Fall 2020 Introduction to Natural Language Processing
Project Final Report

A. Project Team members

1. Tarun Thota
2. Salma Roohi Khayum
3. Bharadwaj Tirunaguru
4. Shaik Kamal Mohammed Adil

B. Project Topic Introduction

Natural-language generation (NLG) is a subfield of Natural Language Processing (NLP) which uses Artificial Intelligence that transforms structured data into natural language and generates human-readable text from computer. NLG is used across a wide range of NLP tasks such as Machine Translation, Speech-to-text, chatbots, Auto-Correct, or text Auto-Completion.

C. Prior work

Team member 1(Tarun Thota): I have gone through an article which talks about the rise of Fake reviews. This article (https://www.rws.com/insights/rws-moravia-blog/the-state-of-natural-language-generation-in-content-creation/#) discusses mainly on how NLP and AI is used by companies to handle and spot the fake reviews and prevent them from having heavy-risk on the products which they sell.

Team member 2(Salma Roohi Khayum): I have written a sample research paper in IEEE format on "NLP techniques used in modern day applications". This paper discusses the concepts like: categorization of oncological response in radiology reports (https://doi.org/10.1007/s10278-017-0027-x), Semantic inference for extracting crime information from text, NLP based sentiment analysis on Twitter data.

Team member 3(Bharadwaj Tirunagaru): I have read the IEEE paper on 'A Methodology for Generating Natural Language Paraphrases' and understood a few concepts on NLG. This idea is similar to the one used in Grammarly. The paper can be viewed from the link attached 'https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7785363'.

Team member 4(Kamal Mohammed Adil Shaik): I have written a sample IEEE paper on 'Introduction to NLP, Techniques involved and Applications of NLP' as part of an assignment by referring multiple IEEE papers on the same as part of my Machine Learning course curriculum. I have read about Natural Language Generation and different models used in generating NLG prior to writing the sample Research paper as part of my Machine Language curriculum. One of the citations from which my paper was based the most is attached below - 'https://www.researchgate.net/publication/325772303_Natural_Language_Processing'.

We have performed statistical data modelling till now in our Course curriculum, Now we would like to do a project on Neural Language Model and develop an RNN-LSTM based language model to perform Natural Language Generation using Pytorch in three phases. So, we have chosen Shaik Kamal Mohammed Adil's topic of NLG and would like to work on the topic of RNN-LSTM model.

D. Data sources

Team member 1(Tarun Thota): I propose Football Commentary data for our project as it is publicly available and suits well for our project in the text generation. Source: Kaggle (https://www.kaggle.com/kerneler/starter-football-commentary-data-set-effe076a-0)

Team member 2(Salma Roohi Khayum): Wiki movie plot. I propose this dataset since this has a large amount of data. This type of data is well suited for summarization and Natural language generation. Source: Kaggle.

Team member 3(Bharadwaj Tirunagaru): OpenSubtitles DataSet. I propose this dataset because it has subtitles for a large number of movies and television shows in many languages. English dataset from OpenSubtitles is sufficiently large and well suited for Natural Language Generation. The advantage of this dataset is the model can be easily extended to 62 languages available in the OpenSubtitles
Dataset (Source: http://opus.nlpl.eu/OpenSubtitles-v2018.php)

Team member 4(Kamal Mohammed Adil Shaik): I have proposed the Sentiment140 twitter tweets dataset used in sentiment analysis consisting of 1.6 million tweets. Source - Kaggle. (https://www.kaggle.com/kazanova/sentiment140)

Based on the data source gathered from all the teammates, we have decided to use Football Commentary data for our project as it is sufficiently large and is well suited for the title and topic of our project

E. Approach

1) Explain what is your solution and how you did it. Which techniques did you use?

   Solution: In our project, to generate Natural Language we used a recurrent neural network (LSTM), using PyTorch as our solution for completing the project. The techniques used are - LSTM - Long Short Term Memory model which is a Recurrent Neural Network model PyTorch - Open Source Deep Learning Framework

2) How did you measure performance?

