INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Statistical Methods in AI

PROJECT-TEAM 37

Syntactic Recurrent Neural Network for Authorship Attribution

Author:	*Student Number:*
Danish Mukhtar Zargar	2018201016
Sivangi Singh	2018201001
Monu Tayal	2018201042
Giridhari Lal Gupta	2018201019

April 29, 2019

Contents

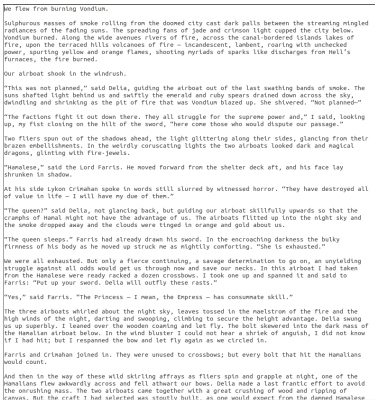
1 Introduction

Writing style is a combination of consistent decisions at different levels of language production including lexical, syntactic, and structural associated to a specific author.While lexical-based models have been widely explored in style-based text classification, relying on content makes the model less scalable when dealing with heterogeneous data comprised of various topics.On the other hand, syntactic models which are content-independent, are more robust against topic variance.
Stylistic features are generally content-independent which means that they are mainly consistent across different documents written by a specific author or author groups. Lexical, syntactic, and structural features are three main families of stylistic features. Lexical features represent author's character and word use preferences,while syntactic features capture the syntactic patterns of sentences in a document. Structural features reveal information about how an author organizes the structure of a document.

One of the basic problems which is rarely addressed in the literature is the interaction of style and content. While content words can be predictive features of authorial writing style due to the fact that they carry information about authors lexical choice, excluding content words as features is a fundamental step for avoiding topic detection rather than style detection . However, syntactic and structural features are content-independent which makes them robust against divergence of topics
The adopted approaches in deep neural network for style-based text classification mainly focus on lexical features despite the fact that lexical-based language models have very limited scalability when dealing with dataset containing diverse topics and genre.While previously proposed deep neural network approaches focus on lexical level, we introduce a syntactic recurrent neural network which hierarchically learns and encodes the syntactic structure of documents.

2 DataSet

We used Pan 2012 dataset URL : https://pan.webis.de/data.html
The Given DataSet (Novels and their Writers)is of the form of text files-



We flew from burning Vondium.

Sulphurous masses of smoke rolling from the doomed city cast dark palls between the streaming mingled radiances of the fading suns. The spreading fans of jade and crimson light cupped the city below. Vondium burned. Along the wide avenues rivers of fire, across the canal-bordered islands lakes of fire, upon the terraced hills volcanoes of fire — incandescent, lambent, roaring with unchecked power, spurting yellow and orange flames, shooting myriads of sparks like discharges from Hell's furnaces, the fire burned.

Our airboat shook in the windrush.

"This was not planned," said Delia, guiding the airboat out of the last swathing bands of smoke, The suns shafted light behind us and swiftly the emerald and ruby spears drained down across the sky, dwindling and shrinking as the pit of fire that was Vondium blazed up. She shivered. "Not planned—"

"The factions fight it out down there. They all struggle for the supreme power and," I said, looking up, my fist closing on the hilt of the sword, "here come those who would dispute our passage."

Two fliers spun out of the shadows ahead, the light glittering along their sides, glancing from their brazen embellishments. In the weirdly coruscating lights the two airboats looked dark and magical dragons, glinting with fire-jewels.

"Hamalese," said the Lord Farris. He moved forward from the shelter deck aft, and his face lay shrunken in shadow.

At his side Lykon Crimahan spoke in words still slurred by witnessed horror. "They have destroyed all of value in life — I will have my due of them."

"The queen!" said Delia, not glancing back, but guiding our airboat skillfully upwards so that the cramphs of Hanal might not have the advantage of us. The airboats flitted up into the night sky and the smoke dropped away and the clouds were tinged in orange and gold about us.

"The queen sleeps." Farris had already drawn his sword. In the encroaching darkness the bulky firmness of his body as he moved up struck me as mightily comforting. "She is exhausted."

We were all exhausted. But only a fierce continuing, a savage determination to go on, an unyielding struggle against all odds would get us through now and save our necks. In this airboat I had taken from the Hamalese were ready racked a dozen crossbows. I took one up and spanned it and said to Farris: "Put up your sword. Delia will outfly these rasts."

"Yes," said Farris. "The Princess — I mean, the Empress — has consummate skill."

The three airboats whirled about the night sky, leaves tossed in the maelstrom of the fire and the high winds of the night, darting and swooping, climbing to secure the height advantage. Delia swung us up superbly, I leaned over the wooden coaming and let fly. The bolt skewered into the dark mass of the Hamalian airboat below. In the wind bluster I could not hear a shriek of anguish, I did not know if I had hit; but I respanned the bow and let fly again as we circled in.

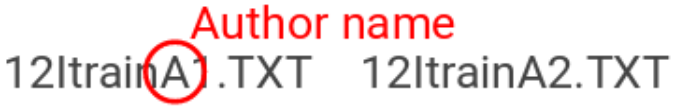
Farris and Crimahan joined in. They were unused to crossbows; but every bolt that hit the Hamallans would count.

