Can language models learn the style of a text author?

Martynas Lukoševičius (marlu207)

Project for a 732A92 Text Mining course

Abstract

The aim of this project is to evaluate if text generating language models are capable to learn the style of an author on which texts/speeches models were trained. To do this, the definition of written style is analysed, defined the components. For text generation, n-gram, LSTM based, and GPT-2 simple version language models were considered. Style similarity was evaluated regarding quality using the M-BLEU4 score and style strength using two trained classifiers. For the first classifier naïve bayes, gradient boosting, SVM and logistic regression with word count vectorization input was considered. The highest accuracy was obtained with naïve bayes classifier = 0.92. The second classifier is BiLSTM based where inputs are parts of speech tags and their dependencies, classifier obtained an accuracy of 0.85. Using these metrics, language model generated texts were compared with original D. Trump rally speeches and other politician rally speeches using the Mann-Whitney U test. With a significance of 0.05, none of the language models were capable to generate texts which would be similar to real D. Trump speeches in all three criteria. Thus, there is no evidence to say that in this project considered language models are capable to learn the style of the author.

1. Introduction

Natural language generation is one of the most active research areas in deep learning. It is a task to train a model in such a way that given text sequence, model would predict the next word. With the development of deep learning algorithms, models can now generate more fluent and semantically meaningful text than conventional methods [1]. Current high–performance text generation language models (LM) such as GPT-2 contains 1.5 billion parameters and were trained on 8 million web pages [2]. These kinds of models are hardly possible to train starting with random weights by individuals. However, pre-trained models are available on the internet. It is hypothesized that it could be possible for non-experts to build a powerful threat model to be used for fake news or review generation [3].

In this project, it will be evaluated if text generating LM are capable to learn the style of an author on which texts/speeches models are trained or fine-tuned. To do this, first we define what is writers/speakers' style, what are the components of the style. Secondly, we review the theory of language models, auto-regressive text generation and define LM which will be used in this project. Later we analyse how automatically evaluate style, according to that, define and train style classifiers. Thirdly, LM are trained. Finally, we compare and discuss the results.

2. Theory

2.1. What is style:

Elaine Danielson Fowler provides 4 definitions of style [4]:

1. "Style is the total effect of a writer's decisions--his choice of words and sentence structures, his selection of details, images and rhetorical patterns"
2. "Style is the way something is put, and people who put things well are said to have "a way with words""
3. "Writer's style is the sound of a voice on the page."
4. "An author's style refers to how an author says something rather than what the author says"

From these definitions we can extract main components of style, which would be:

- Word choices
- Sentence structure – usually it depends on a language, but users personalize it by making shorter or longer sentences, orders of part of speech etc.
- Imagery – it is elements such as simile, metaphor, personifications etc.

In this project, only word choices and sentence structure will be evaluated as there is no automatic evaluation method to estimate the imagery part of a style.

2.2. Language models

A statistical language model is a model which computes the probability distribution over the sequence of tokens w_i (Equation 1), where each token is a word or a part of word from a vocabulary.

$$P(w_1, w_2, \dots w_n)$$

Equation 1. Probability distribution of language model

Language models are trained on a corpus of tokens W. The standard language modelling objective is to maximize likelihood $L(W) = \sum_i P(w_i|w_{i-k}, \dots, w_{i-1})$, where k is the size of the context window. In text generation task language model can be used to compute the conditional probability of the next word (Equation 2).

$$\mathrm{p}(w_n|w_1, w_2, \dots w_{n-1})$$

Equation 2. The conditional probability of the next word given previous words

2.2.1. N-gram model

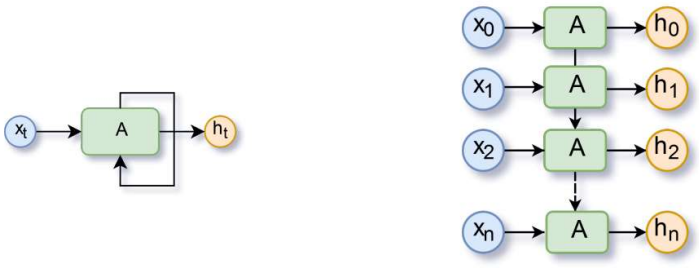
The N-gram model is one of the simplest language models which can be used for text generation. To calculate the conditional probability of the next word w_i, the count of the sequence of w_i and n-1 previous words are divided by the count of the n-1 previous words.

$$p(w_i|w_{i-n}, .., w_{i-1}) = \frac{count(w_i, w_{i-1}, \dots, w_{i-n})}{count(w_{i-1}, \dots, w_{i-n})}$$

Equation 3. The conditional probability of a word given previous n words

2.2.2. RNN

A recurrent neural network is a type of neural network that allows previous outputs to be used as inputs while having hidden states. In feed-forward networks, only the output of the previous layers is fed into the next layer. However, in RNN the output of each cell is given back to the same cell as input. Because of this, the RNN cell can remember the sequence of previous inputs.



