POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Name Sankalp Apharande
UNI: spa2138

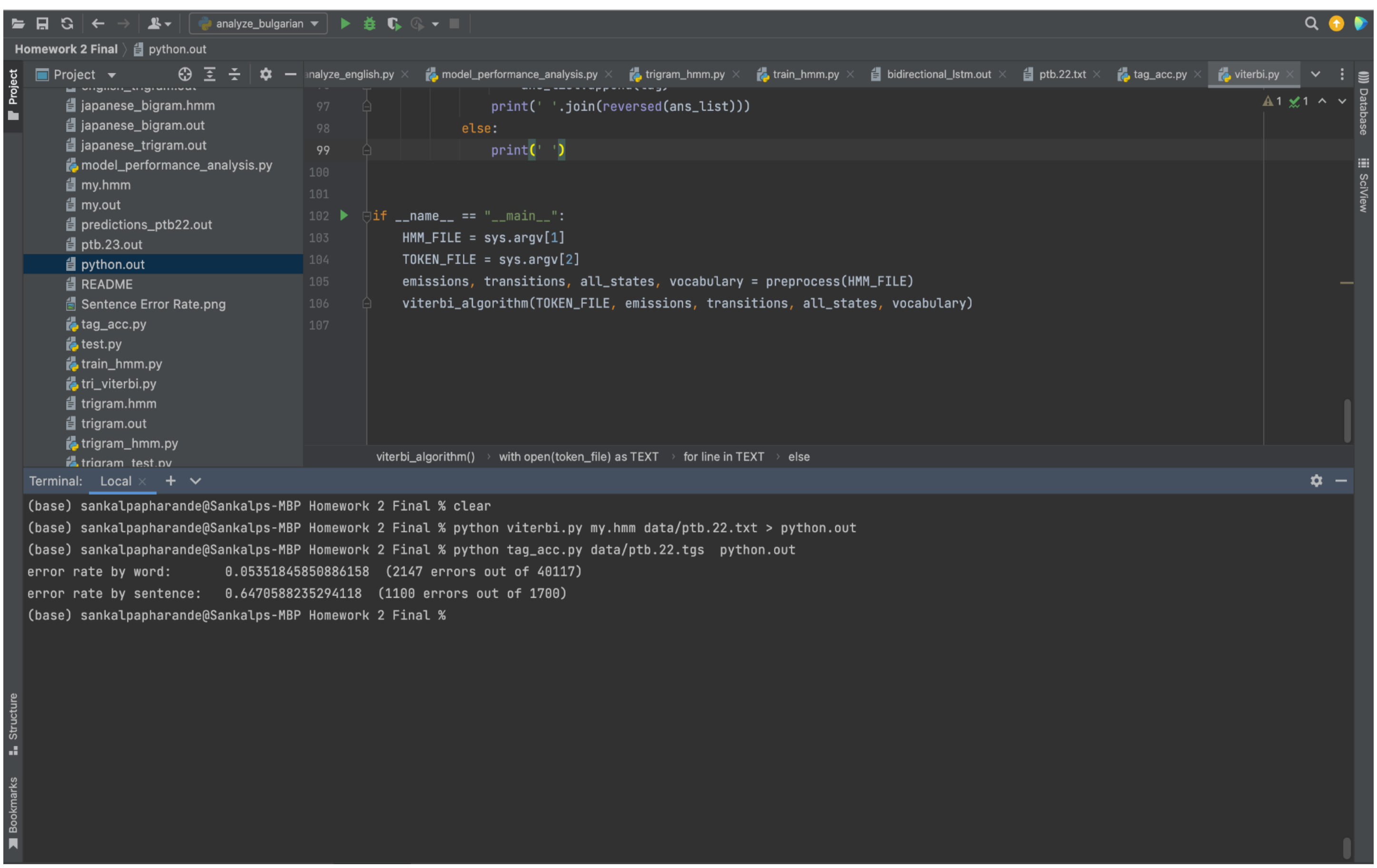
Task 1:

Part 1:

Viterbi.py contains the Viterbi algorithm for POS Tagging using Bigrams in Python.