And then in the way of these wild skirling affrays as fliers spin and grapple at night, one of the Hamallans flew awkwardly across and fell athwart our bows. Delia made a last frantic effort to avoid the onrushing mass. The two airboats came together with a great crushing of wood and ripping of canvas. Not the craft I had selected was stoutly built as one would expect from the damned Hamalese

The name of these Text files are as follows-

12ItrainA1.TXT 12ItrainA2.TXT 12ItrainA3.TXT 12ItrainB1.TXT
12ItrainB2.TXT 12ItrainC1.TXT 12ItrainC2.TXT 12ItrainD1.TXT
12ItrainD2.TXT 12ItrainE1.TXT 12ItrainE2.TXT 12ItrainF1.TXT
12ItrainF2.TXT 12ItrainG1.TXT 12ItrainG2.TXT 12ItrainH1.TXT
12ItrainH2.TXT 12ItrainI1.TXT 12ItrainI2.TXT 12ItrainJ1.TXT
12ItrainJ2.TXT 12ItrainK1.TXT 12ItrainK2.TXT 12ItrainK3.TXT
12ItrainL1.TXT 12ItrainL2.TXT 12ItrainL3.TXT 12ItrainM1.TXT
12ItrainM2.TXT 12ItrainM3.TXT 12ItrainN1.TXT 12ItrainN2.TXT
12ItrainN3.TXT

Such that the name of the Authors are after the keyword 'train' and then after the name of the author the number which is to differentiate between different novels of the same author.



3 Syntactic Recurrent Neural Network

We introduce a syntactic recurrent neural network to encode the syntactic patterns of a document in a hierarchical structure. First, we represent each sentence as a sequence of POS tags and each POS tag is embedded into a low dimensional vector and a POS encoder (which can be a CNN or LSTM) learns the syntactic representation of sentences. Subsequently, the learned sentence representations aggregate into the document representation. Moreover, we use attention mechanism to reward the sentences which contribute more to the prediction of labels. Afterwards we use a soft-max classifier to compute the probability distribution over class labels.