(a) RNN in state representation (b) unfolded RNN sequence

Figure 1. The architecture of RNN [5]

Figure 1 represents RNN at time t and unfolded version where x_t is the input at time step t and h_t is the output at time step t. Typically, RNN are used to model time series, machine translation, text generation, speech recognition etc.

2.2.3. LSTM

With long sequences RNN faces a vanishing gradient problem, to overcome this Long Short Term Memory cell (LSTM) was introduced. LSTM uses a gate mechanism, it contains an update gate, relevance gate, forgets gate and output gate.

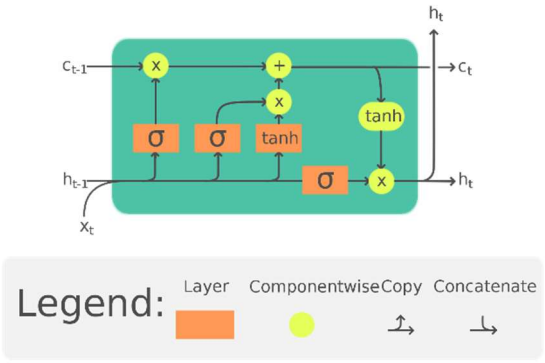


Figure 2. architecture of LSTM [6]

- $x_t - input\ vector$
- $h_t - hiden\ state\ vector$
- $c_t - cell\ state\ vector$
- $\sigma - \ sigmoid\ activation\ function$
- $tanh \ - \ hyperbolic\ tangent\ function$

2.2.4. Bidirectional LSTM

In a sequence classification task, Bidirectional LSTMs can be used. It consists of two LSTM layers of opposite direction where outputs go through the same activation function. Because of this Bidirectional LSTM can consider not only the past sequence as LSTM but also a future sequence.

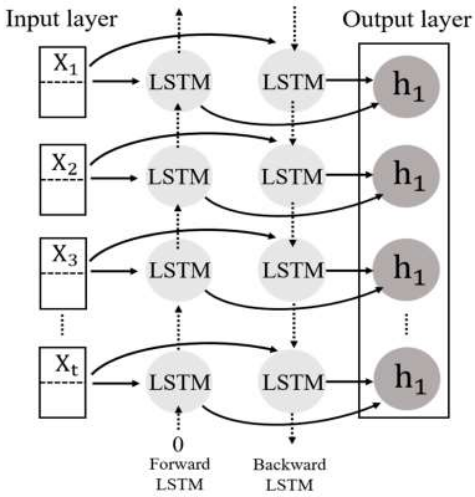


Figure 3. BiLSTM process [7]

2.2.5. GPT

Generative pre-trained Transformer models are unsupervised transformer-based models pre-trained on web articles. After pretraining, a model can be fine-tuned for other tasks such as language modelling, text classification etc.

2.2.5.1. GPT architecture

GPT model uses transformer architecture (Figure 4) which was introduced in the paper "Attention is all you need" [8], but without encoder part. Input to the model, usually text corpus, is tokenized and embedded, later position encoded. Position encoding is a sine or cosine function of different frequencies. This is necessary because the transformer model doesn't contain any recurrence or convolution. The next layer is the decoder part which consists of a masked multi self – attention mechanism, residual connection and layer normalisation followed by an additional feed-forward layer, residual connection and normalization layers. After N decoder parts (depends which GPT version) comes linear layer with SoftMax activation function [9].

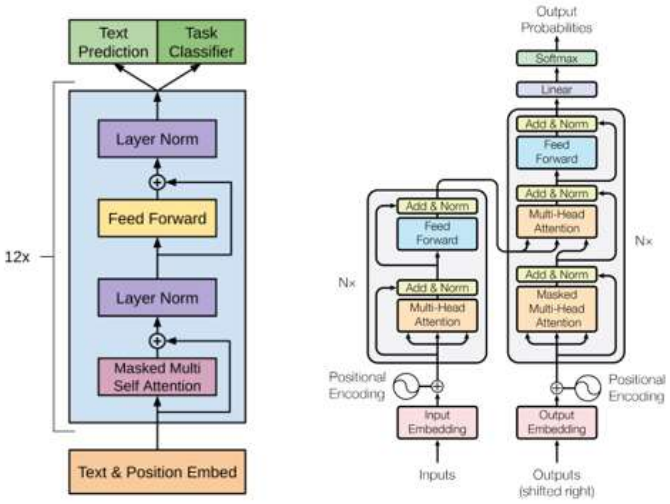


Figure 4. GPT architecture (left) [9] and transformer architecture (right) [8]

2.2.5.2. Multi-Head Attention layer

The attention layer takes 3 inputs: queries Q, keys K, values V. as seen in the figure first dot product is calculated between queries and keys, later they are scaled and sent through a SoftMax function. The last step is to calculate the dot product with V values. In practice Q, K and V are vectors. Mathematically Scaled dot product attention can be written as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Equation 4. Attention layer equation

Where d_k is a dimension of K values.