Results on the output of the Viterbi python algorithm
error rate by word: 0.05351845850886158 (2147 errors out of 40117)
error rate by sentence: 0.6470588235294118 (1100 errors out of 1700)

It is slightly better than the Perl version.



POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Part 2:

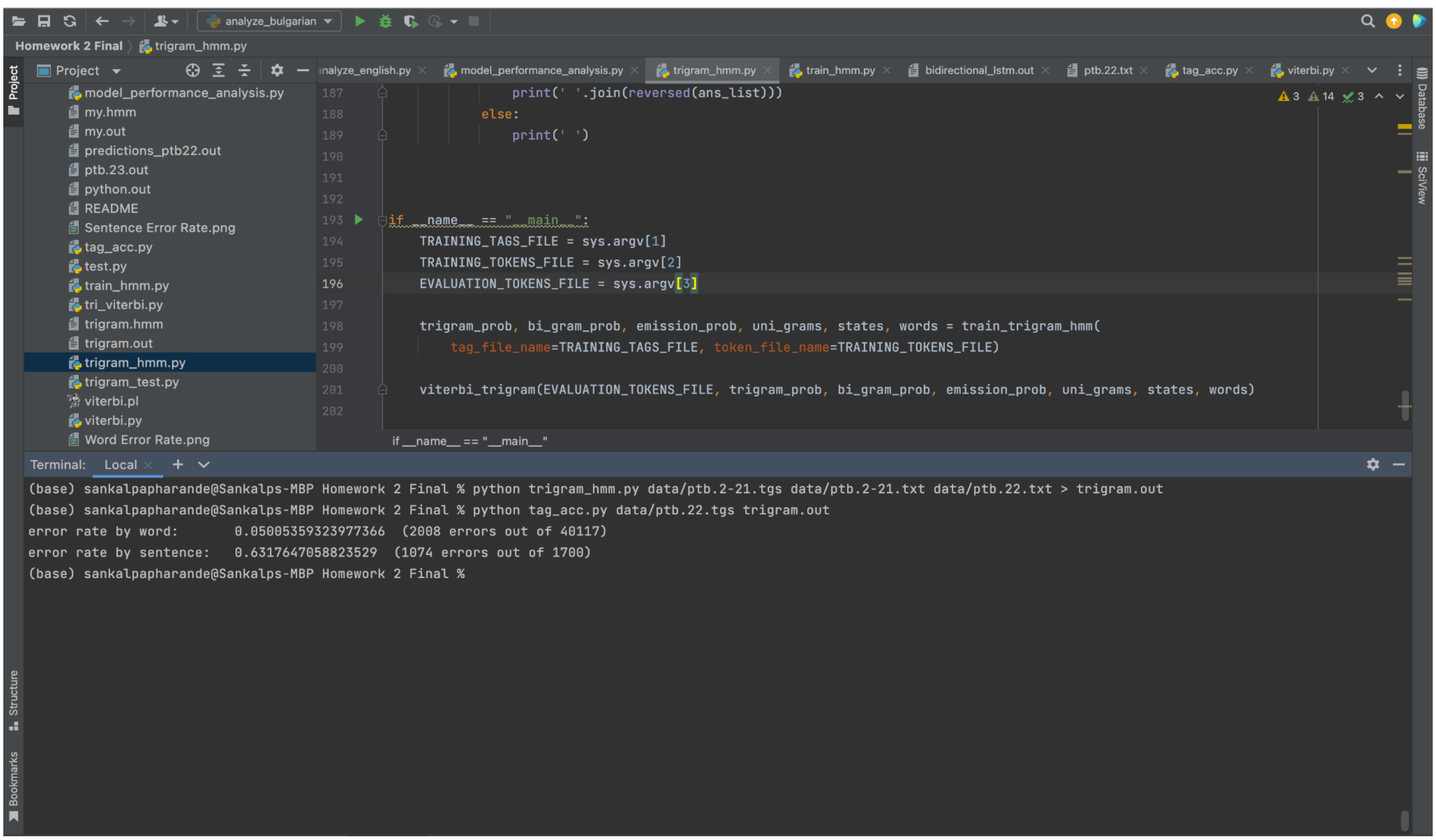
trigram HMM and Trigram Viterbi

Trigram_hmm.py contains training for generating trigram HMM and the corresponding Viterbi algorithm which uses Trigram HMM.

