Rafeeq Shodeinde

Kaggle Bakeoff Project

5/6/2019

CSCI 2930/5930

Irish Language Modelling Competition

Problem Definition:

For this project, the objective was to predict the probability of a word amongst a word pair being the correct word to occur in a sentence. The pair of words under consideration are seen in the list below:

['a','á','ais','áis','aisti','aistí','ait','áit','ar','ár','arsa','ársa','ban','bán','cead','céad','chas','chás','chuig','ch úig','dar','dár','do','dó','gaire','gáire','i','í','inar','inár','leacht','léacht','leas','léas','mo','mó','na','ná','os',' ós','re','ré','scor','scór','te','té','teann','téann','thoir','thóir']

To solve this problem, probabilistic models are used, implementing the N-gram algorithm. For this project, focus was placed on the implementation of the Unigram model and Bigram model. First the training data had to be cleaned before any language processing was performed. The training data was read into the script and split into a list of words. Then using regular expression, all unwanted characters which were not individual words in the sentences were removed from the file.

To measure the accuracy of our model, a log loss function is used to calculate the error which can be explained as the average of the negative log of the probabilities assigned to the correct words. The aim of this project was to reduce the error generated by the model.

Unigram Model:

To effectively create the unigram model, the algorithm depends on the count of the target words to predict the probability of a target word being the correct word in a sentence. In my algorithm the training data is parsed through splitting all the sentences into individual words. Following this, the training data is scanned for occurrences of the target words above and appended into a dictionary of words. The count of the word is used to predict the probability by using the following algorithm

$$P(a|s) = P(a) = \frac{C(a)}{C(a) + C(\text{á})}$$

This simply implies, to predict the probability of the word “a” being the correct word in the sentence, the count of the instances “a” which occur in training set is divided by the total number of times both pair of words occur in the training data. This was performed on each target word in the list of target words. This is a good algorithm because it relies only on the training data and better predictions are made the larger the corpus of words. Using the unigram method, the log loss value calculated was approximately 0.45 which is a good probabilistic value.

Bigram Model:

In a typical bigram model, the model depends on the count of the word occurring immediately before the target word to perform prediction. However, for this experiment and with the hopes of improving my model, the words immediately before and after the target word were considered for analysis. To proceed, the number of times in which the target word occurred after another word, was recorded in a dictionary. To obtain the probability of the count of a word occurring before the target word, the number of times a word occurs before the target is divided

by the total number of times the word occurs before both pair of targets. This is seen below as a formula:

$$P(a|s) = \frac{C(a|s)}{C(a|s) + C(\text{á}|\text{s})}$$

Then the probability of the word occurring after the target word is considered by counting the number of times the target word occurs before a word. This value is then divided by the total number of times that word occurs after either pair of target words:

$$P(s|a) = \frac{C(s|a)}{C(s|a) + C(s|\text{á})}$$

The same process is done to calculate the probability of the immediately before and after word for the other set of words in the pair, e.g. "á". All four probability values are used to compute alpha and beta values which are products of the before and after probability of the target word pairs

$$\alpha = P(a|s) * P(s|a); \quad \beta = P(\text{á}|\text{s}) * P(\text{s}|\text{á})$$

Both alpha and beta values are used to calculate a total probability to predict the word by:

$$total = \frac{\alpha}{\alpha + \beta}$$

The bigram model was relative good with predictions, generating a value of 0.6881.

Combination of Bigram and Unigram:

For further experimentation, the unigram and bigram model were both combined to test for improved predictive power by the model. The process of implementing the algorithm was done by making the model train on the bigram except for situations where no data was available in the dictionary. There are special cases in which the bigram model cannot be used because there is a zero value for the count of instances when a specific word occurs before the target word. The model will then back off one word in order to perform the unigram algorithm. The add one smoothing was also implemented to help improve the training. This model made relatively good predictions, having an error value of 0.44656 after log loss error was calculated.

Comparing the different predictive capabilities of the different models, it is seen that the combination of both the bigram and unigram performs better than when the algorithms are implemented individually.