The overall architecture of the network is

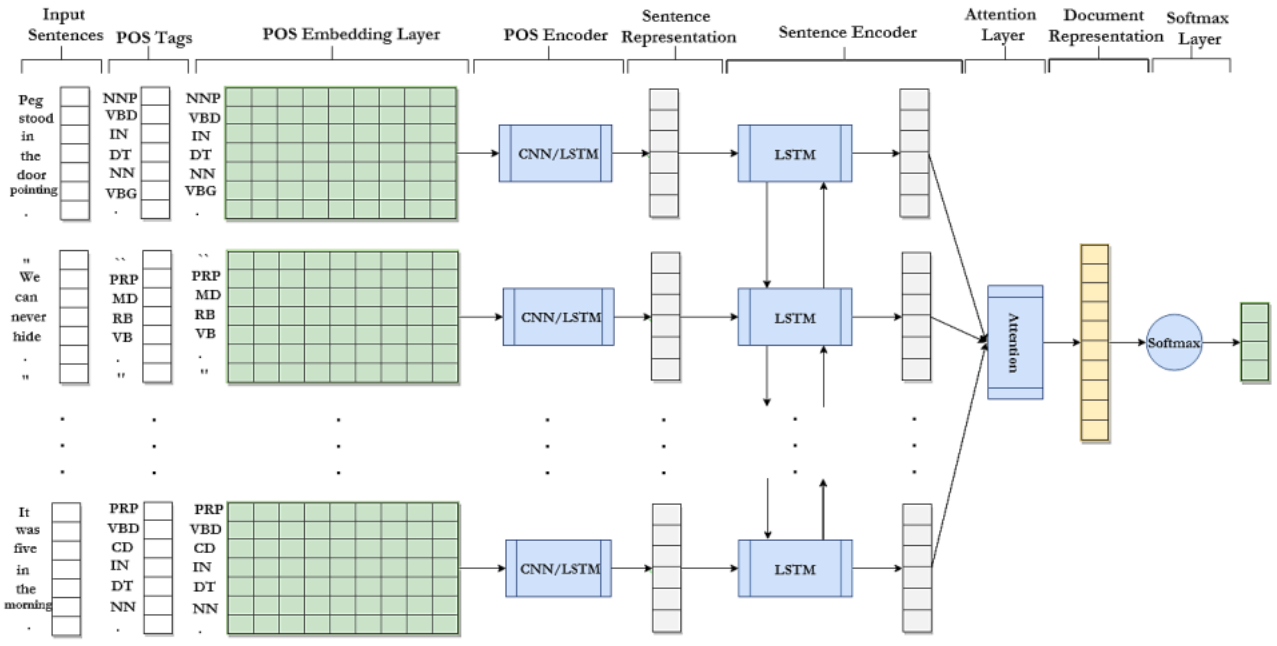


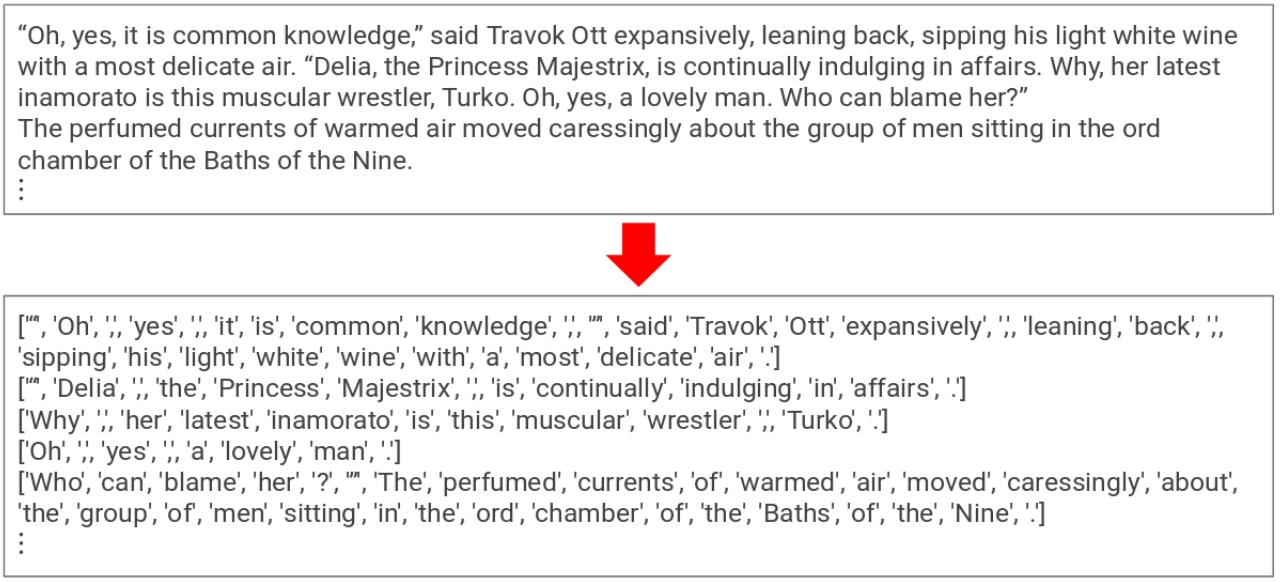
Figure 1: The Overall Architecture of Syntactic Recurrent Neural Network for Style-based Text Classification

4 POS Embedding

We assume that each document is a sequence of M sentences and each sentence is a sequence of N words, where M , and N are model hyper parameters.Given a sentence, we convert each word into the corresponding POS tag in the sentence and afterwards we embed each POS tag into a low dimensional vector P,using a trainable lookup table.

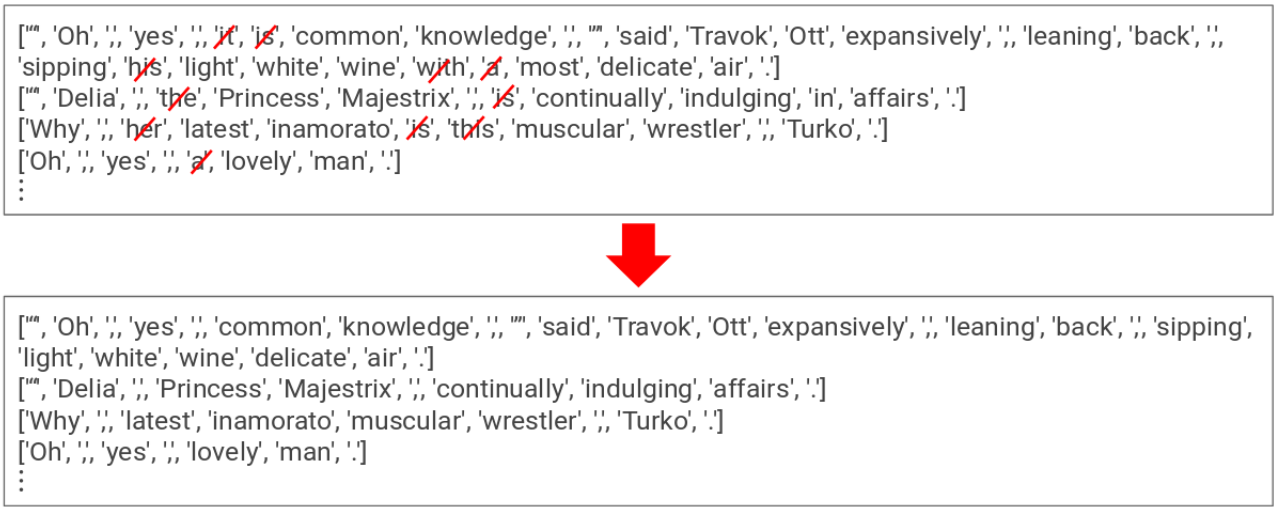
4.1 Loading Novels

The model first takes the Dataset, Reads it and converts all the words to token.
As shown below-



4.2 Removing Stop words-

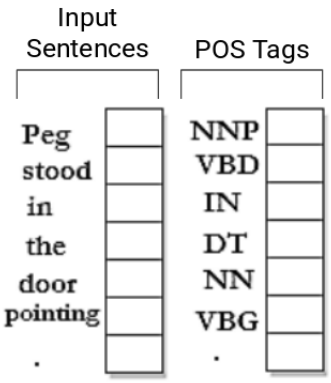
There are certain words which are useless as tokens to aur models and are very frequent in the novels,they do not make any difference and does not even tell the style of the author.Hence we need to remove them.



