**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 model were trained. To do this, 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 M-BLEU4 score and style strength using 2 trained classifiers. For first classifier naïve bayes, gradient boosting, SVM and logistic regression with word count vectorization input was considered. Highest accuracy was obtained with Naïve bayes classifier = 0.92. The second classifier is BiLSTM based where inputs are parts of speech tags and its dependencies, classifier obtained accuracy of 0.85. Using this metrics, language model generated texts were compared with original D. Trump rally speeches and other politician rally speeches using Mann-Whitney U test. Results shown that with a significance of 0.05, only LSTM based and GPT-2-simple language models are capable to generate texts which, regarding style strength by naïve bayes classifier evaluation, would be equal to the texts/speeches of an author on which language models are trained or fine-tuned.

# 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 a 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]. This kind of models are hardly possible to train starting with random weights by individuals. However, pretrained models are available on the internet. It is hypothesized that it could be possible for non-experts to build powerful threat model to be used for fake news or reviews generation [3].

In this project it will be estimated 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.

# Theory

## 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 stye 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 are no automatic evaluation method to estimate imagery part of a style.

## Language models

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

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 , where k is a 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).

Equation 2. Conditional probability of next word given previous words

### N – gram model

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

Equation 3. Conditional probability of word given previous n words

### RNN

Recurrent neural network is a type of neural network that allow 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, RNN cell can remember the sequence of previous inputs.

A picture containing clock

Description automatically generated

Figure 1. Architecture of RNN [5]

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

### LSTM

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

Graphical user interface, diagram, application

Description automatically generated

Figure 2. architecture of LSTM [6]

### Bidirectional LSTM

In sequence classification task, Bidirectional LSTMs can be used. It consists of 2 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.

Diagram

Description automatically generated

Figure 3. BiLSTM process [7]

### GPT

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

#### GPT architecture

GPT model uses transformer architecture (Figure 4) which was introduced in 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 sine or cosine function of different frequencies. This is necessary because transformer model doesn’t contain any recurrence or convolution. Next layer is decoder part which consists of masked multi self – attention mechanism, residual connection and layer normalisation followed by 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].

Diagram

Description automatically generated

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

#### Multi-Head Attention layer

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

Equation 4. Attention layer equation

Where is a dimension of K values.

Diagram

Description automatically generated

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

#### 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 WebText dataset which contains texts from 45 million websites. [2]

## Auto – regressive language generation

Auto regressive language generation is a language generation when predicted word is used to predict next word . As mentioned before next word is predicted using condition probability (Equation 2). Simplest way to generate text is to select next word which has highest probability. This sampling is called greedy search. However, in this way generated text will lack the text diversity and will start to repeat [10]. To avoid this, it would better to sample from the conditional distribution, this will make text more diverse and unpredictable, but rarely will introduce words which are not probable [10]. This can be solved by selecting k most probable words and normalizing distribution (Top - K sampling) or select most probable words until 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 and temperature t, SoftMax is calculated as:

Equation 5. SoftMax with temperature

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

## Models Used for experiments.

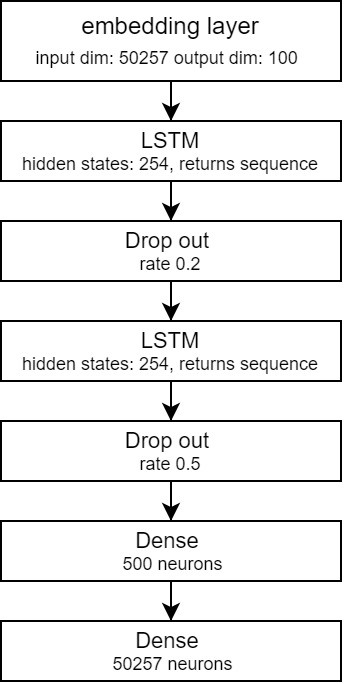


Figure 6. Architecture of LSTM based generator

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 text will be tokenized with a same tokenizer used to train GPT-2. This tokenizer used Byte Pair Encoding and has a vocabulary of 50257 tokens.

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

### LSTM based language model.

This model consists of embedding layer which 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. At the end feed forward network is added with 500 hidden neurons and output of 50257 neurons (Figure 6). This model outputs logits which later during sampling are converted to probabilities.