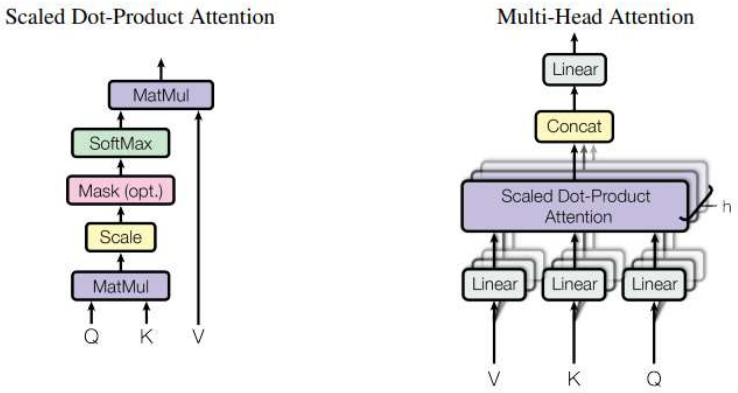


Figure 5. (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several [8]

It was found beneficial to project V, K, Q values linearly with different learned linear projections and then on every projection use scaled dot product attention and later to concatenate results. This layer is called Multi-Head Attention [8].

To maintain the auto-regressive property in the decoder part left part of the input has to be masked. This is done in the attention layer by setting values to $-\infty$. (Figure 5)

2.2.5.3. GPT-2

Is a scaled-up version of GPT, where parameter size and training corpus size was increased 10 times. It has 1.5 billion parameters and is trained on the WebText dataset which contains texts from 45 million websites. [2]

2.3. Auto – regressive language generation

Auto regressive language generation is a language generation when predicted word w_n is used to predict the next word w_{n+1}. As mentioned before next word is predicted using condition probability (Equation 2). The simplest way to generate text is to select the next word which has the highest probability. This sampling is called a greedy search. However, in this way generated text will lack the text diversity and will start to repeat [10]. To avoid this, it would be better to sample from the conditional distribution, this will make the text more diverse and unpredictable, but rarely will introduce words that are not probable [10]. This can be solved by selecting k most probable words and normalizing distribution (Top - K sampling) or selecting the most probable words until a sum of probabilities will be larger than p and then normalizing it (Top – p sampling). Another common technique is to shape a distribution using temperature. Given logits $u_{1:|V|}$ and temperature t, SoftMax is calculated as:

$$p(x = V_l|x_{1:i-1}) = \frac{exp(u_l/t)}{\sum_{l'} exp(u'_l/t)}$$

Equation 5. SoftMax with temperature

However, analysis has shown that while lowering the temperature improves generation quality, it comes at the cost of decreasing diversity [10].

2.4. Models Used for experiments.

In this project for open text generation experiments N-gram model, LSTM based model and pretrained GPT2 small version will be used. For all models, the text will be tokenized with the same tokenizer used to train GPT-2. This tokenizer used Byte Pair Encoding and has a vocabulary of 50257 tokens.

2.4.1. N-Gram language model

N-gram model is a model as described in section 2.2.1

2.4.2. LSTM based language model.

This model consists of an embedding layer that projects tokens to the dimension of 100, followed by 2 blocks of LSTM layer with 254 hidden states which returns sequence and dropout layers with a rate of 0.2 and 0.5. In the end, a feed-forward network is added with 500 hidden neurons and an output of 50257 neurons (Figure 6). This model outputs logits which later during sampling are converted to probabilities.

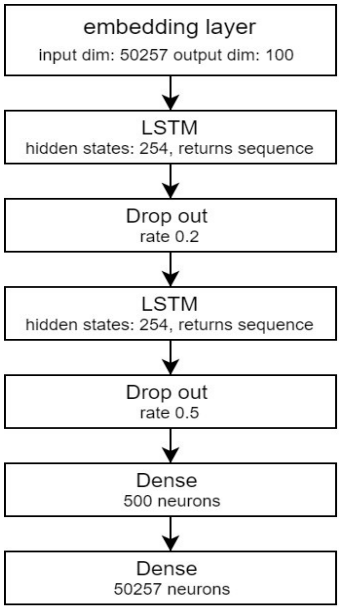


Figure 6. Architecture of LSTM based generator

2.4.3. GPT-2 – simple

Pretrained GPT-2 simple model [11] will be used, it's a smaller version of GPT2 which has 124M parameters. It was pretrained on 40 GB of WebText data.

2.5. Generated text style evaluation

To evaluate if text generating models can learn the style of the text, style strength [12] and quality [13] [14] will be evaluated. For style strength, an evaluation style classifier is used to distinguish the attributes. Originally style classifiers were used in style transfer tasks to judge whether model generated samples belong to the target class by taking N generated samples and calculating the percentage of correctly classified [15]. However, it was shown that on some datasets style classifier results correlate with human evaluation but has no correlation with others [16]. To account for style quality – usually n-gram precision-based metrics are used including BLEU, ROUGE, METEOR.