4.3 Word to POS tags

For further processing of Data , we need to change every word to part-of-speech tag.We use NLTK part-of-speech tagger for the tagging purpose and use the set of 47 POS tags in our model as follows.

$$T = \{\ CC,\ CD,\ DT,\ EX,\ FW,\ IN,\ JJ,\ JJR,\ JJS,\ LS,\ MD,\ NN,\ NNS,\ NNP,\ NNPS,\ PDT,\ POS,\ PRP,\ PRP\$,\ RB,\ RBR,\ RBS,\ RP,\ SYM,\ TO,\ UH,\ VB,\ VBD,\ VBG,\ VBN,\ VBP,\ VBZ,\ WDT,\ WP,\ WP\$,\ WRB,\ ',',\ ':',\ '...',\ ';',\ '?',\ '!',\ '.',\ '\$',\ '(',\ ')',\ "`",\ '"'\ \}$$




```
['“', 'Oh', ',', 'yes', ',', 'common', 'knowledge', ',', '”', 'said', 'Travok', 'Ott', 'expansively', ',', 'leaning', 'back', ',', 'sipping',
'light', 'white', 'wine', 'delicate', 'air', '.']
['“', 'Delia', ',', 'Princess', 'Majestrix', ',', 'continually', 'indulging', 'affairs', '.']
['Why', ',', 'latest', 'inamorato', 'muscular', 'wrestler', ',', 'Turko', '.']
['Oh', ',', 'yes', ',', 'lovely', 'man', '.']
⋮
```

```
['NN', 'NNP', ',', 'UH', ',', 'JJ', 'NN', ',', 'NNP', 'VBD', 'NNP', 'NNP', 'RB', ',', 'VBG', 'RB', ',', 'VBG', 'JJ', 'JJ', 'NN', 'NN',
'NN', '.']
['NN', 'NNP', ',', 'NNP', 'NNP', ',', 'RB', 'VBG', 'NNS', '.']
['WRB', ',', 'JJS', 'NN', 'JJ', 'NN', ',', 'NNP', '.']
['UH', ',', 'UH', ',', 'JJ', 'NN', '.']
⋮
```

4.4 POS tags to sequence number

The POS tags has a numbers assigned to it. The tags generated are converted to sequence number using this mapping.

```
['PRP', 'NN', 'NN', ',', 'NN', 'VBD', ',', 'JJ', 'NNS', 'MD', 'VB', 'JJ', 'NN', 'JJ', 'VBZ', '.']
['VB', 'WRB', 'JJ', 'IN', 'JJ', 'VBD', '.', 'JJ', 'NNP', 'VBD', '.']
['PRP', 'VBD', 'NNP', 'VBD', '.']
['RB', 'NNP', 'RB', 'VBD', '.']
['PRP', 'VBP', 'RB', ',', 'JJ', 'VBG', 'NN', ',', 'NN', 'VBD', 'NNP', 'RB', 'JJ', 'RB', '.']
⋮
```

```
[34, 26, 26, 16, 26, 7, 16, 8, 1, 20, 32, 8, 26, 8, 10, 44]
[32, 33, 8, 40, 8, 7, 44, 8, 15, 7, 44]
[34, 7, 15, 7, 44]
[30, 15, 30, 7, 44]
[34, 29, 30, 16, 8, 17, 26, 16, 26, 7, 15, 30, 8, 30, 44]
⋮
```

Then, padding is added to each sentence to make them of the same length.

```
[34, 26, 26, 16, 26, 7, 16, 8, 1, 20, 32, 8, 26, 8, 10, 44]
[32, 33, 8, 40, 8, 7, 44, 8, 15, 7, 44]
[34, 7, 15, 7, 44]
[30, 15, 30, 7, 44]
[34, 29, 30, 16, 8, 17, 26, 16, 26, 7, 15, 30, 8, 30, 44]
⋮
```

```
[34 26 26 16 26  7 16  8  1 20 32  8 26  8 10 44  0  0  0  0  0  0  0  0  0  0  0  0  0]
[32 33  8 40  8  7 44  8 15  7 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[34  7 15  7 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[30 15 30  7 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
[34 29 30 16  8 17 26 16 26  7 15 30  8 30 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
⋮
```

5 POS Encoder

POS encoder learns the syntactic representation of sentences from the output of POS embedding layer. In order to investigate the effect of short-term and long-term dependencies of POS tags in the sentence, we exploit both CNNs and LSTMs.

5.1 Short-term Dependencies

CNNs generally capture the short-term dependencies of words in the sentences which make them robust to the varying length of sentences in the documents.
Let $S_i = [P_1; P_2; ...; P_N]$ be the vector rep resentation of sentence i and $W \in R^{rd_p}$ be the convolutional filter with receptive field size of r. We apply a single layer of convolving filters with varying window sizes as the of rectified linear unit function (relu) with a bias term b, followed by a temporal max-pooling layer which returns only the maximum value of each feature map $C_i^r \in R^{N-r+1}$.
Consequently, each sentence is represented by its most important syntactic n-grams, independent of their position in the sentence. Variable receptive field sizes Z are used to compute vectors for different n-grams in parallel and they are concatenated into a final feature vector $h_i \in R^K$ afterwards, where K is the total number of filters:

$$C_{ij}^r = relu(W^T S_{j:j+r-1} + b), j \in [1, N-r+1],$$
$$\hat{C_i^r} = max\{C_i^r\},$$
$$h_i = \oplus \hat{C_i^r}, \forall r \in Z$$