Solution: We have used Bleu Score and Rogue Scores as Performance Metrics to evaluate our model.
BleuScore: Bilingual Evaluation Understudy is the score which is used to compare a candidate translation of text to one or more reference translations.
RogueScores: (Recall-Oriented Understudy Gisting Evaluation) is a recall-based evaluation metrics which measures how much of the reference summary is captured within the generated summary.

3) Also explain team member contributions here: which team member contributed to which part of the problem.

Team Members	**Contributions**	**Software Library/Packages and data used**
Shaik Adil	Contributed majorly in choosing Performance Metrics for the model and generating Metrics Performance Results using our Language model. Performed Testing of Model by using different sizes of dataset and summarized the 'Text Generated', ROGUE and BLEU scores of our model for different sizes of dataset. Minor contributions during DataPreProcessing and DataExploration and Model Building of our project.	Data: rawTranscripts.json Packages: nltk, rouge, torch, bleu_score,rouge_score,
Tarun Thota	Contributed to the data cleaning and preprocessing phase of the data. Generalised a global parameter to specify how many sentences(transcripts) we are using as each transcript has approximately 15000 words. Have also contributed mainly by examining the length of each transcript(sentence), then worked on creating a fixed length of sentence(n-grams)	Data: rawTranscripts.json Packages: nltk, pandas,random, numpy, torch, matplotlib

	Contributed to the data model, and implemented a way to calculate the loss function Generated the plots for Zipf’s law, loss function and metrics	
Bharadwaj	Contributed to data cleaning and preprocessing, mainly trimmed the dataset to use 50 transcripts randomly from 1455 transcripts because each transcript consists of around 15000 words which took a long time while training the model with full dataset and observed good accuracy. Then transformed all the transcripts in our dataset into lowercase and removed punctuation to improve the accuracy of the model and handled token to integer and integer to token mapping and converted the text sentences to integer sequences. Contributed in building the model and optimizing the model. Contributed to text generation using the trained model to generate a sentence with given size or with given words	Data: rawTranscripts.json Packages: re, random, pandas, sklearn, numpy, torch
Salma Roohi	I contributed in Building LSTM Model, Model and hyper parameter tuning and plotted visualization for performance metrics of the model. Built a model by inheriting from a neural network module , and then also instantiated its parameters and weight initializations. Implemented the model for different sizes of the learning rate, hidden size, batch size.	Data: rawTranscripts.json Packages: nltk, pandas,random, numpy, torch, matplotlib

	Also implemented a parameter to specify how many dimensions will be used to represent each word. That is, the size of the embedding vector space. During training, plot the summary of performance. The graph of BLEU scores for each parameter set is calculated and compared. At the end of the run, the trained model is saved to a file which is used to compare the results. Choose additional Hyper-Parameters, Dropout layers. This layer helped to prevent overfitting. I was actively involved in the entire process of the project while constantly learning and implementing ways to improve the model performance.	

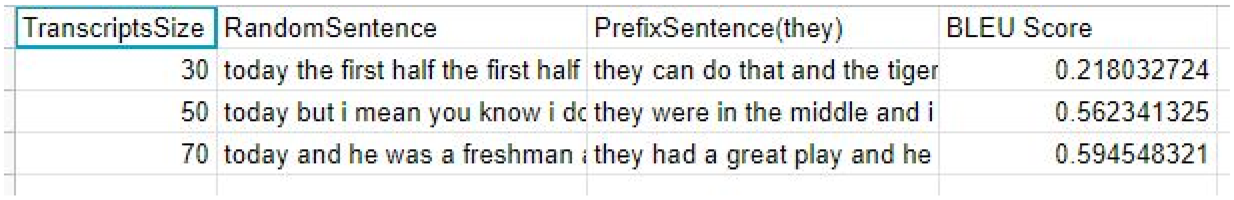
F. Results

Observations:

1. **Transcripts Size effect on Model:**

   We have observed how the number of transcripts have affected the Bleu Score and the sentences generated. The number of transcripts have an effect on the Bleu Score. When more transcripts are used for training the model. The Bleu Score is higher. We have created a report containing a summary of results for 3 different numbers of transcripts size.