I have used a simple backoff model. It is giving better performance than naive trigram
To generate tags for ptb.23.txt; Run:

python trigram_hmm.py data/ptb.2-21.tgs data/ptb.2-21.txt data/ptb.23.txt > ptb.23.out

Results: model's performance on the development data (ptb.22.txt)
error rate by word: 0.05005359323977366 (2008 errors out of 40117)
error rate by sentence: 0.6317647058823529 (1074 errors out of 1700)

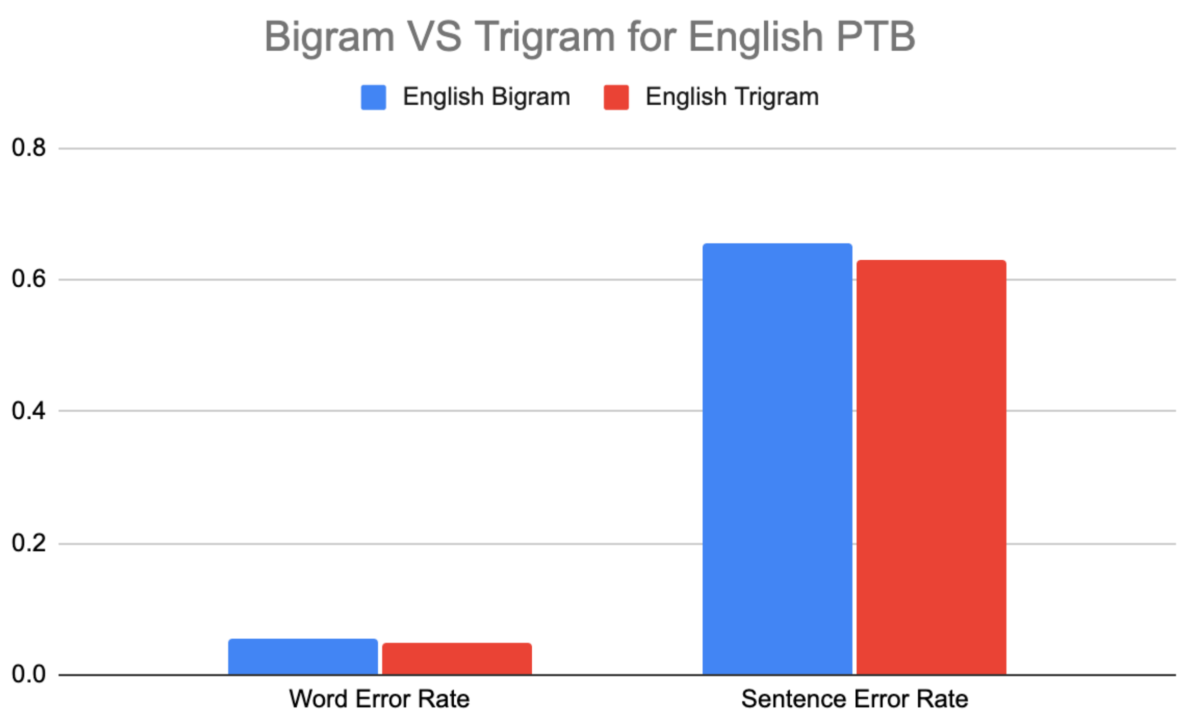


POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Bigram vs Trigram comparision for ENG Penn Tree Bank dataset:

Trigram is better than Bigram:

	English Bigram	English Trigram
Word Error Rate	0.05409178154	0.05005359324
Sentence Error Rate	0.6558823529	0.6317647059

