Vol. 00 no. 00
Pages 1–5

Author Identification : A Deep Approach and Comparative Study

Anand Pandey [1], Ankit Pensia [2],

[1]Department of Computer Science and Engineering,IIT Kanpur
[2]Department of Electrical Engineering,IIT Kanpur

ABSTRACT

Author identification has been one of the classical problem in Natural Language Processing. In this project, we tried different architectures of neural network to identify authors of the text. Each architecture has different requirements for the format of input and output data. Each architecture has its own merits and downside. The task of Author identification can be transformed into a multi-class classification problem given a vector representation of the document which contains all the information needed for author-identification. The objective is to effectively learn this vector-representation of the document.

Keywords: Author Identification, Deep Learning, Natural Language Processing, Tree-LSTM

1 INTRODUCTION

In this project, we wish to explore the problem of author identification in the field of Natural Language Processing. This problem has been studied extensively using hand-designed features. Hand-designed features require domain expertise and great insight of the problem. We want to explore how deep learning can be used to learn the abstract and higher-level features of the document, which could be used to identify author. We wish to explore several variants from deep architecture, which could be used to tackle this problem. Although the title contains the word ”deep”, the models used had single layer for the lack of computational power and large data. Layers could be stacked to produce deep architectures.

Problem statement can be formalised as - Given a text document and a set of authors, learn a function that maps the document to a single author. The training data includes the documents, labelled with their authors.

2 PREVIOUS WORK

A lot of features have been suggested based on n-grams, characters, grammar to identify authors[5][1][3]. Many of these features have been designed by experts of the field. Such features quantify the use of punctuation, sentence structure, vocabulary in the given document. The philosophy behind such features is that these features don’t change much for a specific author across his writings. There has been a recent revival of interest in using deep learning methods for various NLP problems[4], in order to learn more robust features using easily available large data.

For the task of author identification, we came across recent project works which made use of neural network architecture for author attribution which included Long Short-Term Memory, Convolutional Neural Network and Recursive Neural Network[6][7][8]. All of these architectures were tested on very different type of datasets and a comparison couldn’t be drawn. In this project, we have used LSTM and have tried two new architectures for author attribution including Tree-LSTM and Paragraph Vector.

3 DATASET GENERATION

Neural Networks require large amount of data to learn the problems. This is so because as the model architecture grows in complexity and size, the number of parameters also explode.

We required documents with considerable number of words and containing semantically rich sentence. The documents should also have been written by people reflecting their distinct writing style. Copied content shouldn’t appear in dataset, both training and testing as it defeats the purpose of author identification.

Quora is a Qu(estion) or A(nswer) platform where people ask questions belonging to wide variety of domains. These questions are answered by people, usually by those who have expertise in that domain. The answers written by Quora Top-Writers do satisfy aforementioned desired properties. The dataset was generated by using Quora RSS feed. Each answer was treated as a different document. We collected around 1700 answers from 47 Quora Top Writers. All these answers had minimum word length. Such a filter was used so that the document (each answer) should reflect author’s writing style.

Number of Authors	47
Total Answers	1732
Vocabulary Size	46804
Total Words	723502

4 ARCHITECTURES USED

4.1 Long Short-Term Memory

Long Short Term Memory Networks[11], or LSTMs have been very successful in modelling sequences. They have provided a significant increase in our ability to remember the history in sequential data. In the context of natural languages, they have proved to be a great leap in abstract representation of sentences. LSTMs allow us to model

© .

text as a sequence of words. Many state-of-the-art benchmarks in Natural Language Processing currently use LSTM or its variants.

The effectiveness and wider applications of LSTMs compared to traditional Recurrent Neural Network can be attributed to the fact that the addition of gates enable LSTMs to solve Vanishing or Exploring Gradient problem prevalent in Recurrent Neural Networks while using Back Propagation Through Time (BPTT).

The equations of LSTM are:-

$$
\begin{aligned}
i_t &= \sigma(W^{(i)}\mathrm{x}_t + U^{(i)}\mathrm{h}_{t-1} + \mathrm{b}^{(i)}) \\
f_t &= \sigma(W^{(f)}\mathrm{x}_t + U^{(f)}\mathrm{h}_{t-1} + \mathrm{b}^{(f)}) \\
o_t &= \sigma(W^{(o)}\mathrm{x}_t + U^{(o)}\mathrm{h}_{t-1} + \mathrm{b}^{(o)}) \\
u_t &= tanh(W^{(u)}\mathrm{x}_t + U^{(u)}\mathrm{h}_{t-1} + \mathrm{b}^{(u)}) \\
c_t &= i_t \odot u_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

As the sequence length increases, the error doesn't propagate back to the start of sentence. So, the hidden vector at the end of sequence is not able to capture the context of sequence completely. Several answers were longer than a conventional page. So, we first break the answer into chunks of fixed maximum size. To do so, we first split each document into sentences. The consecutive sentences are then grouped together till the combined number of words exceed a word-limit size.
For training and testing purposes on LSTM, each chunk with its author-label was treated as separate example. For this experiment, we chose the max-length of sequence to be 150. This parameter was not tuned.