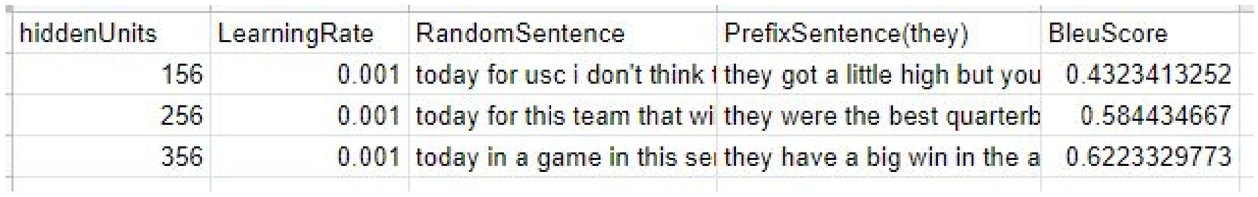
   The report can be accessed using the link here



TranscriptsSize	RandomSentence	PrefixSentence(they)	BLEU Score
30	today the first half the first half	they can do that and the tiger	0.218032724
50	today but i mean you know i dc	they were in the middle and i	0.562341325
70	today and he was a freshman a	they had a great play and he	0.594548321

2. **Observations on HyperParameter_HiddenUnits**

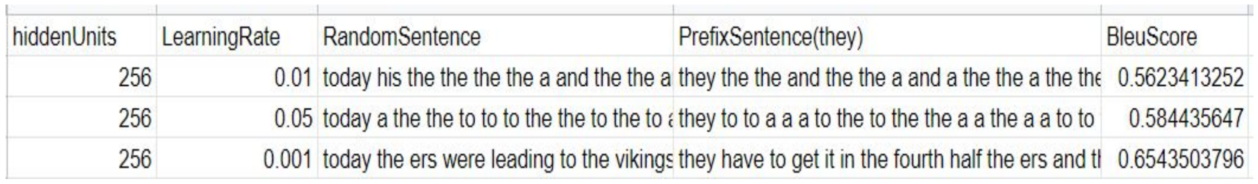
   We tried to tune our model based on the different number of hidden units. The observations are tabulated here. As more number of hidden units were introduced, the model took longer for its execution but fairly improved the performance. Our model has performed best with 0.622 BLUE score and 356 hidden units



hiddenUnits	LearningRate	RandomSentence	PrefixSentence(they)	BleuScore
156	0.001	today for usc i don't think t	they got a little high but you	0.4323413252
256	0.001	today for this team that wi	they were the best quarterb	0.584434667
356	0.001	today in a game in this se	they have a big win in the a	0.6223329773

3. **Observations on HyperParameter_LearningRate**

   We tried to tune our model based on a different number of Learning Rate. The observations are tabulated here. As the Learning Rate decreased, the model took longer for its execution but fairly improved the performance. Our model has performed best with 0.654 BLUE score and 0.001 Learning Rate.



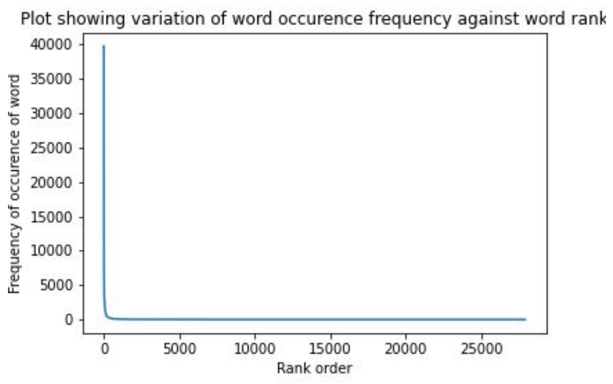
hiddenUnits	LearningRate	RandomSentence	PrefixSentence(they)	BleuScore
256	0.01	today his the the the the a and the the a	they the the and the the a and a the the a the the	0.5623413252
256	0.05	today a the the to to to the the to the to a	they to to a a a to the to the the a a the a a to to	0.584435647
256	0.001	today the ers were leading to the vikings	they have to get it in the fourth half the ers and tl	0.6543503796

