNs are well suited for tasks like language modelling, machine translation, speech recognition, multilabel text categorization, and multimodal sentiment analysis.

Automating the process of authorship identification guarantees accurate and reliable results, both of which are integral for legal and defense related applications. Distinguishing the author of a text is a real and recurring problem that appears across different fields. Archaeologists and chroniclers regularly retrieve literary works and attempt to discern unattributed texts. Academicians are persistently on the lookout for plausible cases of plagiarism. Thus, the authorship identification task can be viewed as a multiclass classification problem of a high dimensional feature space and naturally RNNs seem to be an appropriate choice for the task [2].

The main contributions of this work are as follows: 1) A novel word index based embedding approach as part of the preprocessing step which makes the subsequent classification task more efficient. 2) Use of variants of the RNN model to improve the text classification for different datasets. 3) Appropriate use of different optimization techniques during the training phase to make the neural network models more stable and accurate. The BBC and C50 datasets have been used for carrying out the experimentation and the proposed approach shows promising results. The rest of this paper is organized as follows: Section 2 covers the literature review for our work and Section 3 describes the mathematical background of the various models used. In Section 4, the detailed implementation of the neural networks has been described and experimental results are covered in Section 5. Finally, conclusions and recommendations for further development have been presented in Section 6.

2 RELATED WORK

In the context of neural networks, embeddings are useful for meaningfully representing data in the transformed space as they efficiently reduce the dimensionality of categorical variables. Levy *et. al.* [7] suggested modifications to the initial bag-of-words approach like the dependency based context for word embeddings. Earlier work by G. Salton *et. al.* [13] proposed the assignment of weights

ICISDM 2019, April 6–8, 2019, Houston, TX, USA

ACM ISBN 978-1-4503-6635-9/19/04…$15.00
https://doi.org/10.1145/3325917.3325935

to words by the term frequency-inverse document frequency (TF-IDF) method. It is based on calculating the weight of a word in a document using an inverse proportion on the frequency of the word in that document to the percentage of documents the word is present in. Recently, the word2vec representation developed by T. Mikolov *et. al.* [9] uses unsupervised learning to determine semantic and syntactic meaning from word co-occurrence, which solves the problem of representing contextual word relationships in a feature space. The learned word vectors are aligned in the vector space such that the words with common context have close proximity. Similar pre-trained word embeddings called Global Vectors for Word representation (GloVe) by J. Pennington *et. al.* [11] learn word vectors by using co-occurrence probabilities of words in a vector space. The resulting vectors accurately capture the similarity between related words.

Deep learning frameworks have outperformed most state-of-the-art approaches in a multitude of language processing tasks such as machine translation, sentiment analysis, and speech recognition as suggested in T. Young *et. al.* [16]. Recent works like those by A. Mohsen *et. al.* [10] include a combination of different deep learning models. They propose feature extraction using autoencoders and then performing the author identification using a Support Vector Machine (SVM) classifier. CNNs for sentence classification as introduced by Y. Kim *et. al.* [5] also follow a similar approach. Further, S. Ding *et. al.* [1] examine the different stylometric models for these networks which allow extraction of feature vectors of a document in a diverse manner such as the topical or character level.

In C. Qian *et. al.* [12], RNNs have been explored for the similar datasets to achieve considerable accuracies and they use pre-trained GloVe embeddings. However pre-trained embeddings limit the performance of the model, owing to their high dimensionality. Further, they do not consider the possibilities where the word usage matters as well. Because these word vectors are meant to capture similarities between words, certain words such as 'therefore' and 'furthermore' will have very similar word vectors. However, as pointed out by L. Yao *et. al.* [15], these specific word choices could determine an author's style.

3 PRELIMINARIES

The models used in this work are RNNs, which are suitable for processing sequential information. RNNs are quite distinctive when compared to a standard neural network because they can make use of data in arbitrarily overlong sequences, while allowing the captured information to persist and be passed on from one iteration to the next. The network takes $x = (x_1, x_2, ..., x_T)$ as input, and iterates over it from t=1 to T as follows:

$$h_t = \sigma(b_{h_t} + W_{xh}x_t + W_{hh}h_{t-1}) \tag{1}$$

where $h = (h_1, h_2, ..., h_T)$ is the hidden state vector at a particular time step and W_{xh} represents the weight matrix between the input and the hidden state. The b terms in Eq.1 represent the bias vectors of the hidden layer. The activation function σ could be a non-linear tanh or sigmoid function, depending on the task.