### 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.

## Generated text style evaluation

To evaluate how text generating models learns the style of the text, style strength [12] and quality [13] will be evaluated. For style strength evaluation style classifier is used to distinguish the attributes. This classifier is used to judge whether model generated samples belong to target class. It was shown that on some datasets style classifier results correlates with human evaluation but has no correlation with others [14]. To account for style quality – usually n-gram precision-based metrics are used including BLEU, ROUGE, METEOR.

### BLEU score

The Bilingual Evaluation Understudy [15] 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.

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 brevity penalty will penalize the score if candidate text will be shorter than references.

Another issue with 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 m-BLEU4 score will be used as similar score was used to evaluate Obama speech generation [16].

### 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 only generate words which were in training data, it might end up with a results correlating with BLEU score and it won’t capture sentence structure. In this project this classifier will be called “TextClassifier”.

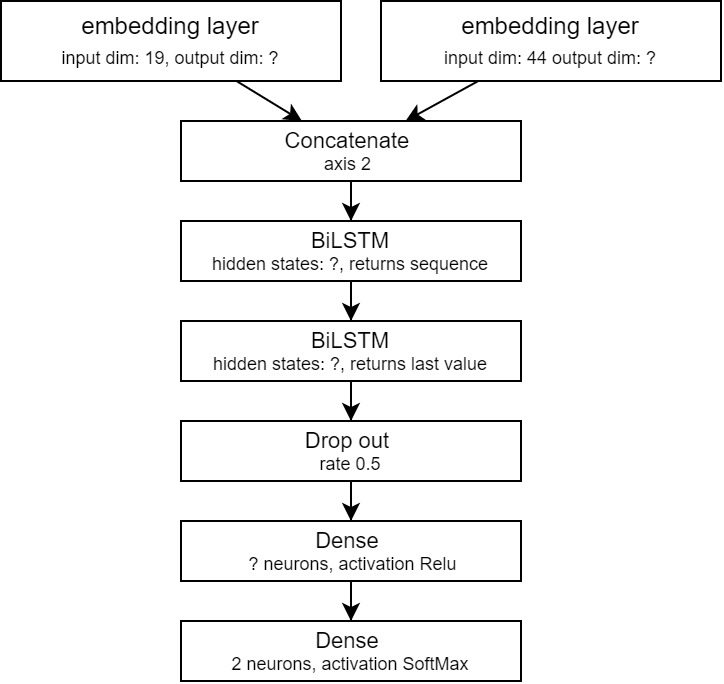


Figure 7. Architecture of “POS\_DEP classifier”

To solve the last problem Bidirectional LSTM based classifier will be trained on parts of speech and their dependencies as inputs. 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 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 “POS\_DEP classifier”.

For this project 35 Donald Trump rally speeches from 2019 -2020 downloaded from “Kaggle” [17] 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 as “others” in this project.

While Donald Trump rally speeches didn’t require any data cleaning except pre-processing depending on the model covered in model section, speeches by others required to remove parts of it as it contained 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 od D. Trump and 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.

Diagram

Description automatically generated

Figure 8. Visualisation of data splits

# Method

To evaluate similarity of style between Trump rally speeches and texts generated by language models trained on his rally speeches, first style classifiers will be trained and fined tuned using grid search on trumpClassifierText.txt and othersClassifierText.txt files.

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 W is a set of tokens. However, models will be validated and parameters tuned of models and sampling techniques to maximise mean of M-BLEU4 score, mean of “TextClassifier” probabilistic predictions and mean of “POS\_DEP Classifier” probabilistic predictions of being a Trump text. In this part for every parameter combination or epoch model will generate N = 10 samples of texts, where context for every sample will be a sequence of 4 words randomly selected from trumpGeneratorTextTrain.txt

Equation 7. Average score

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 |  |
| Initial sampling parameters | Temperature = 0.7  Top-p = 0.9 | Temperature = 0.7  Top-p = 0.9 |  |
| 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 |

Final best models will be compared between themselves regrading 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.

To compare language models, every model will generate 100 sequences of 100 words. Every generated sequence will be evaluated with M-BLEU4 score, where references for M-BLEU4 score will be sequences of 100 words from trumpTestTexts.txt file, “TextClassifier” and “POS\_DEP classifier”. 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 same references. 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 Mann-Whitney U test.