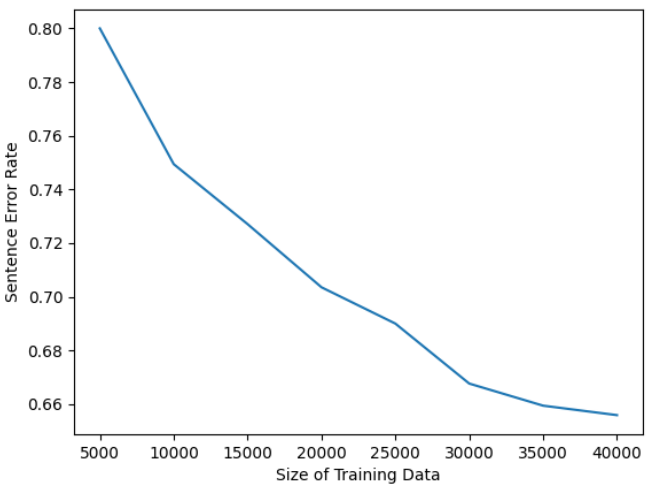


POS Tagging: Viterbi Algorithm and Deep Learning Comparision

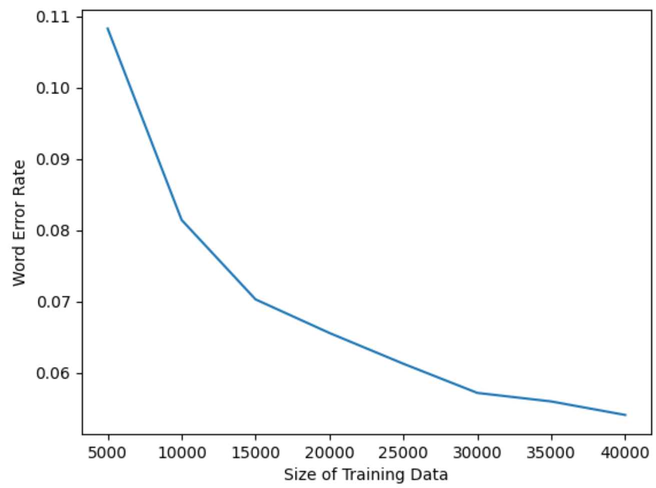
Task 2

Generating learning curve for bigram HMM evaluation
In the code, to generate curves, run

python model_performance_analysis.py



Sentence Error Rate vs Size of Training Data



Word Error Rate VS Size of Training Data

Observation:

1. Both sentence error rate and Word error rate decreases as the size of training data increases.
2. As expected, getting more training data is always helpful.
3. Getting more POS-tagged data would help decrease error rates
4. But improvement in the model gradually decreases as the training data size increases, i.e decrease in error rate is not as fast as it is initially.