5.2 Long-term Dependencies

Recurrent neural networks especially LSTMs are capable of capturing the long-term relations in se- quences which make them more effective com- pared to the conventional n-gram models where increasing the length of sequences results a sparse matrix representation of documents.
Let $S_i = [P_1; P_2; ...; P_N]$ be the vector representation of sentence i. As an alternative to CNN, we use a bidirectional LSTM to encode each sentence. The forward LSTM reads the sentence $S_i from P_1 to P_N$and the backward LSTM reads the sentence from $P_N to P_1$. The feature vector $h_t^t \in R^{2d_l}$is concatenation of the forward LSTM and the backward LSTM, where$d_l$is the dimensionality of the hidden state. The final vector representation of sentence$i, h_i^s \in R^{2d_l}$is computed as unweighted sum of the learned vector represen- tation of POS tags in the sentence. This allows us to represent a sentence by its overall syntactic pattern.

$$\overrightarrow{h_t^p} = LSTM(P_t), t \in [1, N],$$

$$\overleftarrow{h_t^p} = LSTM(P_t), t \in [N, 1],$$

$$h_t^p = [\overrightarrow{h_t^p}; \overleftarrow{h_t^p}]$$

$$h_i^s = \sum_{t \in [1,N]} h_t^p$$

6 Sentence Encoder

Sentence encoder learns the syntactic representation of a document from the sequence of sentence representations outputted from the POS encoder. We use a bidirectional LSTM To capture how sentences with different syntactic patterns are struc- tured in a document.

$$\overrightarrow{h_i^d} = LSTM(h_i^s), i \in [1, M],$$

$$\overleftarrow{h_i^d} = LSTM(h_i^s), i \in [M, 1],$$

$$h_i^d = [\overrightarrow{h_i^d}; \overleftarrow{h_i^d}]$$

Needless to say, not all sentences are equally informative about the authorial style of a document. Therefore, we incorporate attention mechanism to reveal the sentences that contribute more in detecting the writing style. We define a sentence level vector u_s and use it to measure the importance of the sentence i as follows:

$$u_i = tanh(W_s h_i^d + b_s)$$

$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_i exp(u_i^T u_s)}$$

$$V = \sum_i \alpha_i h_i^d$$

Where u_s is a learnable vector and is randomly initialized during the training process and V is the vector representation of document which is weighted sum of vector representations of all sentences.

7 Classification

The learned vector representation of documents are fed into a softmax classifier to compute the probability distribution of class labels. Suppose V_k is the vector representation of document k learned by the attention layer. The prediction $\tilde{y^k}$ is the output of softmax layer and is computed as:

$$\tilde{y_k} = softmax(W_c V_k + b_c)$$

Where W_c, b_c are learnable weight and learn- able bias respectively and$\tilde{y^i}$ is a C dimensional vector (C is the number of classes). We use cross- entropy loss to measure the discrepancy of predic- tions and true labels y_k . The model parameters are optimized to minimize the cross-entropy loss over all the documents in the training corpus. Hence, the regularized loss function over N documents denoted by J() is:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} log \tilde{y}_{ik} + \lambda||\theta||$$

8 Experimental Results

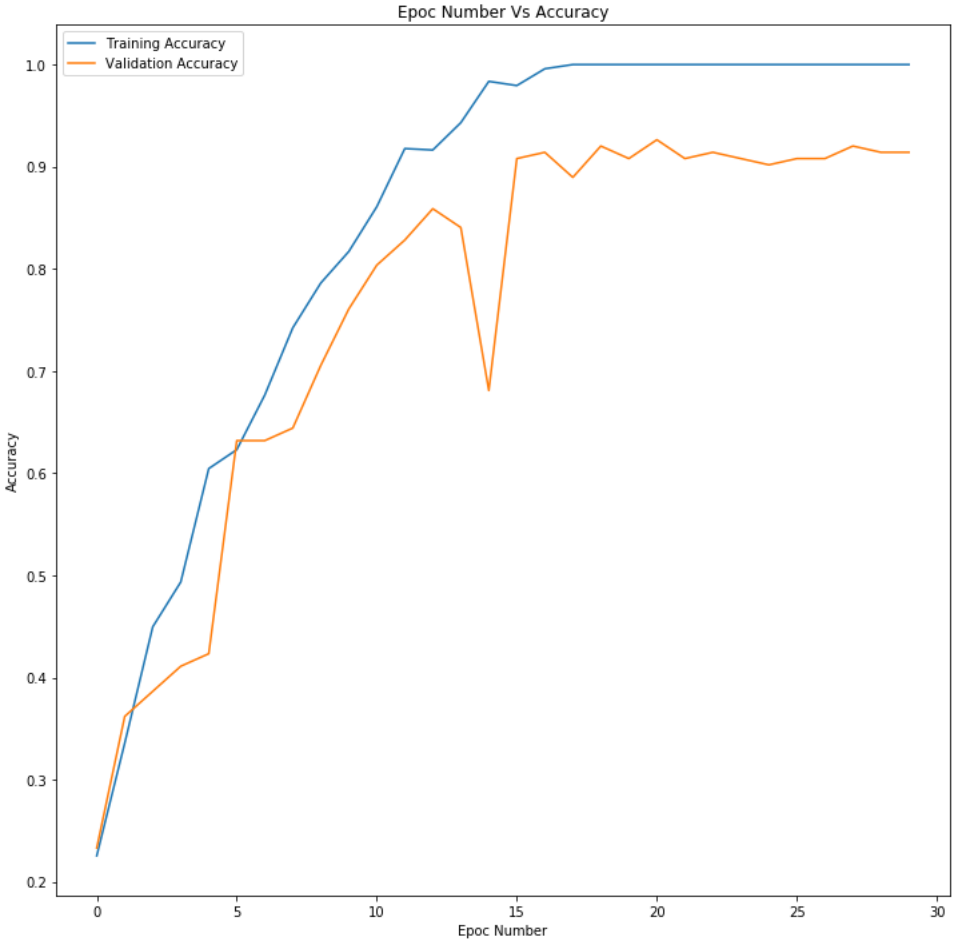
We report both segment-level and document-level accuracy. As mentioned before, each document (novel) has been divided into the segments of 100 sentences. Therefore, each segment in a novel has classified independently and afterwards the label of each document is calculated as the majority voting of its constituent segments.