2.5.1. BLEU score

The Bilingual Evaluation Understudy [17] is a metric used to evaluate the similarity between two sentences. Originally Bleu score was developed to automatically evaluate machine translations. It works by comparing candidate translation with one or many references.

$$BLEU = BP * \exp\left(\sum_{n=1}^{N} w_n \, log(p_n)\right)$$

$$BP = \begin{cases} 1 \; if \; c > r \\ e^{(1-r/c)} \; if \; c \leq r \end{cases}$$

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gra \quad \in C} Count_{clip}(n-gram)}{\sum_{C' \in Candidates} \sum_{n-gra \quad ' \in C'} Coun\text{t}(n-gram')}$$

$$Count_{clip} = min(Count, max\_ref\_count)$$

- $BP \; - \; brevity \; penalty, penalizes \; score \; if \; candiate \; is \; shorter \; compared \; to \; references$
- $w_n \; - \; positive \; weights \; summing \; to \; one.$
- $r - \; the \; effective \; reference \; corpus \; legth$
- $c \; - \; the \; length \; of \; candidate \; translation$

Equation 6. BLEU score

Bleu score is often used to evaluate text style transfer models and open-text generation. In the context of open text generation, references are the text samples. However, the length of reference sentences and generated text should be taken into consideration as a brevity penalty will penalize the score if candidate text will be shorter than references.

Another issue with the Bleu score in measuring style embodiment is the change of the context, to avoid this issue m-BLEU score was introduced [13]. It is the same BLEU score but named entities in candidate and reference sentences are replaced with token <M>. In this project, the m-BLEU4 score will be used as a similar score was used to evaluate Obama speech generation [18].

2.5.2. Style classifiers used in this project

For text classification, Naïve Bayes, Gradient Boosting Classifier, Support vector machine and logistic regression will be considered. For text pre-processing, first named entities will be removed to hide the context of the text, later text will be lowercased, stop words removed and vectorized by counts. This classifier will be based on the count and specific words used by trump. Because of this efficiency of using this classifier can be criticised as generators will be capable of only generating words that were in training data, it might end up with results correlating with BLEU score and it won't capture sentence structure. In this project, this classifier will be called "TextClassifier".

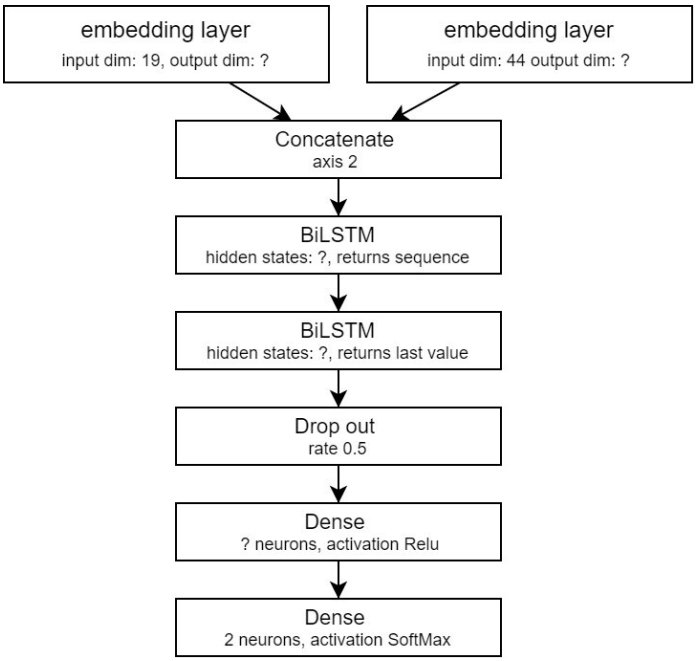


Figure 7. Architecture of "POS_DEP classifier"

To solve the last problem Bidirectional LSTM based classifier will be trained on parts of speech tags and their dependencies as inputs. The first sequence of texts will be transformed to POS and dependencies using another classifier (accuracy of POS tagging 0.97 and accuracy of dependency tagging – 0.9). POS tags and Dep tags will be tokenized and sent through embedding layers, embedding will be concatenated and sent through 2 bidirectional LSTM layers followed by dropout and feedforward network with Relu activation function and outputs with SoftMax activation.

Parameters such as the size of the embedding outputs, hidden units in LSTM layers and number of hidden neurons will be tuned. In this project, this classifier will be called the "POS_DEP classifier".

3. Data

For this project, 35 Donald Trump rally speeches from 2019 -2020 were downloaded from "Kaggle" [19] and 18 Joe Biden Rally speeches, 4 Barack Obama, 2 Bernie Sanders speeches from a campaign for Joe Biden, and 1 Kanye West rally speech scrapped from "Rev.com". Speeches given not by Donald Trump will be called "others" in this project.

While Donald Trump rally speeches didn't require any data cleaning except pre-processing depending on the model covered in the model section, speeches by others required to remove parts of it as it contained a description of what happened in the background such as applauses etc.

After cleaning speeches, files were joined into 2 separate files where one contained all speeches of D. Trump and the second of other politicians. Trump file contains 387632 words and Others – 67110 words.