POS Tagging: Viterbi Algorithm and Deep Learning Comparision

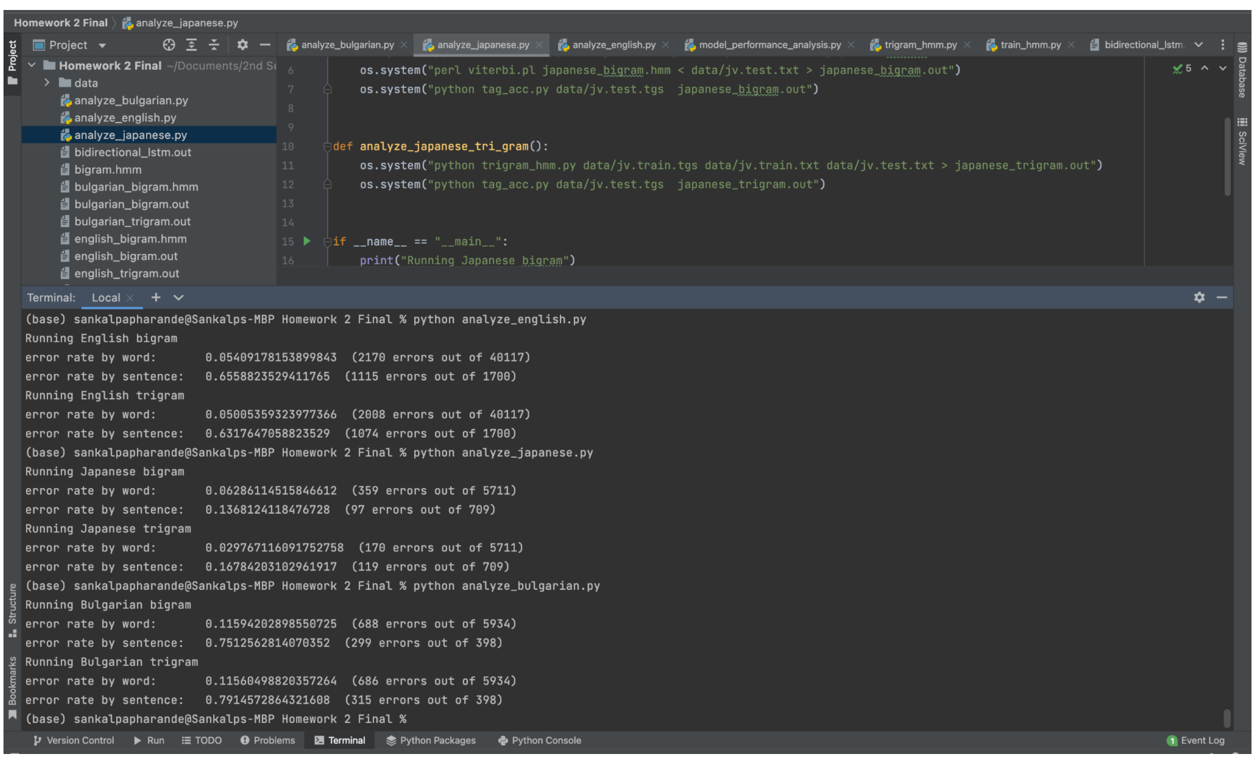
Task 3

Part 1:

Q. Train the bigram model and your trigram model on the Japanese (jv.*) and Bulgarian (btb.*) training datasets, and compare them on the test sets. Report the performance and compare the performance differences between English, Japanese, and Bulgarian.

I have created separate python scripts to analyze the performance of these 3 languages.

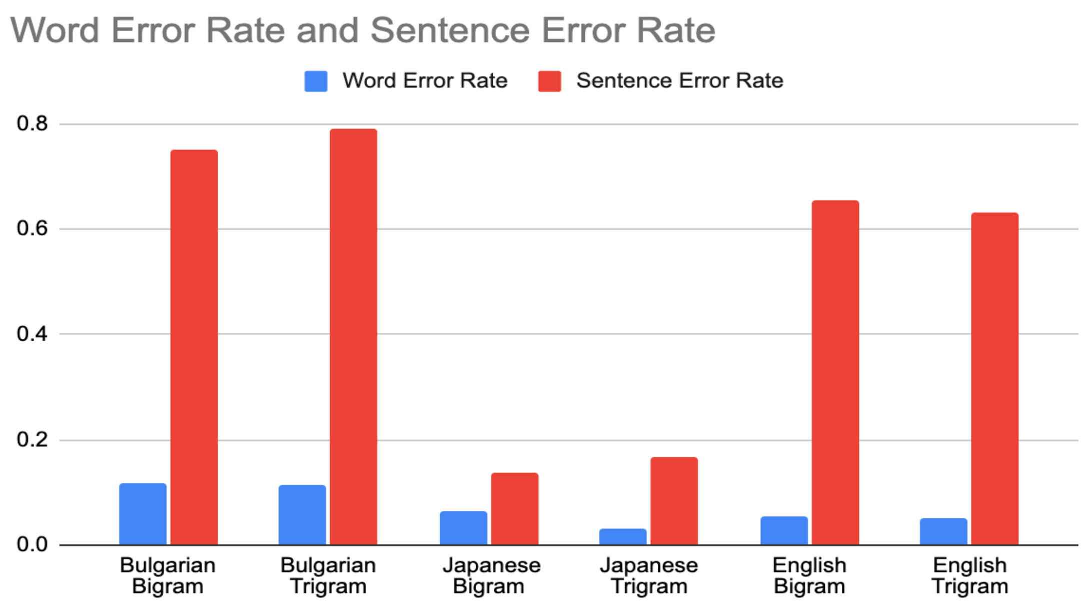
1. English: To generate bigram and Trigram output for English
   Run:
   python analyze_english.py
2. Japanese: To generate bigram and Trigram output for Japanese
   Run:
   Python analyze_japanese.py
3. Bulgarian: To generate bigram and trigram output for Bulgarian.
   Run:
   Python analyze_bulgarian.py



POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Performance differences between these languages:

	Bulgarian Bigram	Bulgarian Trigram	Japanese Bigram	Japanese Trigram	English Bigram	English Trigram
Word Error Rate	0.115942029	0.1156049882	0.06286114516	0.02976711609	0.05409178154	0.05005359324
Sentence Error Rate	0.7512562814	0.7914572864	0.1368124118	0.167842031	0.6558823529	0.6317647059



Observations:

1. Both Bigram and Trigram perform best on Japanese Data
2. Performace is worst for Bulgarian Language
3. Trigram performs better for English than corresponding Bigram
4. For Bulgarian, Trigram performs worse than bigram

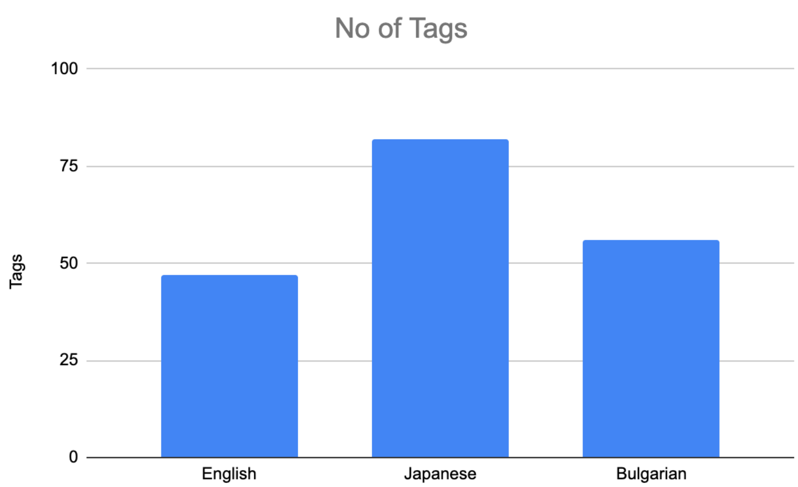
POS Tagging: Viterbi Algorithm and Deep Learning Comparision

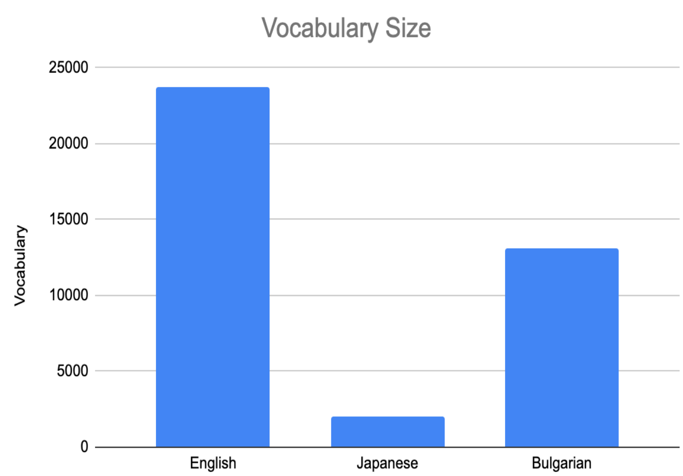
Part 2

You sould look at the data and give some analysis – what factors lead to the differences among performance on English, Japanese, and Bulgarian? (

Data Analysis:

	English	Japanese	Bulgarian
Tags	47	82	56
Vocabulary	23768	2030	13122





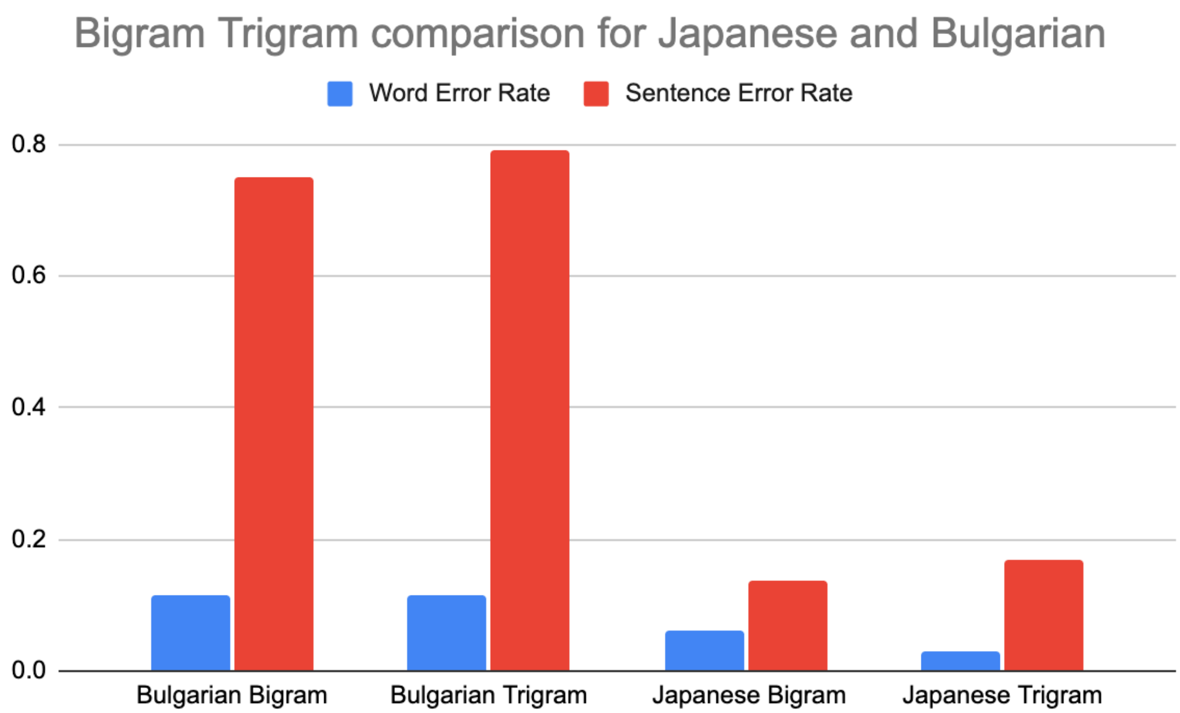
Observations:

1. Japanese has the highest number of tags 76 and smallest vocabulary size among given three languages.
2. Markov assumption is holding much better for Japanese, possibly because In English, a sentence is written word-by-word. On the other hand. A sentence in Japanese has no word boundary character. (Source)
3. In Bulgarian Language, the Markov assuption is not that strong.
4. English is relatively better because enough words in English sentence depends on previous word. Even better when we use trigram
5. Hence HMM Bigram has best performance for Japanese, then English and worst for Bulgarian

POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Part 3

what about the trigram HMM makes them relatively better or worse on the data
in these two languages? Explain your result.
Trigram comparison for Japanese and Bulgarian:



Observation:

1. Bigram performs better for Japanese and Bulgarian
   Reason:
   a. Markov assumption is holding much better for Japanese. A sentence in Japanese has no word boundary character. (Source)
   b. In Bulgarian Language, the Markov assuption is not that strong.