# Results

## Style classifiers

### TextClassifier

To train “TextClassifier” (2.5.2), trumpClassifierText.txt and othersClassifierText.txt are split into sequences of 100 words. Named entities and stop words 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 | Weighted average F-1 score |
| Naïve bayes | **0.94** | **0.94** |
| Gradient boosting classifier | 0.86 | 0.87 |
| Support vector machine | 0.9 | 0.9 |
| Logistic regression | 0.92 | 0.92 |

Because Naïve bayes classifier has 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 and weighted average of F1 = 0.92.

### 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 maximum number of tokens = 200. Data is split into train, validation and test. For training Adam optimizer with learning rate = 0.001 is used. After tuning parameters, model with 18 embedding output dimensions, 19 LSTM hidden units and 34 neurons in dense hidden layer was chosen as it got highest accuracy score of 0.82 on validation data. For a final model, model with chosen parameters were trained on train and validation dataset and validated on test set, model with highest validation accuracy = 0.85 was chosen as a final model.

## Language models

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

### N-gram model

N-gram model was trained and validated for a parameters mentioned in Table 1. Highest average score = 0.704 was obtained with parameters: n = 3, temperature = 0.6, k = 10, p = 0.6.

### LSTM model

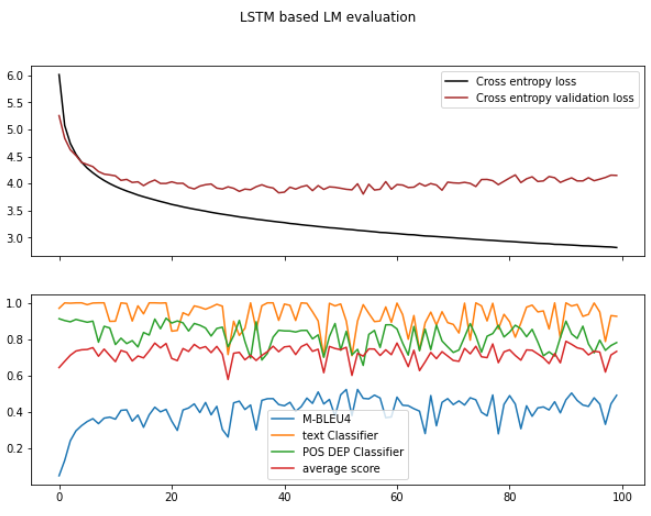


Figure 8. LSTM based LM training scores

While training LSTM based language model (Figure 8), on every epoch 10 generated texts were evaluated with M-BLEU4 score, “textClassifier” and “POS\_DEP classifier”. Model was chosen by highest average score which was 0.79156 at epoch 90. After that sampling parameters, temperature, K and P were tuned using grid search. Best score = 0.8078 was then temperature = 0.8, k = 5, p = 0.6.

### GPT-2 -simple

While training GPT-2 -simple (Figure 9), on every epoch until the 10th and later every 10th epoch, 10 generated texts were evaluated with M-BLEU4 score, “TextClassifier” and POS\_DEP Classifier. Model was chosen by highest average score which was 0.76516 at epoch 6. After that sampling parameters, temperature, K and P were tuned using grid search. Best score = 0.796 was then temperature = 0.6, k = 50, p = 1.

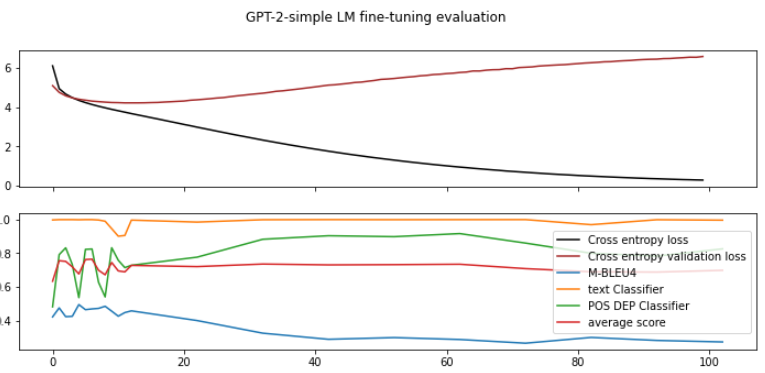


Figure 9. GPT-2-simple fine- tuning training scores

## 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 texts of D. Trump. While N-gram model most of the scores are close to 0. According to 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.