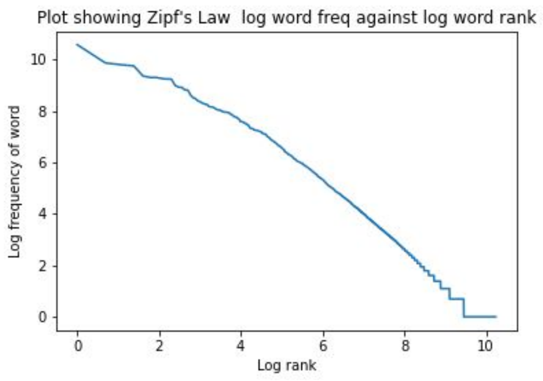
Measurements and Statistics:

Zipf's Law:

Here we have compared the frequency and rank of a word, and compared both in a plot and check if the data validates the Zipf's law or not.

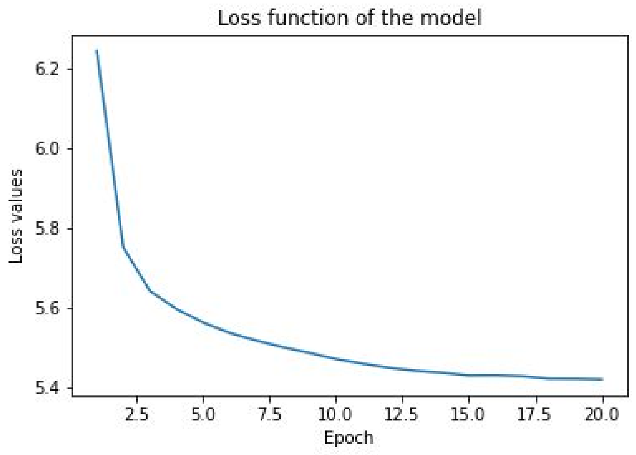
Below are the plots generated with the dataset used to generate the plots for Zipf's law.





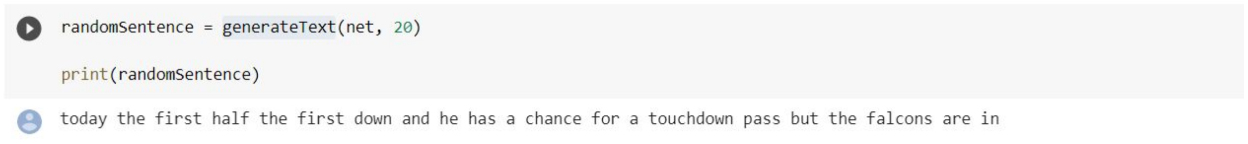
Loss Function:

While training the model, we take the loss values obtained from each and every Epoch, and get the average values and plot it by taking the epoch in x-axis and Loss values in the y- axis. We can see that the loss function rapidly decreases as the epoch increases.



Text Generation:

Generated a sentence of 20 words using the trained model when size 20 is passed to the text generation function generateText.

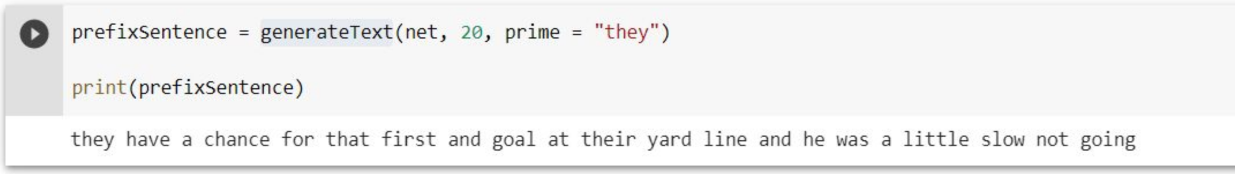


```
randomSentence = generateText(net, 20)

print(randomSentence)
```

```
today the first half the first down and he has a chance for a touchdown pass but the falcons are in
```

Generated a sentence of 20 words with a prefix word “they” using the trained model when size 20 and prefix word “they” are passed to the text generation function generateText.