2. Both Bigram and Trigram perform better for Japanese than Bulgarian
   Reason:
   a. Both in Japanese and Bulgarian, long dependencies are not helpful. It could be because there are no word boundaries in Japanese and Bulgarian like English
3. Word Error is smaller in trigram than Bigram for Japanese
   Reason:
   a. Trigram is helpful to identify the tags but not good to identify the sequence of tags
4. Sentence Error rate are higher in Trigram for both Japanese and Bulgarian
   Reason
   a. Markov assumption is not that strong in these Bulgarian and long dependencies are not helpful in identifying sequence

POS Tagging: Viterbi Algorithm and Deep Learning Comparision

Bonus Task:

1. Implemented Bidirectional LSTM with Word Embeddings are initial weights for English
2. Implemented Bidirectional LSTM without word Embeddings for Japanese and Bulgarian because I could not find word embeddings in these languages

Resons for LSTM:

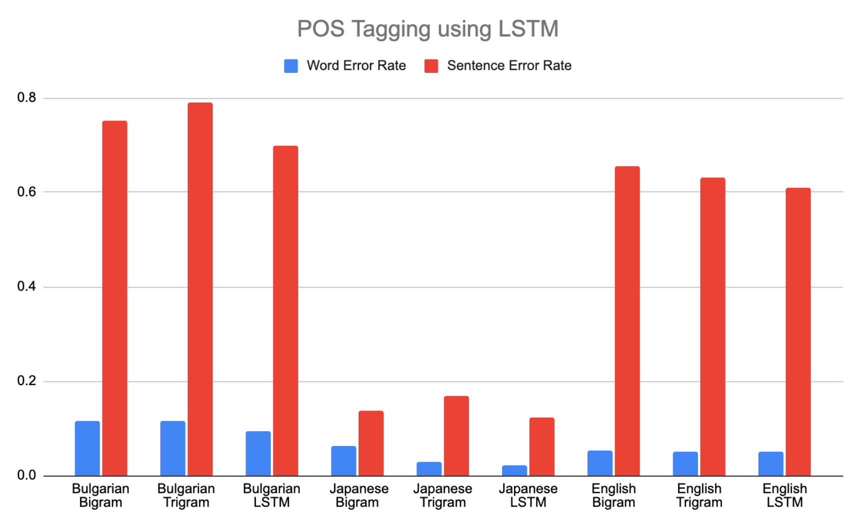
1. LSTM are a form of Recurrent Neural Network which is helpful in learning many to many sequence learning
2. LSTM handles Vanishing and Exploding gradient problem by implementing forget gate and Add Gate
3. Birdirectional learns from both the ends hence good for POS Tagging

Colab Notebooks can be found here:

Reference: https://nlpforhackers.io/lstm-pos-tagger-keras/

Methodology:

1. English:
    a. Trained and Validated using data/ptb.2-21.tgs data/ptb.2-21.txt/ and
    b. Evaluated on data/ptb.22.txt and data/ptb.22.tgs
2. Japanese:
    a. Trained and Validated using data/jv.train.tgs data/jv.train.txt
    b. Evaluated on data/jv.test.txt and data/jv.test.tgs
3. Bulgarian:
    a. Trained and Validated using data/btb.train.tgs data/btb.train.txt
    b. Evaluated on data/btb.test.txt and data/btb.test.tgs



Observations:

1. Bidirectional LSTM Performs better than both bigram and trigram for all three languages
2. Both Sentence Error Rate and Word Error Rate is lowest using LSTM for all three languages.

POS Tagging: Viterbi Algorithm and Deep Learning Comparision

	Bulgarian Bigram	Bulgarian Trigram	Bulgarian LSTM	Japanese Bigram	Japanese Trigram	Japanese LSTM	English Bigram	English Trigram	English LSTM
Word Error Rate	0.115942029	0.115604988	0.094202	0.062861	0.029767	0.0211871	0.0540917	0.0500535	0.04997353687
Sentence Error Rate	0.7512562814	0.79145728	0.698492	0.136812	0.167842	0.122708	0.6558823	0.6317647	0.6088390501

Acknowledgement and References:

1. https://github.com/KangboLu/Natural-Language-Processing
2. https://github.com/nadiahyder/POS-Tagger
3. https://www.kaggle.com/code/tanyadayanand/pos-tagging-using-rnn/notebook
4. https://www.pythonpool.com/viterbi-algorithm-python/