Document level Accuracy for both LSTM-LSTM and CNN-LSTM Model was 100% (14/14 novels)

	Train Data I		Train Data II		Test Data	
	Word Count	Sentence Length	Word Count	Sentence Length	Word Count	Sentence Length
Candidate 01	73,449	17	76,602	19	70,112	20
Candidate 02	180,660	13	117,024	14	82,317	13
Candidate 03	158,306	17	121,301	19	151,049	15
Candidate 04	84,080	14	79,413	18	93,055	14
Candidate 05	109,857	18	141,086	15	96,663	15
Candidate 06	61,644	19	46,549	16	42,808	16
Candidate 07	71,106	16	70,563	18	84,996	21
Candidate 08	106,024	18	113,475	15	94,700	13
Candidate 09	66,840	15	41,093	15	194,547	15
Candidate 10	86,681	14	35,699	16	60,998	16
Candidate 11	53,960	19	48,037	13	80,330	24
Candidate 12	49,543	25	64,495	26	50,636	27
Candidate 13	32,900	21	153,994	32	77,780	27
Candidate 14	89,908	23	71,058	22	52,633	35

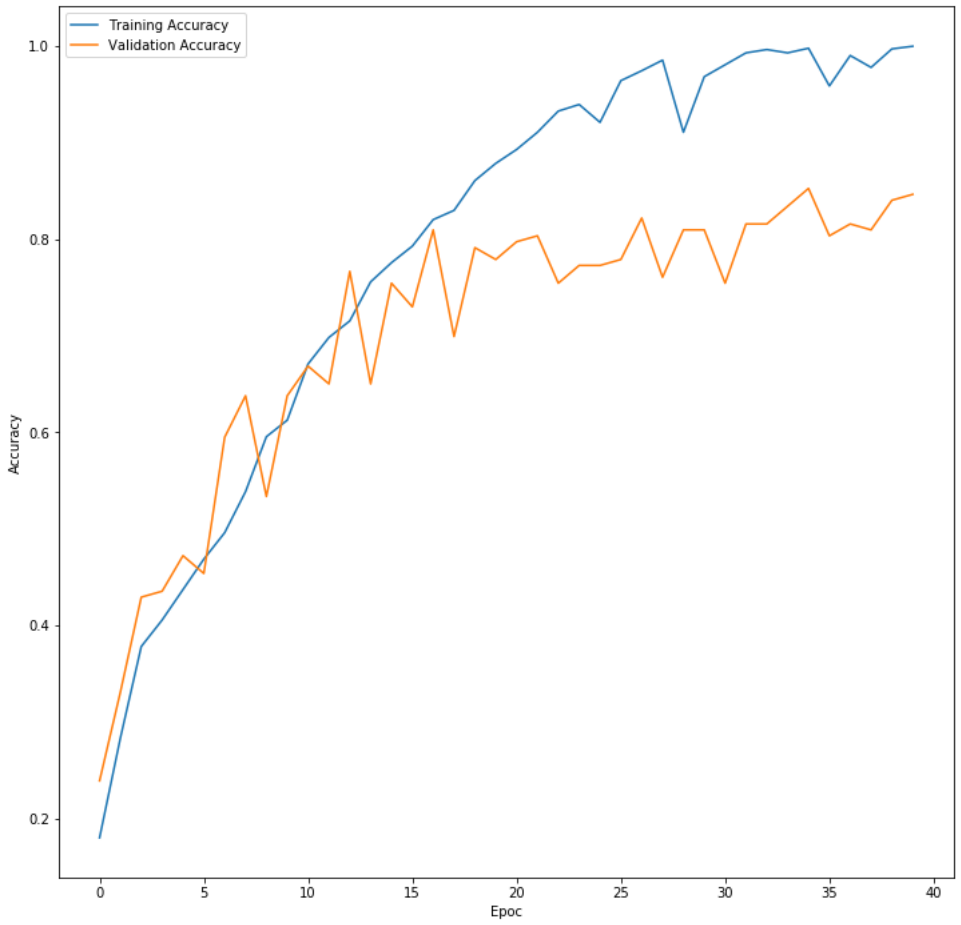
Table 1: Corpust Statistics.