To train style classifiers and generators, files were split as shown in the figure. trumpClassifier.txt and otherClassifierText.txt were used to train style classifiers, trumpGeneraorTextTrain.txt and trumpGeneratorTextValid.txt were used to train and tune language models to generate text. trumpTestText.txt and othersTestText.txt used to test and compare generators.

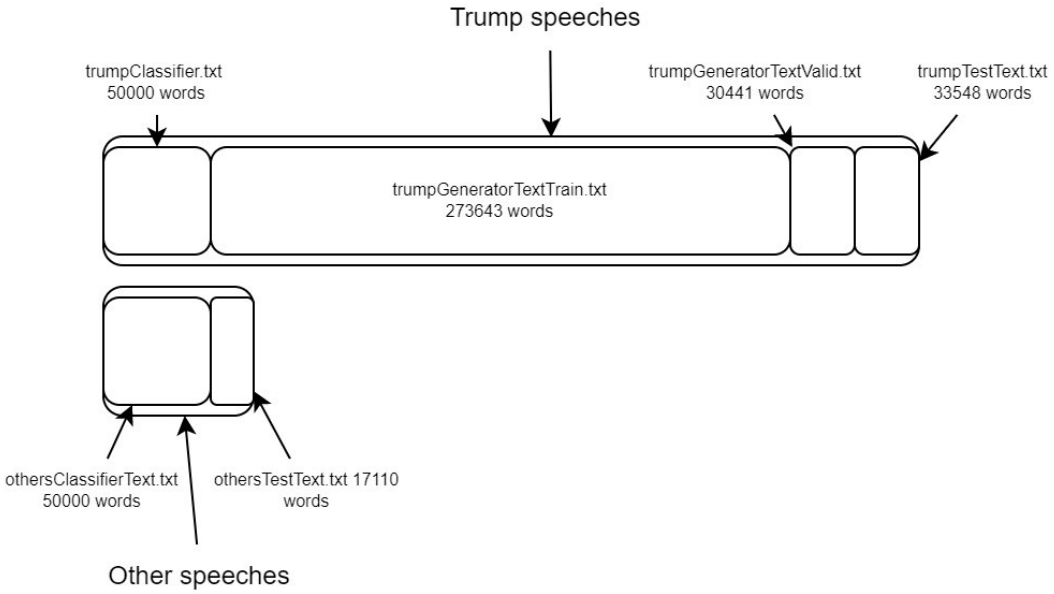


Figure 8. Visualisation of data splits

4. Method

To evaluate if language models can learn the style of D. Trump rally speeches, first style classifiers will be trained and fined tuned using grid search on trumpClassifierText.txt and othersClassifierText.txt files. Models will be evaluated regarding the accuracy as a dataset is balanced (both files have 50000 words).

After training and evaluating classifiers, n-gram and LSTM based language models will be trained and GPT-2-simple will be finetuned using trumpGeneratorTextTrain.txt file to maximise likelihood $L(W) = \sum_i P(w_i|w_{i-k}, \dots, w_{i-1})$ $where$ W is a set of tokens. LSTM based model and GPT-

2-simple model will be chosen to minimize validation loss. Parameters of sampling techniques will be optimized using grid search to maximise the mean of the M-BLEU4 score. “TextClassifier” and “POS_DEP Classifier” probabilistic predictions of being a Trump text could be used, but before final comparison, it would be necessary to retrain it on new data. Because we don’t have enough data, these style classifiers won’t be used for parameter tuning. In this part for every parameter combination model will generate N = 10 samples of texts, where the context for every sample will be a sequence of 4 words randomly selected from trumpGeneratorTextTrain.txt

Parameters of models:

Table 1. Training parameters

Parameter names	GPT-2-simple	LSTM based generator	N-gram
Input block size (tokens)	200	1000	
Batch size	1	1	
Num. epochs	100	100	
Optimizer	Adam Learning rate = 0.00001 Beta1 = 0.9 Beta2 = 0.999 Fine- tunning	Adam Learning rate = 0.001 Beta1 = 0.9 Beta2 = 0.999 Stateful training	
Sampler parameter tunning	Temperature – 0.2,0.4,0.6,0.8,1 Top – P - 0.2,0.6,0.8,1.0 Top – K - 5,10,20,50		
N-gram tunning			2,3,4,5

The final best models will be compared between themselves regarding M-BLEU4 score, “TextClassifier” probabilistic predictions and “POS_DEP classifier” probabilistic predictions. Also, generator texts will be compared with not generated trump texts and texts of others.

Instead of calculating one score value, mean or percentage of correctly classified, distributions of scores and probabilistic predictions will be formed and compared. This decision was made because one value or means cannot explain the variance – uncertainty. This can be thought of as a parametric bootstrap method. For this reason, every model will generate 100 sequences of 100 words. Every generated sequence will be evaluated with the M-BLEU4 score, where references for the M-BLEU4 score will be sequences of 100 words from trumpTestTexts.txt file, “TextClassifier” and “POS_DEP classifier” probabilistic predictions. To compare models with real D. Trump texts, TrumpClassifierText.txt and OthersClassifierText.txt will be divided into sequences of 100 words and scored with M-BLEU4 with the same references as generated texts. To make distributions for “TextClassifier” and “POS_DEP classifier”, sequences of 100 words from trumpTestText.txt and othersTestText.txt will be used. Distributions of scores will be compared using the Mann-Whitney U test. It is a nonparametric test, where the null hypothesis – two populations are equal. If for a specific model, any of the distributions will be rejected, it will be concluded that the model is not capable to learn the style of a text author.