3.1 Long Short Term Memory Network

LSTMs are a unique variant of recurrent networks that explicitly deal with the vanishing gradient problem faced in traditional RNNs. LSTMs are suitably designed to classify time series data and are relatively insensitive to gap length unlike RNNs. They have a similar chain like structure but are composed of different repeating units with the neural network layers interacting in the way shown in Figure 1.

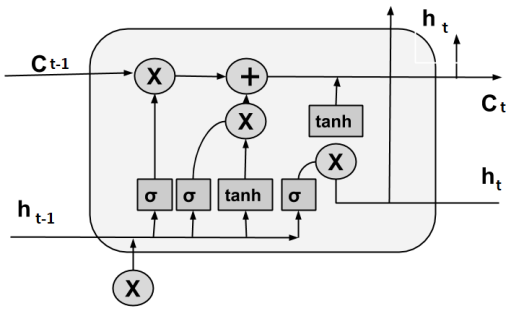


Figure 1: LSTM network

The cell equation for the LSTM model is given in Eq.2 where C_t, i_t, f_t and o_t are the cell memory, input, forget and output gates respectively at time step t, and W is the corresponding weight matrix. At time t, x_t is used as the input to the memory cell in its initial state h_{t-1}. Here C_t is the updated cell memory configuration with the final state h_t.

$$\begin{cases} \tilde{C_t} & = \tanh(W^c[x_t, h_{t-1}] + b^c) \\ C_t & = f_t \circ C_{t-1} + i_t \circ \tilde{C_t} \\ i_t & = \sigma(W^i[x_t, h_{t-1}] + b^i) \\ o_t & = \sigma(W^o[x_t, h_{t-1}] + b^o) \\ f_t & = \sigma(W^f[x_t, h_{t-1}] + b^f) \\ h_t & = o_t \circ \tanh(C_t) \end{cases} \tag{2}$$

3.2 Gated Recurrent Unit Network

An improved version of the LSTM is the GRU network which modifies the forget and input gates to form a single update gate and an associated reset gate which collectively decide the information that is passed to the output. As shown in Figure 2, it also combines the hidden state and cell state, leading to a simpler but more powerful model than the standard LSTM.

The cell equation for the GRU model is given in Eq. 3 where z_t and x_t stand for update vector and input vector at time t. Also, h_t and r_t are the output and reset gate vectors respectively and U is the weight matrix.

$$\begin{cases} z_t & = \sigma(W^z[x_t, h_{t-1}] + b^z) \\ r_t & = \sigma(W^r[x_t, h_{t-1}] + b^{(r)}) \\ \tilde{h_t} & = \tanh(r_t \circ U^h h_{t-1} + W^h x_t + b^h) \\ h_t & = (1 - z_t) \circ \tilde{h_t} + z_t \circ h_{t-1} \end{cases} \tag{3}$$

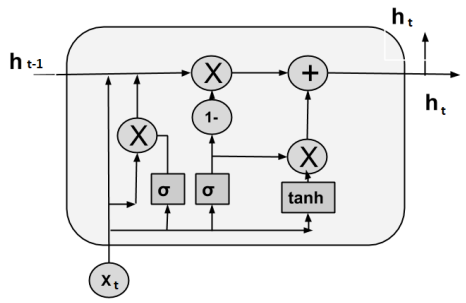


Figure 2: Unit cell of GRU

4 METHODOLOGY

The methodology section of the paper begins with a description of the datasets used for the task and the preprocessing of the input data. Next, the implementation of the neural networks is explained and the optimization method applied to the models.

4.1 Dataset

The Reuters_50_50 (C50) dataset is a subset of the Reuters Corpus Volume I (RCV1) which is a sizable collection of nearly 800,000 manually labelled press agency articles recently made accessible by Reuters, Ltd. for experimentation [8]. From those, the top 50 authors (based on total length of text) were chosen. The corpus consists of 2,500 texts i.e. 50 per author and is usually used in research related to authorship identification.

