

HW2 Report

For Part 0:

I used punctuation removal, in this phase, all punctuation from the text is removed. Python's string library includes a pre-defined collection of punctuations such as

'!"#\$%&'()*+,-./:;?@[\\]^_`{|}~'

#library that contains punctuation:

```
import string
string.punctuation
```

#defining the function to remove punctuation:

```
def remove_punctuation (text: str) -> str:
    tokenizer = ToktokTokenizer()
    tokens = tokenizer.tokenize(text)
    tokens = [token for token in tokens if token != '&']
    filtered_tokens = [token for token in tokens if token not in string.punctuation]
    preprocessed_text = ' '.join(filtered_tokens)

    return preprocessed_text
```

We can see in the below output the punctuation is removed:

```
output: here is a dog
● (base) ralphkedywillensbuteau@Ralphs-MacBook-Pro HW2 % python preprocess.py
[nltk_data] Downloading package stopwords to
[nltk_data]   /Users/ralphkedywillensbuteau/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]   /Users/ralphkedywillensbuteau/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]   /Users/ralphkedywillensbuteau/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]   /Users/ralphkedywillensbuteau/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Here is a dog!
output: Here is a dog
○ (base) ralphkedywillensbuteau@Ralphs-MacBook-Pro HW2 %
```

For Part 1:

Briefly explain the concept of perplexity in report and discuss how it will be influenced.

- Perplexity is a notion used to assess the quality of language models. It assesses how well a language model predicts a word sequence. Simply said, confusion is a measure of how shocked a language model is when it encounters fresh text. The lower the perplexity, the better the model.

Perplexity is calculated by taking the inverse probability of the test set normalized by the number of words in the test set. Perplexity is mathematically defined as:

$$\text{perplexity} = 2^{(-\log P(w_1, w_2, \dots, w_n)/n)}$$

where $P(w_1, w_2, \dots, w_n)$ is the probability of the test set and n is the number of words in the test set.

Perplexity is influenced by a variety of parameters, including the size and quality of the training data, the complexity of the language model, and the smoothing techniques used to handle rare or unknown terms. A larger training corpus often leads to higher perplexity scores because the model has more material to learn from. Similarly, more complicated models, such as deep neural networks, can capture more complex patterns in data, resulting in higher perplexity ratings. Smoothing approaches, such as add-k smoothing or backoff smoothing, can help enhance perplexity by lowering the influence of unusual or unseen words in the test set.

- Without preprocess

```
python: can't open file '703613/ralphkedywillensbuteau/Downloads/HW2/ngram.py':  
[Errno 2] No such file or directory  
● (base) ralphkedywillensbuteau@Ralphs-MacBook-Pro HW2 % python ngram.py  
{'a': 0.5, 'saw': 0.25, '.': 0.25}  
Perplexity: 0.6042750794713536  
○ (base) ralphkedywillensbuteau@Ralphs-MacBook-Pro HW2 %
```

With remove step words:

```
● (base) ralphkedywillensbureau@Ralphs-MacBook-Pro HW2 % python main.py --model_type ngram --preprocess 0 --part 2
[nltk_data] Downloading package stopwords to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package omw-1.4 is already up-to-date!
Perplexity of ngram: 1.0134929981518455
F1 score: 0.7057, Precision: 0.7088, Recall: 0.7065
```

With remove punctuation:

```
● (base) ralphkedywillensbureau@Ralphs-MacBook-Pro HW2 % python main.py --model_type ngram --preprocess 1 --part 2
[nltk_data] Downloading package stopwords to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]   /Users/ralphkedywillensbureau/nltk_data...
[nltk_data] Package omw-1.4 is already up-to-date!
Perplexity of ngram: 1.1022500494398426
F1 score: 0.6769, Precision: 0.6926, Recall: 0.6815
○ (base) ralphkedywillensbureau@Ralphs-MacBook-Pro HW2 %
```

WE can observe that the perplexity of Ngram with preprocess is greater than that of Ngram without preprocess, which corresponds to the definition of perplexity.

For part 2:

Briefly explain the two pre-training steps in BERT.

BERT is a pre-trained natural language processing model developed by Google. Its pre-training procedure consists of two steps: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM involves masking a random set of words in the input text and training the model to predict them based on context. NSP educates the model to predict if two input sentences are sequential or not, assisting it in understanding the link between sentences and capturing long-term dependencies. BERT's training on these tasks enables it to represent the meaning of words and phrases in a comprehensive manner, making it suitable for a variety of NLP tasks such as sentiment analysis, question answering, and text classification.

Briefly explain four different BERT application scenarios.