5. Results

5.1. Style classifiers

5.1.1. TextClassifier

To train "TextClassifier" (2.5.2), trumpClassifierText.txt and othersClassifierText.txt are split into sequences of 100 words. Named entities and stop words were removed, later vectorized by counts. Data is split into train, validation and test. Naïve bayes, gradient boosting classifier, support vector machine and logistic regression are trained on test data and validated on validation data.

Table 2. Model selection scores

	Accuracy
Naïve Bayes	**0.94**
Gradient boosting classifier	0.86
Support vector machine	0.9
Logistic regression	0.92

Because the Naïve Bayes classifier has the highest scores, we will use it as a classifier. To fine-tune additive smoothing parameter, cross-validated grid search was used over train and validation data. Best accuracy 0.9417 was received with additive smoothing = 0.5. Classifier with tuned parameters on test set scored accuracy = 0.92.

5.1.2. POS_DEP classifier

To train POS_DEP classifier (2.5.2), trumpClassifierText.txt and othersClassifierText.txt are split into sequences of 100 words. Named entities removed, part of speech tags and their dependencies extracted, tokenized and right padded to a maximum number of tokens = 200. Data is split into train, validation and test. For training Adam optimizer with a learning rate = 0.001 is used. After tuning parameters, a model with 18 embedding output dimensions, 19 LSTM hidden units and 34 neurons in a dense hidden layer was chosen as it got the highest accuracy score of 0.82 on validation data. For a final model, the model with chosen parameters was trained on train and validation dataset and validated on a test set, the model with the highest validation accuracy = 0.85 was chosen as a final model.

5.2. Language models

To train all language models, TrumpGeneratorTextTrain.txt and TrumpGeneratorTextValid.txt were used.

5.2.1. N-gram model

N-gram model was trained and validated for parameters mentioned in Table 1. Highest average M-BLEU4 = 0.0585 was obtained with parameters: n = 4, temperature = 0.4, k = 5, p = 0.2.

5.2.2. LSTM model

LSTM (Figure 8) model was trained to minimize cross-entropy loss. The models lowest validation loss was 3.802 at epoch 54. After that sampling parameters, temperature, K and P were tuned using grid search to maximise the M-BLEU4 score. Best score = 0.564204 was then temperature = 0.8, k = 20, p = 0.6.

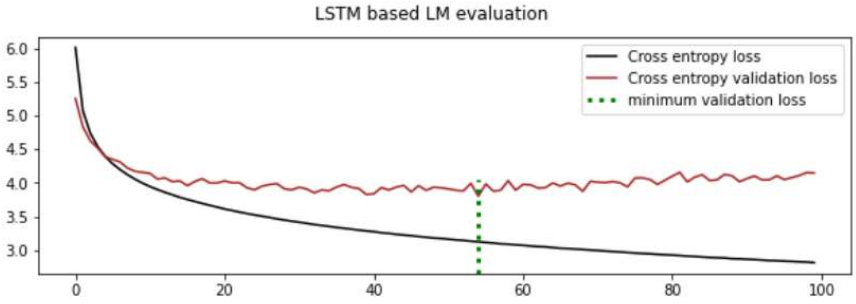


Figure 8. LSTM based LM training

5.2.3. GPT-2 -simple

GPT-2 -simple (Figure 9) model was trained to minimize cross-entropy loss. The model was chosen by the lowest validation loss which was 4.2128 at epoch 11. After that sampling parameters, temperature, K and P were tuned using grid search to maximise the M-BLEU4 score. Best score = 0.505712 was then temperature = 0.8, k = 5, p = 1.

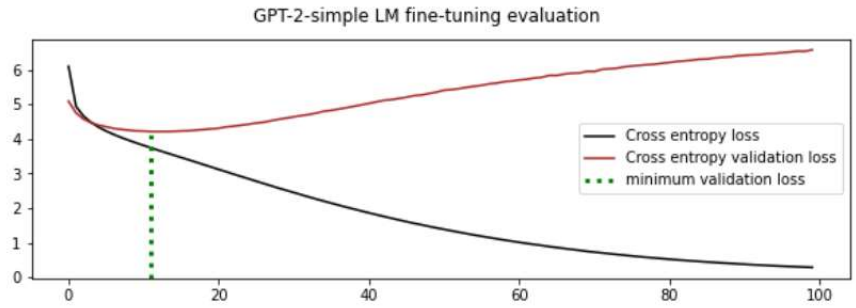


Figure 9. GPT-2-simple fine- tuning

5.3. Language model comparison

Comparing M-BLEU4 scores of the models (Figure 10) we can see that GPT2-simple and LSTM based models got almost equal scores, but it's much higher than the actual distribution of D. Trump texts. While N-gram model most of the scores are close to 0. Median values of LSTM based LM and GPT-2-simple are 0.575 and 0.504 (Table 4). According to the Mann-Whitney U test (Table 3), with a significance of 0.05, the distributions of M-BLEU4 scores of language models are not equal to distributions of M-BLEU4 scores of D. Trump and Others.