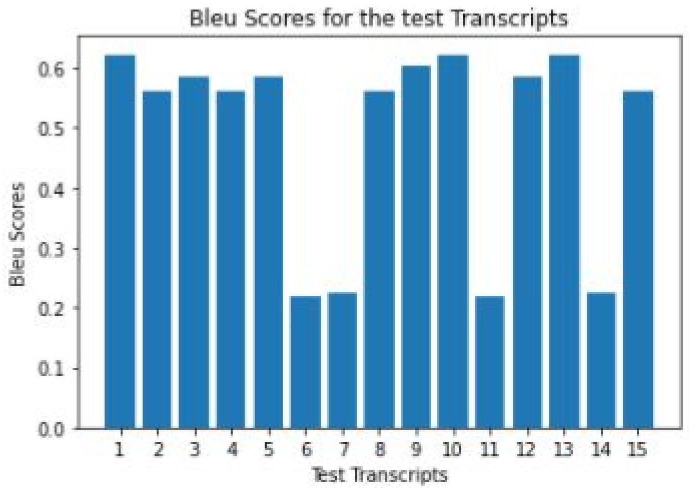
```
prefixSentence = generateText(net, 20, prime = "they")

print(prefixSentence)
```

```
they have a chance for that first and goal at their yard line and he was a little slow not going
```

Bleu Scores:

BLEU scores are based on an average of unigram, bigram, trigram and 4-gram precision and provide an overall assessment of model quality.. After the model is trained we take the test transcripts and generate bleu scores for each and every transcript generated for each and every sentence and plot it in the form of Bar Plot by taking test transcripts in x-axis and bleu scores in the y axis. Here, we can see that the average bleu score is around 0.56.



Rouge Score:

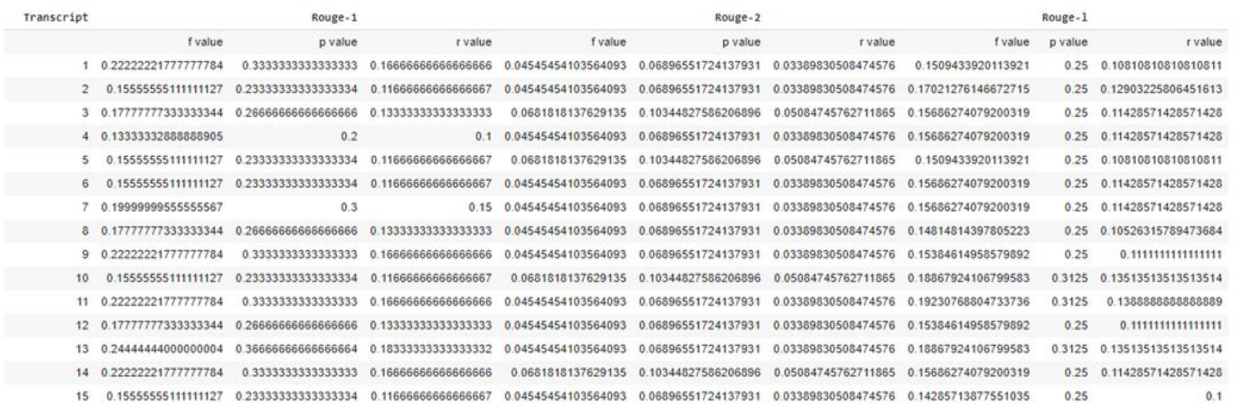
For each and every transcript, we generate the rogue score by using the model and generate it in a tabular view.

ROUGE (Recall-Oriented Understudy Gisting Evaluation) is a recall-based evaluation metric which measures how much of the reference summary is captured within the generated summary.

ROUGE-1 - measures the overlap of unigrams between the system summary and reference summary

ROUGE-2 - measures the overlap of bigrams between the system summary and reference summary

ROUGE-L - measures the longest matching sequence of words using LCS between the generated text and reference text.