BERT (Bidirectional Encoder Representations from Transformers) is a versatile natural language processing approach with many applications. The following are four BERT application scenarios:


1. BERT can be used to analyze the sentiment of text, such as social media posts, product reviews, or customer feedback. BERT can accurately assess the sentiment of the text by comprehensively handling the meaning of words and sentences.
2. Question-Answering: BERT can be used to answer questions based on text input, such as questions from a document or a website. BERT can deliver accurate answers to complex questions by comprehending the context of the question and the text.
3. BERT can be used to create chatbots that can interact with users in natural language. Chatbots can be used for customer service, personal assistants, and other applications requiring human-like conversation.
4. Text Classification: BERT can be used to categorize text, such as news articles, emails, or social media posts. BERT can effectively identify text based on its content by understanding the meaning of words and sentences in a comprehensive manner, making it helpful for a variety of applications.

Discuss the difference between BERT and distilBERT?

BERT is a larger and more accurate model, whereas DistilBERT is a smaller and more computationally efficient version that achieves equivalent performance on a variety of natural language processing tasks. The decision between BERT and DistilBERT is determined by the task's specific requirements and available processing resources.

Screenshot the required test F1-score.

```
100% 5000/5000 [30:37<00:00, 2.72it/s]
100% 10000/10000 [01:56<00:00, 86.20it/s]
Epoch: 0, F1 score: 0.9307, Precision: 0.9307, Recall: 0.9307, Loss: 0.2272
```



BONUS:

Explain the relation of the Transformer and BERT and the core of the Transformer:

The Transformer is a neural network architecture created by Google for natural language processing (NLP) tasks, allowing the model to balance the importance of distinct words in the input text based on their relevance to the present context. It is founded on the concept of self-attention, which enables the model to focus on different parts of the input text at each layer, capturing both local and global word dependencies.

BERT, on the other hand, is a Google-developed pre-trained NLP model based on the Transformer architecture. It extends the Transformer architecture by pre-training the model on a huge corpus of text data with two different training goals: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Masking some of the words in the input text and training the model to predict the masked words based on the context provided by the other words in the sentence is the goal of MLM. The NSP goal is training the model to predict if two input phrases are consecutive or not, hence assisting it in capturing long-term relationships in the input text.

BERT learns to represent the meaning of words and sentences more comprehensively by pre-training on these two objectives, making it a valuable tool for a wide range of NLP tasks. The self-attention mechanism, at the heart of the Transformer, is critical to BERT's ability to comprehend the meaning of language in its entirety, making it a critical component of the model.

For part 3:

Briefly explain the difference between vanilla RNN and LSTM.

Both vanilla RNNs and LSTMs are neural network architectures for sequence modeling, but they differ in their capacity to manage long-term dependencies. Vanilla RNNs have the vanishing gradient problem, but LSTMs use a memory cell and gating techniques to selectively remember or forget information from prior time steps. This allows LSTMs to capture long-term dependencies and produce more accurate predictions, making them better suitable for applications such as language modeling, machine translation, and speech recognition. While vanilla RNNs are simpler and more computationally efficient, LSTMs are more complicated and better suited for handling long-term dependencies.

Please explain the meaning of each dimension of the input and output for each layer in the model. For example, the first dimension of input for LSTM is batch size.

Each layer's input and output dimensions in a neural network are determined by the type of layer and the nature of the data being processed. The first dimension of the input tensor is typically the batch size, while the remaining dimensions are determined by the type of data being processed. The output dimensions are determined by the layer's architecture and characteristics, such as the number of neurons or filters. It is critical to maintain track of the input and output dimensions to ensure layer compatibility as well as proper model training and evaluation.

Screenshot the required test F1-score.

```
-----F1-----
100% 5000/5000 [00:40<00:00, 122.18it/s]
100% 10000/10000 [00:10<00:00, 953.17it/s]
Epoch: 0, F1 score: 0.4992, Precision: 0.5425, Recall: 0.5315, Loss: 0.6943
```

Discussion:

Discuss the innovation of the NLP field and your thoughts of why the technique is evolving from ngram -> LSTM -> BERT.

The evolution of NLP models from n-grams to LSTM to BERT may also be considered as a progression towards models that are more context-aware and can better capture all aspects of language. N-grams and other classic language modeling techniques are predicated on the notion that a word's probability depends only on its immediate environment, whereas LSTM and other recurrent models can capture longer-term dependencies. BERT goes this a step further by being bidirectional and capable of capturing a word's context depending on both its previous and future context. This has resulted in considerable increases in performance on a wide range of NLP tasks, from sentiment analysis to question answering.

Describe problems you meet and how you solve them.

For part 1, I found this Key error: "&";, Because I was not familiar with the dictionary, I found it difficult to utilize it appropriately. It took me a long time to test until I apply the 2d dictionary.

For part 3, This, in my opinion, is the most difficult challenge in this homework; until now, my RNN performance has not met the standards.