The BBC dataset is a newswire dataset, derived from BBC News, offered for use as reference for machine learning experiments [3]. It is comprised of 2225 articles from the BBC website, associated to reports of five topical areas ranging from 2004-2005. The five themes are sports, business, politics, entertainment, and technology.

4.2 Pre-Processing

In this work, the word tokens of the input data are converted to an integer format with the GloVe look-up table by using the corresponding word indices instead of the word vectors themselves. The Glove vocabulary contains 400,000 tokens and their pre-trained distributed word vectors. During the training phase, the gradients are propagated to this embedding layer. A representative vector mapping of a sentence is given as $v_k = (w_{1,k}, w_{2,k} ... w_{i,k} ... w_{l_k,k})$ where l_k is the size of sentence k. Here $w_{i,k}$ is the index of the GloVe embedding vectorization of word i in sentence k.

In order to make use of the parallel computing advantage of the processor, we adopted a batch input which further helped to accelerate the training process of our model. Since the batch has a fixed length, the input is truncated if it exceeds the specified length.

4.3 Recurrent Neural Networks

This paper uses an article-level GRU which is based on a model that takes entire paragraphs of the article as the input, and predicts its author. In this model, the input to each sub-unit is a sentence as shown in Figure 3 . A complete sentence is characterized by the concatenation of its constituent word vectors. Firstly, the network is initialized using a Xavier initializer which ensures that the starting weights are assigned a precise value and keeps the variance of the weights in an acceptable range across many layers. For an input X with n attributes and a linear neuron with arbitrary weights W, we can work out the variance of output Y if we allow another presumption that the X_i and W_i are all similar and statistically independent. By Eq.4 we can take the variance as $Var(W_i) = 1/n$ where n specifies the number of input neurons.

$$\begin{cases} Var(Y) = Var(W_1X_1 + W_2X_2 + ... + W_nX_n) \\ \therefore Var(Y) = nVar(W_i)Var(X_i) \end{cases} \tag{4}$$

For every time stamp, the input vector is sent to the model while the hidden state is updated and some output is produced. The equation for the hidden state, update and output computation is given in Eq.3. Here h_t represents both the new hidden state and the output. After no more input exists, all outputs at previous time stamp are sent to an average pooling layer, whose output is taken as the average of all inputs. The output of the pooling layer will be used as the input to a softmax classifier. For all set of pairs (x,y) such that x∈X and y∈Y, Eq.5 gives the class probabilities for each class label. Based on this, the final decision that which author wins is determined.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \forall j \in 1, ..., K \tag{5}$$

Next, the cross entropy loss as given in Eq.6 is calculated, which evaluates the classification performed by the model. The magnitude of the loss increases as the predicted probability deviates from the true probability for the actual label. For this multilabel classification task, we estimate an individual loss associated with each label and then take the summation. Here, x represents a binary value (indicates whether class label v assigned to an observation o is the correct value), p is the predicted probability of an observation o belonging to class value v and N is the number of classes.

$$\sum_{v=1}^{N} x_{o,v} log(p_{o,v}) \tag{6}$$

Finally, the Backpropagation Through Time (BPTT) algorithm is employed to determine the appropriate network weights. The errors are calculated for each timestep and accumulated to derive the gradients. Then the entire network is rolled back and the weights and biases are updated as a whole. The block diagram summarizing the working of the article level GRU is shown in Figure 3.

The main idea beneath this work is that the context information as well as the word sequence information is captured by the model, at an article level during the training phase. Article-level LSTM is nearly identical to the Article-level GRU, having the same type of input and data representation, including the structure of the model. The only difference lies in the cell; for the LSTM model, we use Eq.2 instead of Eq.3.

