CS410
Shuopeng Zhou
Fall 2020

Technology review on Language Model

Introduction

Language model including language detection model and language translation model are the most fundamental models for NLP area/fields, and also, it's considered as easiest model in NLP field. For industrial leading machine learning service providers, they use Language model as basic model or pre-processing data before it comes to more complex model such as Topic Modeling or Name Entity Recognition. Language model usually performances well even in simple linear model, but the data quality for training has become the key for a success model, also domain expertise on different languages is also essential for well-performed language model. Confusing language prediction is another bottleneck for providing a success language model, for example, Croatian and Boasian, Arabic and Egyptian Arabic, etc. The industrial solution is avoiding classifying them into sub-domain language including Azure Cognitive Service and AWS Machine Learning service, except for Google Cloud, which has huge data source with giant advantage on creating language model. As a discussion on language model, this paper will cover BPE tokenization, token storage, non-lexical/lexical comparison and "long-tail" problem and language model bottleneck

BPE tokenization

A simple approach of creating a language model will be using BPE tokenization, which is a common approach in NLP area. It's intuitive to define a set of languages the system will support, and for each language, the system will create a BPE dictionary storing N high frequency tokens inside. The token itself can be lexical or non-lexical, for example, we can have "Happy Birthday"

extract as “Happy” and “Birthday” or non-lexical extraction such as “ha-ppy-bir-thday”. There is no agreement which approach, or method is better, one advantage of non-lexical extraction would be saving a large space and less bias to well-known language. However non-lexical model tends to more instable as it can performs low on some simple case. Back to “Happy Birthday” example, you can see tokenizer can produce a token named “bir”, which is common word in Turkish, then in later weighting system, this sentence may be predicted as Turkish, however it’s the most common sentence in English.

Storage of BPE Dictionary

It’s an interesting topic once we utilize our model intro a production service. The naïve approach would be loading this dictionary into memory during service startup stage as a list or dictionary. System will read sentence for prediction and separate them into existing token from training data and calculate the weighted score based on BPE dictionary distribution. However, since there are existing large number of combinations for tokens, it will be time-consuming to go through BPE dictionary read the weight for each token. Thusly, the industry tends to use “Trie” tree or more complex tree structure to store the BPE dictionary for space and time efficiency

Long Tail and Bottlenecks

For every model we create in Machine Learning, it has its own sweet time and bitter time. For the early training stage, with more data it consumes, the better model has become. However, it can easily hit a bottleneck no matter how much training data provided or bigger BPE dictionary can

become. The reason behind is with more training data coming to our model, the dictionary/model only increase its long tail pattern not actually improving the model performance. Think about if the training data covers some hardly used word for English, named “Z”. It will become the long-tail word/token for BPE dictionary with low frequency such as 1 or 2. Later on even though the system sees the unknown data including this word, it still not able to classify it as English because this token has extreme low weight in our model. It won’t affect the final score in prediction phase. Thusly once the model hits its performance bottleneck, two ideal approaches will be data cleaning and multi-layer language model. Feeding more training data to the model won’t solve the problem

Conclusion

Language model is great starting topic/area for researching on NLP field. It’s simple to implement as linear model would be sufficient, however it’s also challenging to create one successful model such as Google Translator. For most industry language model provider, they tend to adopt engineering approach for creating great model such as skewing model to dominated languages (English/Spanish), feedback mechanism and multiple layer decision model for confusing languages.