Graph for Training and Validation Accuracy for LSTM-LSTM



The Validation Accuracy for LSTM-LSTM Model achieved was 72%

Graph for Training and Validation Accuracy for CNN-LSTM



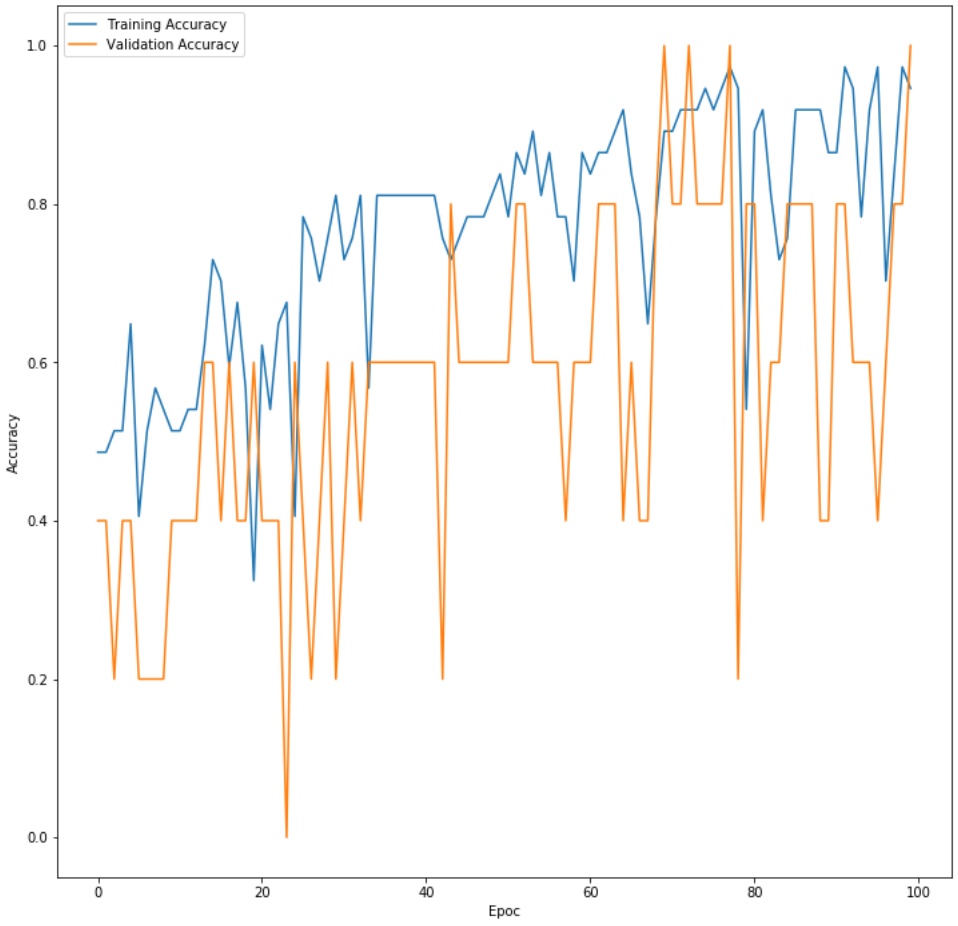
The Training Accuracy for CNN-LSTM Model achieved was 84 % and Validation Accuracy was 71%

9 Hindi-Dataset Results

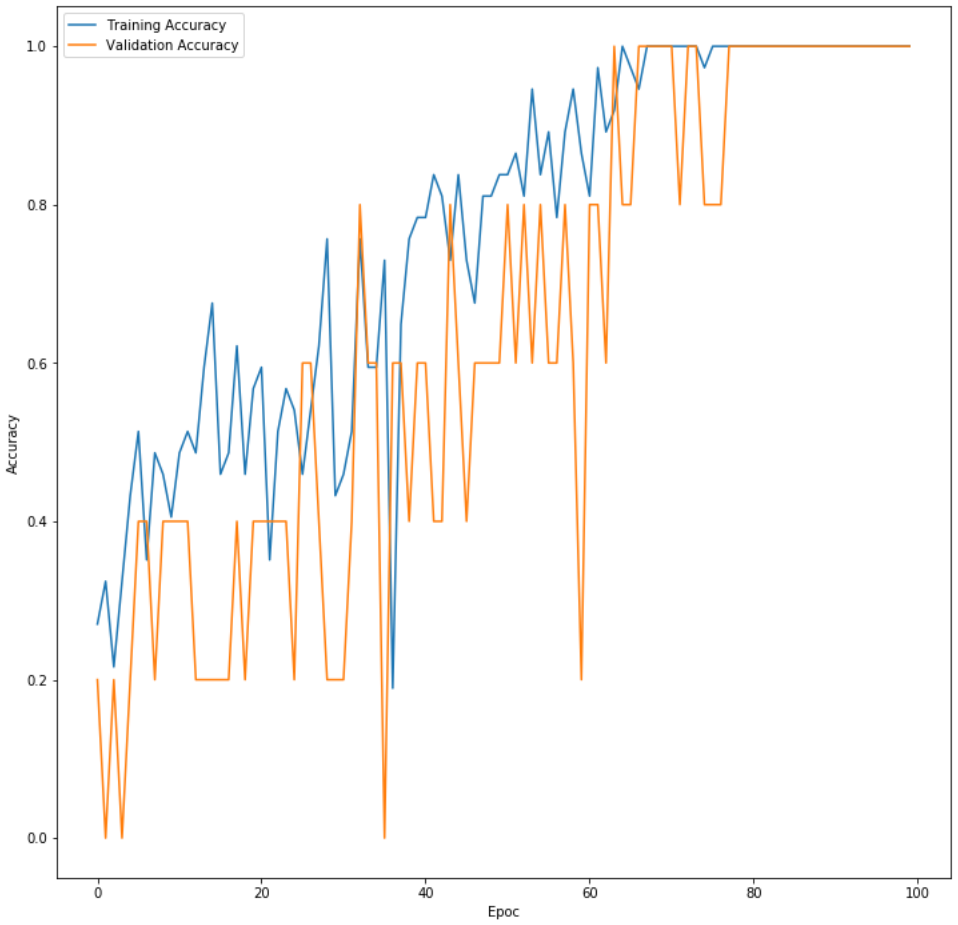
Apart from English novels and their writers, we also experimented using Hindi novel.
First, we Downloaded Hindi novels in the form of text files. Converting them to POS tags by using NLTK part-of-speech tagger for the tagging purpose of type 'INDIAN' and then after this passing the tags through the LSTM-LSTM model the same way as english data set.