4.4 Optimization

In this work we have used the Adaptive Moment Estimation (Adam) optimizer which iteratively updates the network weights similar to the classical stochastic gradient descent method. It maintains a

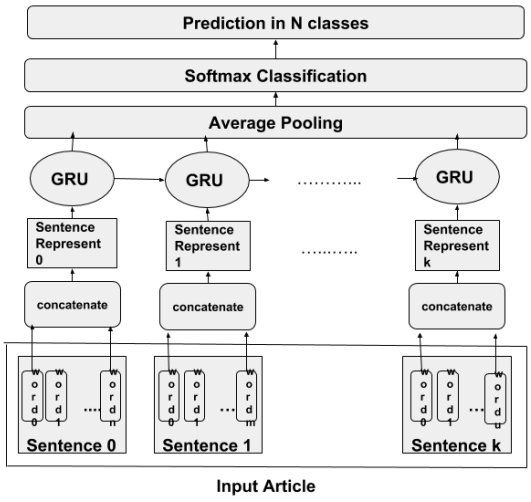


Figure 3: Article level GRU model

learning rate for each parameter by calculating an exponentially decaying average of the gradient m_t and the squared gradient v_t. We first compute the biased estimates of the moving averages for the gradients m_t and v_t as in Eq.7 and then the bias-corrected values.

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{cases} \tag{7}$$

where β_1 and β_2 are the decay rates. Adam is also computationally effective and comparable to other modern methods.

5 EXPERIMENTS AND RESULTS

This section discusses the performance evaluation metrics, the frameworks used, and the results of the experiments carried out. Finally, the variation in accuracies due to hyperparameter tuning is summarized.

5.1 Performance Measure

In this work, we report the accuracy obtained on the various datasets along with their corresponding confusion matrices. As the classes were balanced, we used the micro F1-score for evaluation as given in Eq.8. It ranges over 0 to 1 attaining its best value at 1.

$$\begin{cases} Precision = \frac{\Sigma TP_i}{\Sigma(TP_i+FP_i)} \\ Recall = \frac{\Sigma TP_i}{\Sigma(TP_i+FN_i)} \\ F1-score = \frac{2*Precision*Recall}{Precision+Recall} \end{cases} \tag{8}$$

where TP_i, FP_i, FN_i, and TN_i is the count of true positives, false positives, false negatives and true negatives respectively for the i^{th} class of the multiclass classification problem.

5.2 Frameworks

The experiments were carried out on Google Colaboratory, Google's free cloud service for developing deep learning applications on its GPU facilities. The work was implemented in Python, and the Tensorflow library was used for creating the neural networks.

5.3 Empirical Results

In the first case, the C50 corpora was split into a 9:1 (train:test) split ratio. 90% of this data was used for the training phase of the model and the remaining 10% of the data was used for the evaluation. Here, Table.1 depicts the results obtained for the C50 dataset using the word-index based embedding on the input layer. Table.2 summarizes the results for the models using the GloVe embedding vectors for the inputs, analyzed on the same dataset.

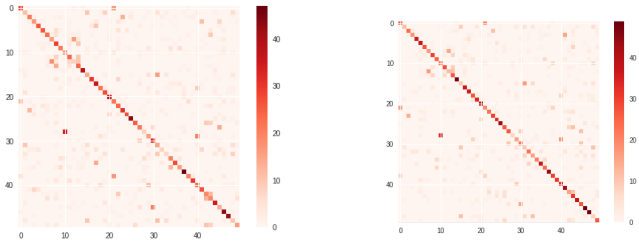
Table 1: Proposed word-index embeddings (C50 dataset)

Scenario	LSTM	GRU
Test Accuracy	66.67%	78.1%
Train Accuracy	98.2%	100.0%

Table 2: Pre-trained GloVe word embeddings (C50 dataset)

Scenario	LSTM	GRU
Test Accuracy	61.47%	69.2%
Train Accuracy	98.33%	100.0%

The results for the LSTM network is shown in the confusion matrix of Figure.4a and for the GRU network in Figure.4b which are based on the word index embedding. The proposed model attains a best accuracy of 78.1%, an improvement over the 69.2% attained using pre-trained embeddings as reported in [12].



(a) LSTM Network **(b) GRU Network**

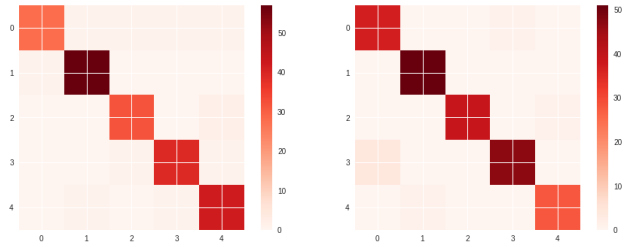
Figure 4: Results for the C50 dataset

Similarly in the second case, the results were analyzed for the BBC dataset with a 9:1 (train:test) split ratio and the model achieves an accuracy of upto 96.7% as shown in Table.3.

