Section 1:

1. Present your chosen data, does it have any special attributes that you expect to have an influence on the text generation of the model?

   I chose the *Friends* TV show, my favorite American sitcom, from the Cornell Movie Dialogs Corpus as the dataset for this project. Given the corpus's large size, I wrote a Python script to download it and limit it to 8,000 dialogue segments for efficient processing. The language in TV show dialogues is highly conversational, concise, and informal. For example, phrases like "C'mon" and "you're" appear frequently. These utterances reflect spontaneous, everyday speech, and are closely associated real-life situations.

   As a result, the model trained on this dataset tends to generate text in a dialogue format that is natural and conversational. The outputs often are short, capturing the informal tone of spoken language.

2. Take a look at the sample generation, what are your impressions?

   Fristly, the generated texts include the tags "<eos>" and "<unk>" to present the end of the sentence and unkown words. Secondly, the generated texts are tokenized because there is space between punctuation and words. Thirdly, the sentences in the sample is short spoken-style and conversational, indicating it grasps the main characters of the trained texts. More specifically, it also contains some informal languages such as "it's" and "you're".

   Although it mimics the trained text to generate the almost-similar sentence, there is grammatical inconsistence and semantic incoherence. For example, "<eos> Hell , yeah , what died to her ? <eos>", the expression of the latter sentence is not common in English and the whole sample is not semantically cohenrent.

Section 2:

I trained 6 language models with varying dropout rates (0.0, 0.2 0.3, 0.6, 0.8 and 0.9) by changing the train.sh script. I chose a value of 256 for the embedding size and the number of hidden units per layer, and 50 for the epochs. The 6 models were training on the data prepared in Task 1.

1. Can you see a connection between the training, validation and test perplexity? Based on your results, which dropout setting do you think is the best and why?

   Yes, a clear relationship can be observed between training, validation, and test perplexity across different dropout rates. Across all models, training perplexity consistently decreases over epochs, particularly at lower dropout levels (e.g., 0.0 and 0.3), suggesting effective learning. However, very low training perplexity at these settings may indicate overfitting.

   For low dropout rates (0.0, 0.3), validation perplexity often plateaus or worsens after early epochs, further suggesting overfitting. Dropout values around 0.6 offer a better trade-off: training perplexity steadily declines, and validation perplexity also decreases, indicating improved generalization. In contrast, at very high dropout (e.g., 0.9), the model fails to fit the training data effectively, resulting in high training and validation perplexities, characteristic of underfitting.

   The test perplexity is lowest when validation perplexity is also lowest, indicating the importance of good generalization. Interestingly, at a dropout rate of 0.2, all perplexities (training, validation, and test) are much higher than at 0.0. This may suggest that the model neither overfits nor learns meaningful patterns, possibly due to the dropout inadvertently removing crucial features.

In this dataset, the best overall performance is achieved with dropout = 0.0, likely due to strong similarity between the training, validation, and test sets, which allows for generalization without the need for regularization. As dropout increases, test performance generally degrades, suggesting increasing underfitting.

In conclusion, dropout = 0.0 yields the optimal results for this dataset. However, the optimal dropout rate may vary depending on the model architecture and dataset characteristics.

2. Sample some text from the model that obtains the lowest test perplexity, for instance by changing the script scripts/generate.sh. What do you think of its quality? Does it resemble th original training data?

   Although there is incorrect syntax and incohenrent semantic, the generated text roughtly capture the structure and expression in the *Friends* corpus. There are some informal and conversational expressions such as “you know” and “Yeah”. The sentence like “we both have some, a stomach” indicates the syntactic inconsistence. Additionally, the expression such as “intense dollars impression” is meaningless and there is little semantic connection between sentences. The sentence like “we both have some, a stomach” indicates the syntactic inconsistence. In a nutshell, despite acquiring the informal and spoken-style expression from the training data, it still generates the meaningless sentences.

3. Sample some text with the highest test perplexity. Can you see a difference to the lowest scoring one?

   This output is markedly less coherent and has more grammatical error (e.g., “He have work ?”). The model fails to produce logical utterances and even for the letter case use the model does not acquire very well such as “You got Anything did .” It is obvious that high dropout generated text grasps limited features of training data, only the converstationl characteristics. But for the grammatical and syntactically correct expressions, it has a poor performance. The difference between low and high dropout indicates that when with high dropout, the model becomes underfit and unable to learn the patterns completely and effectively. It can not learn the meaningful context features from the training data. Consequently, the output has lots of linguistic mistakes as well as meaningless expressions.

Link to git repositoty: https://github.com/JingClever/mt-exercise-02.git