Exercise 1

Leong Eu Jinn

March 2023

1 Introduction

This report outlines the plan of work for developing a new program to estimate language models for improving speech recognition quality. The program will be developed in Python from scratch and will be aimed at addressing the issues faced by our customer who dictates books. The report also highlights the data-sets and resources required for developing a better language model.

- Problem Description: Making a language model to improve speech recognition quality
- Methods and metrics planned for evaluation: Set matrics such as error rate between language model and validation data-set.
- Outline of solution: The solution will be a language model that the customer is satisfied with

2 Data

- Examples/samples of dataset(s): Penn Treebank Corpus, Wikitext-2
- Key characteristics: Most commonly used for sequence labeling
- Necessary data preparation for further processing: pre-processing such as word segmentation, normalisation and lower-casing.

3 Methods

[2]Methods implemented for the language model are pre-processing, training, probability calculation and perplexity (intrinsic evaluation).

- Pre-processing: Lowercase of all tokens, deletion of empty lines adding SOS(start of sentence) and EOS (end of sentence) tokens, splitting the document into a token sequence.
- Training: Trains the model on all tokens received from the training set and a dictionary of bigrams throughout the text.
- Perplexity: Is calculated through the weighted average branching factor of a language.

4 Experiments

Upon creating the perplexity calculation method, I trained the with the ptb.train dataset that was given and then calculated the perplexity with the ptb.test dataset. A key-error was given as expected as the bigram ”no it” did not appear and so we have to further implement the language model with back-off to solve the problem of unknown tokens that are not in the bigram dictionary that was trained. If not the probability of an unknown bigram such as "no it" would be (count of "no it")/(count of "no") = 0/60(estimate) and then the perplexity calculation would be just be 0. [1]Two solutions to avoid the problem of unknown tokens is to:

- 1. Choose a vocabulary (word list) that is fixed in advance.
  2. Convert in the training set any word that is not in this set (any OOV word) to the unknown word token ¡UNK¿ in a text normalization step.
  3. Estimate the probabilities for ¡UNK¿ from its counts just like any other regular word in the training set.

- The second alternative, in situations where we don't have a prior vocabulary in advance, is to create such a vocabulary implicitly, replacing words in the training data by ¡UNK¿ based on their frequency. For example we can replace by ¡UNK¿ all words that occur fewer than n times in the training set, where n is some small number, or equivalently select a vocabulary size V in advance (say 50,000) and choose the top V words by frequency and replace the rest by ¡UNK¿. In either case we then proceed to train the language model as before, treating ¡UNK¿ like a regular word.

5 Evaluation and discussion

Upon further testing with the Wikitext2 train dataset with the 2-gram language model. A perplexity result was calculated for the trainset and similarly to the ptb did not function with the testing set as the bigrams were not similar for both sets. The problem of unknown bigrams and its probability can be resolved with many methods such as back-off and various methods of smoothing. The stretch-goal of the assignment should be made compulsary as so that the model will work on any dataset.

6 Conclusion

Further implementation of the stretch goals such as back-off. The model is clearly not suitable and ready to be pushed to the customer unless the further stretch goals of a back-off variant or a method of smoothing is completed.

References

[1] Daniel Jurafsky James H. Martin N-gram Language Models.2023

[2] https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9