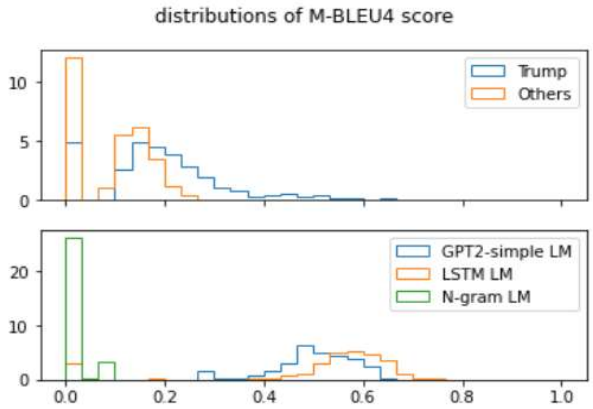


Figure 10. LM comparison: M-BLEU4 scores

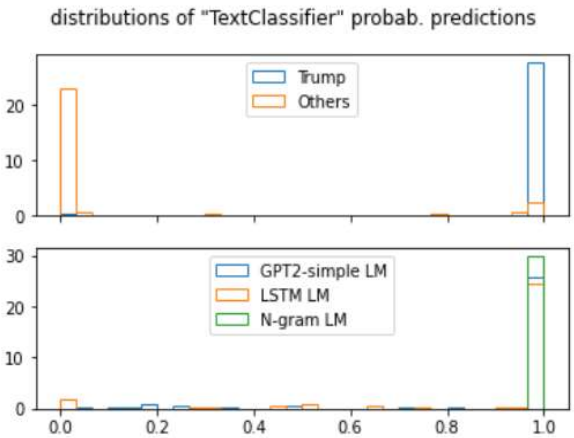


Figure 11. LM comparison: "TextClassifier" scores

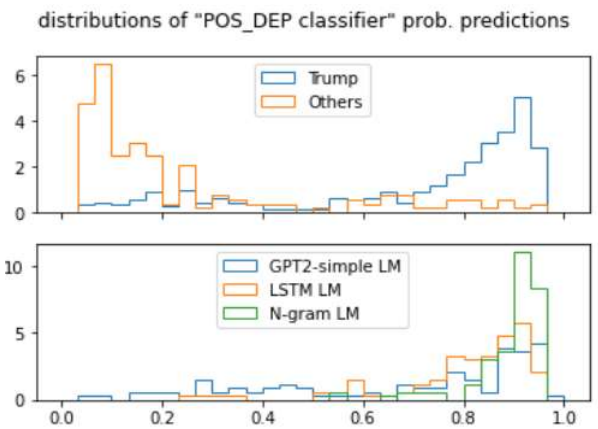


Figure 12. LM comparison: POS_DEP classifier scores

Comparing style strength by "textClassifier" scores (Figure 11), most of the texts generated by all models are close to 1. However, with a significance of 0.05. there is no evidence to reject that distributions of scores are not equal with a distribution of D. Trump speeches only for the N-Gram model (Table 3).

Table 3. Mann-Whitney U test p values

	M-BLEU4		"TextClassifier"		POS_DEP Classifier	
	Trump	Others	Trump	Others	Trump	Others
N-Gram	1.1e-53	4.45e-48	0.44	1.04e-40	6.8e-15	1.7e-37
LSTM	1.4e-30	5.05e-34	5.9e-05	2.2e-33	0.048	5.3e-32
GPT-2-simple	1.6e-44	5.2e-56	9.8e-06	2.7e-36	0.3194	4.3e-24

Distributions of scores obtained by POS-DEP Classifier (Figure 12) by the shape looks very similar for all the language models. According to the Mann-Whitney U test (Table 3), with a significance level of 0.05, the null hypothesis that distributions are equal must be rejected for LSTM LM and N-gram LM.

Table 4. median values of the distributions

	N-Gram	LSTM	GPT-2-simple	Trump	others
M-BLEU4	2.2e-78	0.575	0.504	0.1858	0.115
"TextClassifier"	0.9999998	0.999997	0.999992	0.9999998	3.5e-05
POS_DEP Classifier	0.916	0.845	0.7837	0.826	0.146

6. Discussion

Style comparison between generated texts and real texts/speeches of D. Trump and others were evaluated using M-BLEU4 score and style classifiers: "TextClassifier" and "POS_DEP Classifier". As mentioned in [12], it was expected that the M-BLEU4 score will capture quality and style classifiers - style strengths.