Theoretically, the hidden vector of at the last time-step of sequence should be able to capture every information of sentence and if a classifier is trained on this fixed-size vector, it would be able to predict author completely. But as told earlier, LSTMs sometimes suffer from the problem of having memory of finite-length past. To overcome this, the classifier was trained on the average of hidden-representation of every node. Instead of simple average, any differentiable function of hidden-representations of every time-step can be used.

Experiment results are given at the end.

4.2 Tree-LSTM

LSTMs model each sentence as a causal system, with the assumption that the future depends only on the past(captured using hidden vector) and the present (input). But as we know, the sentences have more structure than a simple Causal linear relation. Although, Bidirectional LSTM try to capture anti-causal relation, but a more natural and comprehensive structure of a sentence is given by the parse tree of the sentence.

Tree-LSTMs are a recursive neural network, first proposed in 2015 in [2]. In [2], author makes use of the parse-tree by using the recursive neural network architecture.

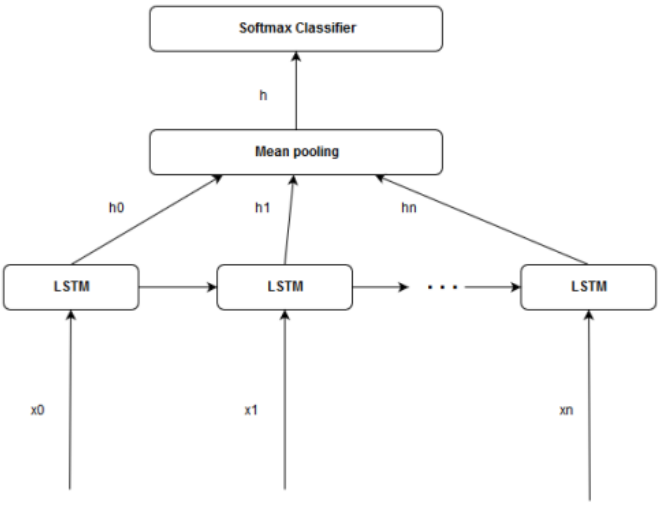

Fig. 1. Model used for training LSTM for author identification.

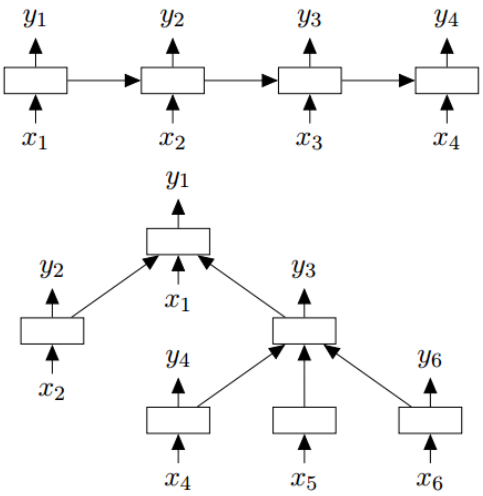

Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.

Fig. 2. Tree-LSTM, as shown in [2]

Author proposes two variants of Tree-LSTM. We have used the 'Constituency Tree-LSTMs' which handles the left and right child of a node differently (for binary tree).
The equations of Tree-LSTM are :-

$$
i_j = \sigma(W^{(i)}\mathrm{x}_j + \sum_{l=1}^{N} U_l^{(i)}\mathrm{h}_{j^l} + \mathrm{b}^{(i)})
$$

$$f_{jk} = \sigma(W^{(f)}\mathrm{x}_j + \sum_{l=1}^{N} U_{kl}^{(f)}\mathrm{h}_{jl} + \mathrm{b}^{(f)})$$

$$o_j = \sigma(W^{(o)}\mathrm{x}_j + \sum_{l=1}^{N} U_l^{(o)}\mathrm{h}_{jl} + \mathrm{b}^{(o)})$$

$$u_j = tanh(W^{(u)}\mathrm{x}_j + \sum_{l=1}^{N} U_l^{(u)}\mathrm{h}_{jl} + \mathrm{b}^{(u)})$$

$$c_j = i_j \odot u_j + \sum_{l=1}^{N} f_{jl} \odot c_{jl}$$

$$h_j = o_j \odot tanh(c_j)$$

The obvious requirement for using Tree-LSTM is a parser. As suggested by author and keeping the size of data, time and computational power in mind, we are using Binarized version of parse trees. We split each document into sentences and each sentence is labelled with the same author-label as the original document. Each sentence thus becomes a separate example, for training or testing purpose.