Transcript	Rouge-1			Rouge-2			Rouge-1		
	f value	p value	r value	f value	p value	r value	f value	p value	r value
1	0.22222221777777784	0.3333333333333333	0.16666666666666666	0.045454541035640933	0.0689655172413793	0.03389830508474576	0.150943392011392 1	0.25	0.10810810810810811
2	0.1555555511111127	0.23333333333333334	0.11666666666666667	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.1702127614667271 5	0.25	0.12903225806451613
3	0.17777777333333344	0.26666666666666666	0.13333333333333333	0.0681818137629135	0.10344827586206896	0.0508474576271186 5	0.15686274079200319	0.25	0.1142857142857142 8
4	0.13333332888888905	0.2	0.1	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.15686274079200319	0.25	0.1142857142857142 8
5	0.1555555511111127	0.23333333333333334	0.11666666666666667	0.0681818137629135	0.10344827586206896	0.0508474576271186 5	0.150943392011392 1	0.25	0.10810810810810811
6	0.1555555511111127	0.23333333333333334	0.11666666666666667	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.15686274079200319	0.25	0.1142857142857142 8
7	0.19999999555555567	0.3	0.15	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.15686274079200319	0.25	0.1142857142857142 8
8	0.17777777333333344	0.26666666666666666	0.13333333333333333	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.14814814397805223	0.25	0.10526315789473684
9	0.22222221777777784	0.3333333333333333	0.16666666666666666	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.15384614958579892	0.25	0.1111111111111111
10	0.1555555511111127	0.23333333333333334	0.11666666666666667	0.0681818137629135	0.10344827586206896	0.0508474576271186 5	0.18867924106799583	0.3125	0.13513513513513514
11	0.22222221777777784	0.3333333333333333	0.16666666666666666	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.19230768804733736	0.3125	0.1388888888888889
12	0.17777777333333344	0.26666666666666666	0.13333333333333333	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.15384614958579892	0.25	0.1111111111111111
13	0.24444444000000004	0.36666666666666664	0.18333333333333332	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.18867924106799583	0.3125	0.13513513513513514
14	0.22222221777777784	0.3333333333333333	0.16666666666666666	0.0681818137629135	0.10344827586206896	0.0508474576271186 5	0.15686274079200319	0.25	0.1142857142857142 8
15	0.1555555511111127	0.23333333333333334	0.11666666666666667	0.04545454103564093	0.0689655172413793 1	0.03389830508474576	0.14285713877551035	0.25	0.1

G. Summary

Natural-language generation (NLG) is a subfield of Natural Language Processing (NLP) which uses Artificial Intelligence that transforms structured data into natural language and generates human-readable text from computer. NLG is used across a wide range of NLP tasks such as Machine Translation, Speech-to-text, chatbots, Auto-Correct, or text Auto-Completion. We have analysed different models like RNNs and RNNs LSTM for Natural-language generation. RNNs cannot store words that were encountered remotely in the sentence and make predictions based on only the most recent word. RNN based Long short-term memory (LSTM) solves the problem of long-range dependencies. In real time scenarios, the RNN-LSTM model is found to be one of the highly efficient Neural Network models when compared to traditional statistical language models for Natural Language Generation. Hence, We used the RNN LSTM model for Natural-language generation in our project. For our model, we have used football transcripts to train and test the model. We have performed DataCleaning and Data-PreProcessing and generated fixed-length sentences to pass to the model. The model is built with optimal hyperparameters like learning_rate and hidden_units followed by training the model. We have checked the efficiency of the model Finally, the Text generation was performed from the model where the model can generate sentences of given size or sentences starting with any prefix word. We have tuned the model with different hyperparameters to improve the performance of the model and stored the results.

Important Conclusions -

From the results, we conclude with the following observations

1. The dataset follows the Zipf's law
2. The loss values gradually decrease as the epochs increase
3. After hyper parameter tuning, , we observed that the model performed better with good rogue and bleu scores when hidden units=256, learning rate=0.001 when used as hyperparameters.

H. References

1. https://www.rws.com/insights/rws-moravia-blog/the-state-of-natural-language-generation-in-content-creation/#
2. https://doi.org/10.1007/s10278-017-0027-x
3. ‘https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7785363’.
4. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7785363’
5. https://www.researchgate.net/publication/325772303_Natural_Language_Processing
6. https://towardsdatascience.com/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66
7. https://www.analyticsvidhya.com/blog/2020/08/build-a-natural-language-generation-nlg-system-using-pytorch/