Median values of M-BLEU4 scores of LSTM based language model and GPT-2-simple LM are much higher than the real texts of Donald trump. This is expected because it was shown that real texts sequences are much more unpredictable than generated sample sequences [10]. However, these scores are reasonable as the SeqGAN model scored 0.427 with BLEU-4 on Obama political speeches [18]. Our model scores are higher. This can be because of 2 reasons:

1. In this project M-BLUE4 and not BLUE4 scores were used, meaning that named entities are removed and this should increase the score compared to BLUE4.
2. Sampling parameters are tuned to maximise M-BLUE4.

It might be argued if "TextClassifier" (classifier based on word counts) is necessary in text generation because language models would not use or unlikely use (in case of fine-tuning) any other words which were not in texts used for training LM's. This metric is originally used in text-style transfer tasks [12]. Also, a similar evaluation is done by the M-BLEU4 score as it captures n-gram precision. That is why it's not surprising that most of the texts generated by language models got a high probability of being a D. Trump speech.

Another limitation of this project is optimizing sampling parameters to maximise the M-BLEU4 score. As it is seen in Figure 10, M-BLEU4 scores of real D. Trump rally speeches are not high. However, no better solution for how to choose sampling parameters wasn't found.

7. Conclusion

The idea of this project was to estimate if LM are capable to learn the style of an author on which texts/speeches models are trained or fine-tuned. For this reason, N-gram, LSTM based and GPT-2-simple language models were trained on D. Trump rally speeches and sampling parameters tuned to maximise M-BLEU4 score. The style was evaluated regarding quality and style strength, where quality was evaluated with M-BLEU4 score, for style strength two classifiers were trained, where one is based on word counts (accuracy = 0.92) and another on a sequence of part of speech tags and their dependencies (accuracy = 0.85). Distributions of scores that were obtained from model generated texts were compared with real D. Trump and other speeches. Results show that regarding the M-BLEU4 score none of the distributions is equal to real D. Trump rally speech score distribution (with a significance level of 0.05). Regarding "TextClassifier" probabilistic prediction distributions n-gram language model generated texts are equal to real D. Trump rally speeches. While regarding the "POS_DEP classifier" distribution of GPT-2-simple language model is equal to real speeches of D. Trump.

It was noticed that real D. Trump speeches M-BLEU4 score is lower than LSTM based LM and GPT-2-simple, however, these results were expected.

It could be argued about the use of word count-based models for style strength evaluation in the case of language generation and not the text-style transfer task for which it was originally used.

To conclude, according to the Mann-Whitney test with a significance of 0.05, none of the language models were capable to generate texts which would be equal to real D. Trump speeches in all three criteria. Thus, there is no evidence to say that considered language models are capable to learn the style of the author.

8. References

[1] S. Lu, Y. Zhu, W. Zhang, J. Wang and Y. Yu, “Neural text generation: Past, present and beyond,” *CoRR,* vol. abs/1803.07133, 2018.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” 2019.

[3] D. I. Adelani, H. Mai, F. Fang, H. H. Nguyen, J. Yamagishi and I. Echizen, “Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-based Detection.,” *AINA,* 2020.

[4] E. D. Fowler, “Improving Style in Students' Written Compositions.,” 1999.

[5] S. Santhanam, “Context based text-generation using lstm networks,” *arXiv preprint,* vol. arXiv:2005.00048, 2020.

[6] Wikipedia, “Long short-term memory,” 2021.

[7] B. Jang, M. Kim, G. Harerimana, S.-u. Kang and J. Kim, “Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism.,” *Applied Sciences,* vol. 10(17):5841, 2020.

[8] A. Vaswani, N. Shazee, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems,* pp. 6000-6010, 2017.

[9] A. Radford and K. Narasimhan, “Improving Language Understanding by Generative Pre-Training.,” 2018.

[10] A. H. Choi, J. Buys, L. Du, M. Forbes and Yejin, “The Curious Case of Neural Text Degeneration,” *arXiv,* 2020.

[11] HuggingFace, *gpt2,* https://huggingface.co/gpt2?text=A+long+time+ago%2C.

[12] D. J. Mihalcea, Z. Jin, Z. Hu, O. Vechtomova and Rada, “Deep Learning for Text Style Transfer: A Survey,” *arXiv,* 2021.

[13] S. L. Hu, W. Wang, Z. Yang, X. Liang, F. F. Xu, E. Xing and Zhiting, “Data-to-Text Generation with Style Imitation,” *arXiv,* 2020.

[14] A. Nguyen, “Language Model Evaluation in Open-ended Text Generation,” 2021.

[15] Z. Fu, X. Tan, N. Peng, D. Zhao and R. Yan, “Style Transfer in Text: Exploration and Evaluation,” in *The Thirty-Second AAAI Conference*, 2017.

[16] J. LI, R. Jia and P. Liang, “Delete, Retrieve, Generate: a Simple Approach to Sentiment and Style Transfer,” in *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, Louisiana, Association for Computational Linguistics, 2018, pp. 1865-1874.

[17] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, Association for Computational Linguistics, 2002, pp. 311-318.

[18] L. Y. Yu, W. Zhang, J. Wang and Yong, *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient,* 2017.

[19] C. Lillelund, *Donal Trump Rally Speeches,* Kaggle.