After forward propagation through Tree-LSTM, we will have a hidden vector representation of the sentence at the root node of the tree. A Softmax Classifier was trained on hidden vector of the root node. As the number of matrices have increased compared to the LSTM model, the number of parameters have also increased rapidly. For a recursive Neural Net, the parameters converge to a good optima if at each node(can be thought of as neural network at each time step) has a label to achieve. For author attribution task, only the sentences have label and thus, the error gradient decreases. It leads to the slow convergence (if any) of the parameters of model.

4.3 Paragraph Vectors

Bag-of-words feature have two major weaknesses: they lose the ordering of the words and they also ignore semantics of the words. In Word2vec[10] also, after learning the word-embeddings, the paragraph embedding is not obvious. To overcome this, Paragraph Vector model was proposed in [9].

As proposed in [9], the paragraph-id is also fed to neural network, and corresponding vector is also learnt while training the word-vectors of the corpus.

The paragraph vector is then representative of the whole paragraph and could be used to find similarity among other paragraphs. Our idea is to check whether we are able to learn the vector representation of each author and use some similarity metric to identify author.

To use it for the task of author identification, in place of paragraph-id, we provide the author-label of the document. The training process will then learn a vector-representation for each author-label and vector-representation for each word. The optimization task is same as that of word2vec[10] and paragraph vector [9], which is to correctly predict the word, given its context words and author-vector.

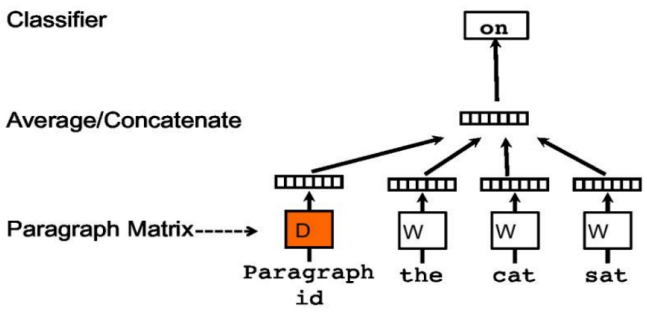

Fig. 3. Paragraph Vector, as proposed in [9]

As mentioned in [9], while testing, the author-vector for document is initialised randomly. Then keeping the word-embeddings fixed, the author-vector is allowed to change for the same prediction problem, i.e. predicting the word given its context-word and author-vector. Once author-vector is obtained (i.e. after convergence), its similarity with the existing authors is checked using cosine similarity. The author-label corresponding to the most-similar author-vector is then reported as prediction.

5 EXPERIMENT AND RESULTS

Implementation of LSTM was done in theano, by modifying the tutorial code given for LSTM on theano official website[12][13]. The paragraph-vector implementation was available in gensim[15]. For tree-LSTM, the code was modified slightly for our purpose, taking the implementation provided by [14] as underlying base.

5.1 LSTM

As mentioned earlier, for training and testing of LSTM, each example was a chunk of original document. The word-embeddings of words were initialised randomly as opposed to using word2vec embeddings as initialization. Random initialization was done because similar words are close in vector representation of word2vec but such a property is not desired here. Instead, the use of different words (although similar in sense) to convey the same information, is one of the key feature to identify authors.

A vocabulary was learnt on training set, prior to training LSTM. The size of vocabulary was fixed to 10,000 keeping the computational power in mind. For LSTM, input was sequence of word-indices, which represented the words with respect to the vocabulary. Each input represented a chunk of document. The maximum length of such sequence was set to 150 with the batch size of 16 examples. The dimensions of word-embedding and the hidden vector were both set to 100. The accuracies are with respect to the chunks of size 150.

Dataset	Top-1 Accuracy	Top-5 Accuracy
Train	0.929334011	0.954289905
Valid	0.285024155	...
test	0.274973712	0.542060988

5.2 Tree-LSTM

Tree-LSTM is most effective when input is a sentence instead of a pragragraph. So, for Tree-LSTM, each document was split into sentences. Each sentence was labelled with the author-label of original document. Each labeled sentence was then a separate example. Stanford Parser was used to generate the parse-trees of each example (i.e. sentence). The tree was then binarized so that the binary architecture model could be used. Each node in tree then had either two,one or no children. Very large sentences were ignored due to parsing constraints and errors.