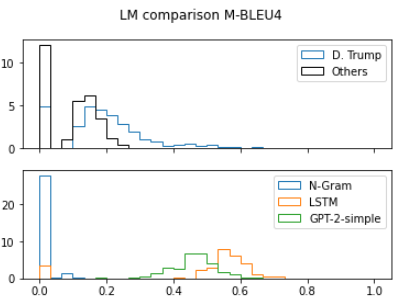


Figure 10. LM comparison: M-BLEU4 scores

Comparing style strength by “textClassifier” scores (Figure 11), all models performed reasonably well compared to the scores obtained with D.Trump speeches. With a same significance there is no evidence to reject that distributions of scores obtained by LSTM based model and GPT-2- simple, are same with a scores from Trump speeches (Table 3).

Distributions of scores obtained by POS-DEP Classifier (Figure 12) by the shape looks very similar for all the language models. But according to mann-whitney U test (Table 3), with a significance level of 0.05, null hypothesis that distributions are equal must be rejected.

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 | 0.0019 | 3.97e-41 | 0.00077 | 3.9e-32 |
| LSTM | 1.4e-30 | 5.05e-34 | 0.1056 | 4.1e-38 | 0.014 | 4.5e-32 |
| GPT-2-simple | 1.6e-44 | 5.2e-56 | 0.4389 | 1.4e-40 | 1.3e-05 | 7.3e-33 |

# Discussion

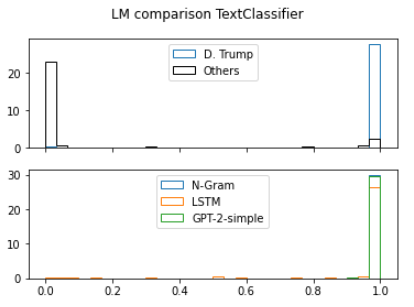


Figure 11. LM comparison: TextClassifier scores

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 M-BLEU4 score will capture quality and style classifiers - style strengths. However, 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 task [12]. Also, similar evaluation is done by M-BLEU4 score as it captures n-gram precision. Thats why it’s not surprising that most of the distributions of “textClassifier” probabilistic predictions are equal to the distribution of the probabilistic prediction of real D. Trump speeches.

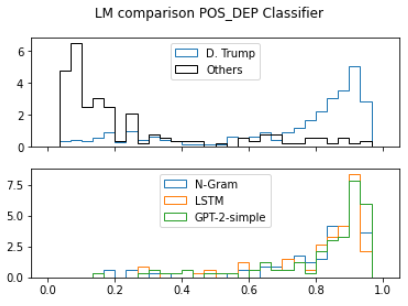


Figure 12. LM comparison: POS\_DEP classifier scores

Another limitation of this project is using trained style classifiers for model validation and not only for evaluation of trained and fine-tuned language models, because then it can be said that data which was used for training classifiers was already used and results of models with tuned parameters are biased. To solve this bias problem, it would be necessary to train style classifiers on other, not used dataset before evaluating final models. Also, as it seen it Figure 10, M-BLEU4 scores of real D. Trump rally speeches are not high and chosen methodology to select a model which maximises average score (Equation 7) is not correct and it would be better to check which model minimizes cross entropy loss.

# 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 average score (Equation 7). Style was evaluated regarding quality and style strength, where quality was evaluated with M-BLEU4 score, for style strength 2 classifiers were trained where one is based on word counts (accuracy = weighted avg. F1 = 0.92) and another on sequence of part of speech tags and their dependencies (accuracy = 0.85). Distributions of scores which were obtained from model generated texts were compared with test real D. Trump and other speeches. Results shown that regarding M-BLEU4 score and POS\_DEP classifier probabilistic predictions, none of the distributions are equal to real D. Trump rally speech score distribution (with a significance level of 0.05). While regarding “TextClassifier” probabilistic prediction distributions, LSTM based and GPT-2-simple language model generated texts are equal to real D. Trump rally speeches. However, it could be argued about the use of word count based models for style strength evaluation in case of language generation and not text-style transfer task for which it was originally used.

To conclude, with a significance of 0.05, only LSTM based and GPT-2-simple language models are capable to generate texts which, regarding style strength by naïve bayes classifier evaluation, would be equal to the texts/speeches of an author on which language models are trained or fine-tuned.

# 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] | 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 {A}merican 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. |
| [15] | 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. |
| [16] | L. Y. Yu, W. Zhang, J. Wang and Yong, *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient,* 2017. |
| [17] | C. Lillelund, *Donal Trump Rally Speeches,* Kaggle. |