There is an abnormal behavior of Model for HIndi dataset, As for the POS Tagging we used NLTK standard library but for hindi one this is incomplete .Majority of times it assigns UNK i.e., UNKNOWN POS TAG for the word.Hence finds difficult to learn.

Results obtained for Hindi dataset in LSTM-LSTM model-



Results obtained for Hindi dataset in CNN-LSTM model-



Results for Training and Validation Accuracy for CNN-LSTM with different architecture

```
Epoch 6/30
1622/1622 [==============================] - 452s 279ms/step - loss: 1.0215 - acc: 0.6097 - val_loss: 1.1883 - val_acc: 0.5304
Epoch 7/30
1622/1622 [==============================] - 452s 278ms/step - loss: 0.9032 - acc: 0.6806 - val_loss: 1.0811 - val_acc: 0.6077
Epoch 8/30
1622/1622 [==============================] - 451s 278ms/step - loss: 0.8462 - acc: 0.6948 - val_loss: 0.8266 - val_acc: 0.6740
Epoch 9/30
1622/1622 [==============================] - 452s 279ms/step - loss: 0.7003 - acc: 0.7546 - val_loss: 1.0451 - val_acc: 0.6354
Epoch 10/30
1622/1622 [==============================] - 453s 280ms/step - loss: 0.6285 - acc: 0.7663 - val_loss: 0.8501 - val_acc: 0.7017
Epoch 11/30
1622/1622 [==============================] - 452s 279ms/step - loss: 0.5004 - acc: 0.8335 - val_loss: 0.7052 - val_acc: 0.7238
Epoch 12/30
1622/1622 [==============================] - 454s 280ms/step - loss: 0.4221 - acc: 0.8607 - val_loss: 0.5035 - val_acc: 0.8011
Epoch 13/30
1622/1622 [==============================] - 454s 280ms/step - loss: 0.3425 - acc: 0.8909 - val_loss: 0.6911 - val_acc: 0.7569
Epoch 14/30
1622/1622 [==============================] - 452s 278ms/step - loss: 0.3005 - acc: 0.9026 - val_loss: 0.4181 - val_acc: 0.8287
Epoch 15/30
1622/1622 [==============================] - 456s 281ms/step - loss: 0.1974 - acc: 0.9414 - val_loss: 0.4078 - val_acc: 0.8508
Epoch 16/30
1622/1622 [==============================] - 453s 280ms/step - loss: 0.1592 - acc: 0.9544 - val_loss: 0.5474 - val_acc: 0.8122
Epoch 17/30
1622/1622 [==============================] - 455s 280ms/step - loss: 0.1237 - acc: 0.9686 - val_loss: 0.3155 - val_acc: 0.8785
Epoch 18/30
1622/1622 [==============================] - 456s 281ms/step - loss: 0.0810 - acc: 0.9815 - val_loss: 0.9125 - val_acc: 0.7127
Epoch 19/30
1622/1622 [==============================] - 457s 282ms/step - loss: 0.1532 - acc: 0.9519 - val_loss: 0.2483 - val_acc: 0.9171
Epoch 20/30
1622/1622 [==============================] - 456s 281ms/step - loss: 0.0330 - acc: 0.9975 - val_loss: 0.2206 - val_acc: 0.9227
Epoch 21/30
1622/1622 [==============================] - 463s 285ms/step - loss: 0.0126 - acc: 1.0000 - val_loss: 0.2062 - val_acc: 0.9171
Epoch 22/30
1622/1622 [==============================] - 464s 286ms/step - loss: 0.0071 - acc: 1.0000 - val_loss: 0.2105 - val_acc: 0.9337
Epoch 23/30
1622/1622 [==============================] - 461s 284ms/step - loss: 0.0053 - acc: 1.0000 - val_loss: 0.2160 - val_acc: 0.9171
Epoch 24/30
1622/1622 [==============================] - 465s 286ms/step - loss: 0.0037 - acc: 1.0000 - val_loss: 0.2176 - val_acc: 0.9171
Epoch 25/30
1622/1622 [==============================] - 468s 289ms/step - loss: 0.0029 - acc: 1.0000 - val_loss: 0.2145 - val_acc: 0.9227
Epoch 26/30
1622/1622 [==============================] - 464s 286ms/step - loss: 0.0024 - acc: 1.0000 - val_loss: 0.2189 - val_acc: 0.9171
Epoch 27/30
```

Github link - https://github.com/danish241194/Syntactic-Recurrent-Neural-Network-for-Authorship-Attribution

Work distribution-

1. Danish Mukhtar Zargar
    - For PAN 2012 dataset, Data pre processing and Syntactic LSTM-LSTM model training and validation.
2. Sivangi Singh
    - Preparation of hindi data set. For this data set, data pre-processing and LSTM-LSTM Model training and validation.
3. Monu Tayal
    - For PAN 2012 and hindi dataset, Syntactic CNN-LSTM Model training and validation.
4. Girdhari Lal Gupta
    - CNN-LSTM model with different architecture training and validation.

THANKYOU