The reported accuracies are with respect to each sentence.

Dataset	Top-1 Accuracy
Train	0.197
Test	0.182

5.3 Paragraph-Vector

The pre-processing required for training paragraph-vector was minimal. Every document was given a paragraph-id which was the same as that of author-label. The standard implementation of paragragph-vector was then run which after convergence, learnt the word-embeddings for each word and author-embedding for each author.

At inferring stage, the model returned a vector-embedding of the document, representing in this problem-setting, a vector-representation of author. Its similarity with the trained author-representations were checked and most similar author-id reported.

Dataset	Top-1 Accuracy	Top-5 Accuracy
Train	0.038106236	0.123556582

6 CONCLUSION

For our limited dataset, LSTM architecture outperformed other models for our problem. Tree-LSTM suffered from lack of large data and vanishing gradient (only root node had a label). With the addition of more data, we think Tree-LSTM will be able to learn the grammatical style of author better. Our hypothesis that we could learn an author-embedding was not applicable.

- The reason why we think Tree-LSTM was not able to outperform LSTM was that only the root node had a label. The errors vanished when back propagated through time after certain node-depth. A heuristic to replicate the root label to few of the children can be tried.
- For Paragraph-Vector, the unexpectedly bad performance could be attributed to the absence of a penalty term for wrongly classifying authors in the loss function. The parameters were tuned for the problem of correctly classifying the word, given context words. For the problem of fixed-set of authors, instead of appending paragraph-vector in the input of neural-network as proposed in [9], predicting the author in output while simultaneously learning the word-embedding could be tried.

7 FUTURE WORK

With the exciting results shown by LSTM and Tree-LSTM without tuning many hyper-parameters and being trained on only a small dataset, the availability of large dataset and proper-training of models show a great promise in improving result. Apart from these, the key ideas (the reason why present models didn't work or the way they could be improved are) which could be tried in future are

- Replicating the root label to some of children using a heuristic.
- Instead of only taking paragraph-vector as an input, the paragraph vector should be given as output with a large context window, with a soft-max classifier on top of it to predict author directly. The end-to-end neural network can then be trained and should outperform the current paragraph-vector model.
- To see how real deep architecture would work, LSTMs could be stacked on top of each-other.

8 ACKNOWLEDGEMENT

We thank Prof. Amitabha Mukherjee for his valuable support and consistent guidance throughout the project.

9 REFERENCES

[1] Green, R. M., Sheppard, J. W. Comparing Frequency- and Style-Based Features for Twitter Author Identification (2013, May). Twenty-Sixth International Florida Artificial Intelligence Research Society Conference :64-69

[2] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree- structured long short-term memory networks.(2015),ACM Computing Research Repository Vol: abs/1503.00075

[3] Patrick Juola. Authorship attribution. Foundations and Trends in Information Retrieval , 1(3):233334, 2006

[4] Ronan Collobert, Jason Weston, L eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. The Journal of Machine Learning Research , 12:24932537, 2011.

[5] Stamatatos, E. (2009), A survey of modern authorship attribution methods. J. Am. Soc. Inf. Sci., 60: 538556. doi: 10.1002/asi.21001

[6] Pranav Jindal, Ashwin Paranjape: Deanonymizing Quora Answers (2015).CS224,Stanford University

[7] Stephen Macke, Jason Hirshman: Deep Sentence-Level Authorship Attribution (2015). CS224,Stanford University

[8] Dylan Rhodes: Author Attribution with CNNs(2015). CS224,Stanford University

[9] Quoc Le, Tomas Mikolov ;Distributed Representations of Sentences and Documents arXiv:1405.4053

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of phrases and their compositionality. In NIPS.

[11] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. Addition of the forget gate to the LSTM model

[12] Bastien, Frdric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2012.

[13] Bergstra, James, Breuleux, Olivier, Bastien, Frdric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010.

[14] Shenxiu Liu, Qingyun Sun: Conquering vanishing gradient: Tensor Tree LSTM on aspect-sentiment classification.(2015) CS224,Stanford University

[15] EHEK, Radim and Petr SOJKA. Software Framework for Topic Modelling with Large Corpora. In Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. Valletta, Malta: University of Malta, 2010. p. 46–50, 5 pp. ISBN 2-9517408-6-7.