As observed in both the cases, the GRU network performs better as compared to the LSTM network. The corresponding results for the LSTM network are shown in Figure.5a and for the GRU network in Figure.5b.

Table 3: Proposed word-index embeddings (BBC dataset)

Scenario	LSTM	GRU
Test Accuracy	94.73%	96.65%
Train Accuracy	100.0%	100.0%



(a) LSTM Network **(b) GRU Network**

Figure 5: Results for the BBC dataset

5.4 Hyperparameter Tuning

As seen from the results above, the models suffer from considerable overfitting because the model has few inputs and many parameters. Applying dropout lead to an ensemble type approach and helped reduce the overfitting to some extent. Increasing the learning rate from 0.000 to 0.004 caused a faster convergence of the weights. Further, varying the hidden layer size from 50 to 300 lead to a monotonous increase in accuracy due to better learning capacity of network. The results show the improvement in accuracy with increase in number of epochs for the GRU network on C50 dataset. These figures show the variation in hidden layer size from 200 to 300 with a marked improvement in accuracies. The testing data results are summarized in Figure.6 and the training data results in Figure.7.

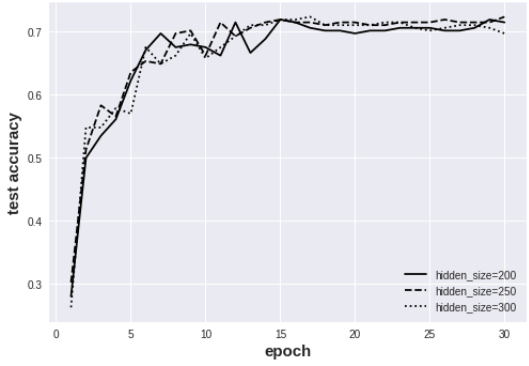


Figure 6: Variation in test accuracy

6 CONCLUSION

A comparative view of the different RNN models on the authorship identification task indicates that the GRU model performs remarkably better than the LSTM model due to its enhanced learning capacity. Besides, the index based embedding outperforms the pre-trained embeddings as it scales well for specific datasets. Future directions would include exploring variants of RNN for the author identification tasks and using different distributed word representations so as to enhance the performance of the neural networks currently explored. The current models can also be tried out on larger datasets for better weight updates of the network.

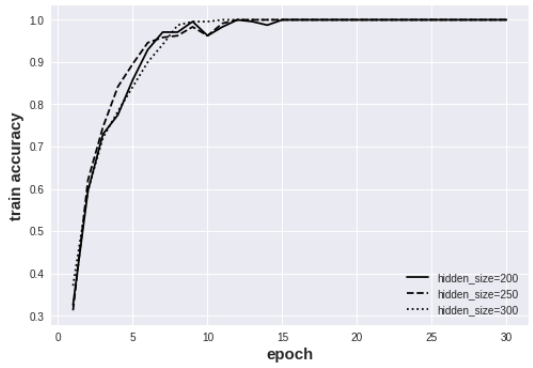


Figure 7: Variation in train accuracy

REFERENCES

[1] Steven HH Ding, Benjamin CM Fung, Farkhund Iqbal, and William K Cheung. 2017. Learning Stylometric Representations for Authorship Analysis. *IEEE Transactions on Cybernetics* (2017).

[2] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[3] Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 377–384.

[4] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[5] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[7] Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 302–308.

[8] Zhi Liu. 2017. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Reuter_50_50

[9] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 746–751.

[10] Ahmed M Mohsen, Nagwa M El-Makky, and Nagia Ghanem. 2016. Author identification using deep learning. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*. IEEE, 898–903.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[12] Chen Qian, Ting He, and Rao Zhang. 2017. Deep Learning based Authorship Identification.

[13] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[14] Oliver G Selfridge. 1958. Pandemonium: A paradigm for learning. *the Mechanisation of Thought Processes, 1958* (1958).

[15] Leon Yao and Derrick Liu. 2015. Wallace: Author Detection via Recurrent Neural Networks.

[16] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine* 13, 3 (2018), 55–75.