Report part 1

Contents

Introduction

I Explore the use and potential improvements using n-gram language models for the purpose of sentence completion. I expand upon my work done in labs in which I used unigram and bigram models to generate solutions to the Microsoft sentence completion challenge (Zweig & Burges, 2011). The Microsoft sentence completion challenge requires a model to choose between 5 answers to fill the missing word in a sentence with 1 being correct. My interest in exploring optimisation possibilities with n-gram models is partly due to the low hardware requirements compared to other more complex solutions to the sentence completion challenge.

I compare both a random guess baseline, unigram, bigram and trigram model. Absolute discounting is then added to the bigram and trigram models in order to explore how the smoothing effects the accuracy of the models. N-gram models have been shown to have varying effects based on smoothing type used and domain used (Dumoulin, 2012).

This report ask, does increasing the complexity of n-gram models negatively or positively affects accuracy with this challenge and what effect so absolute discount smoothing have on the increase in complexity. My bigram and trigram solutions use an unknown token to deal with low frequency words in order to deal with data sparsity. The tests used to optimise the threshold of the unknown token gave further insight into the uses of the different models and where they perform best.

Method

The methods I use are a unigram, bigram and trigram model. The bigram and trigram models will include tests with and without the use of absolute discounting as a smoothing method. The trigram model will be tested with and without using probabilities from the bigram and unigram models. Each model contains an unknown word token used to replace low frequency words and combinations.

The unigram model is simple and used mainly as a baseline comparison for the success of the other models. During the training of this model, each word is counted and turned into a frequency dictionary. Any word seen fewer times than the “unknown threshold” is removed and added to the unknown token. The unknown token is used to represent out of vocabulary words as well as used in place of low frequency words. To generate a solution to the Microsoft sentence completion challenge, each answer token is run through the unigram in order to get a probability. The answer with the highest probability is chosen otherwise answer A is chosen if the answers are all the same probability or are all unseen. This use of default A is used with all models to control for the accuracy of random guessing. The probability of each answer is simply decided by P(w), calculated by the probability that word appears as frequency divided by total training words.

The bigram was created by taking word tokens, adding a start and end token to each side then sliding along with a window size of 2. For each pair of words in the window, the second word was added to a frequency dictionary which was specific to the first word. So each seen word had its own dictionary containing frequencies of every word seen after it. The bigram nested dictionaries then had the same process occur as with the unigram data. Where with every frequency dictionary, the tokens seen in total below the unknown threshold where replaced with the unknown token. This process was then repeated on the out layer where each key with

its value as a frequency dictionary was added to the unknown dictionary if the key was seen less than the threshold. The bigram dictionary then had every nested frequency reduced by 0.75. The total reduction was then added as a “discount token” in order to maintain the probabilities summing to 1. This nested dictionary is then converted to probabilities in the same way unigrams are except that each nested dictionary probability adds to 1.

The bigram model generates an answer by looking up the probability of each answer using the previous word as context. So, the probabilities for the I’th word will be found by using I-1 word in the question sentence. This effectively allows the probability of bigrams to be searched.

The trigram model follows the same approach as the bigram model except the sliding window size is increased to 3 meaning that the frequencies are stored in a dictionary containing nested dictionaries as values. The smoothing and unknown token are calculated the same with discount token being added to the bottom layers of the nested frequency dictionaries and unknown tokens being added to each layer. The probabilities are then looked up the same as with bigrams except the context window is increased from 1 to 2.

An extra version of the trigram model is used. The methodology is like the baseline 4gram model used in the sentence completion challenge paper (Zweig & Burges, 2011). I create the trigram along with bigram and unigram versions, the only difference being that when deciding on word choice probability, the probability methods from unigram bigram and trigram models are summed to provide a total probability for each potential answer. The hope is that this will allow the specificity of the trigram model while minimising the problem of data sparsity due to large amounts of trigrams being unseen. Between this and the absolute discounting, the hope is that the negative sides of using a trigram model over a bi or unigram can be minimised.

Parameter Settings

Due to my exclusive use of n-gram models the amount of parameter settings to control for an explore is low. The parameter setting for all models are “unknown threshold”, “discount amount” and “amount of training files”. The intention of this report is to investigate the effects of adding this smoothing as opposed to the optimisation of this as a parameter.

Unknown threshold was explored through multiple tests in order to determine the optimum number of times a word needs to be seen to be relevant to the model. I controlled this variable by performing both smoothed and unsmoothed versions of the bigram, trigram and the unigram solutions. Accuracy used to measure the optimum threshold. Below table 1, 2 and 3 show the tests performed while optimising the unknown threshold, the measurements are based on a scale of 0 to 1 representing percent accuracy of correct answers. Most results seemed to improve as training data size was increased with table 3 showing the biggest drop-off. This led to unknown token being fixed at applying to tokens with a frequency lower than 2 when in use.

Table 1: No use of unknown token

Absolute Discounting		No Smoothing		Baseline			
Bigram Score	Trigram Score	Bigram Score	Trigram Score	Unigram	Random Guessing	Training File Amount	Unknown word Threshold
0.26	0.26	0.24	0.21	0.26	0.2	10	0
0.20	0.23	0.20	0.21	0.23	0.2	20	0
0.21	0.23	0.21	0.22	0.24	0.2	30	0
0.22	0.23	0.22	0.22	0.22	0.2	40	0
0.22	0.24	0.23	0.22	0.23	0.2	50	0

0.23	0.23	0.23	0.22	0.23	0.2	60	0
0.24	0.22	0.24	0.22	0.24	0.2	70	0
0.24	0.24	0.24	0.22	0.23	0.2	80	0
0.25	0.25	0.25	0.22	0.23	0.2	90	0

Table 2: Unknown token applied to frequency below 2

Absolute Discounting		No Smoothing		Baseline			
Bigram Score	Trigram Score	Bigram Score	Trigram Score	Unigram	Random Guessing	Training File Amount	Unknown word Threshold
0.20	0.22	0.18	0.20	0.21	0.20	10	1
0.21	0.23	0.20	0.20	0.23	0.20	20	1
0.22	0.23	0.21	0.20	0.25	0.20	30	1
0.21	0.23	0.21	0.21	0.23	0.20	40	1
0.21	0.24	0.20	0.21	0.24	0.20	50	1
0.21	0.24	0.20	0.22	0.25	0.20	60	1
0.21	0.24	0.21	0.22	0.24	0.20	70	1
0.22	0.23	0.22	0.21	0.23	0.20	80	1
0.23	0.23	0.22	0.21	0.23	0.20	90	1

Table 3: Unknown token applied to frequencies below 3

Absolute Discounting		No Smoothing		Baseline			
Bigram Score	Trigram Score	Bigram Score	Trigram Score	Unigram	Random Guessing	Training File Amount	Unknown word Threshold
0.19	0.19	0.17	0.19	0.19	0.20	10	2
0.21	0.23	0.20	0.20	0.23	0.20	20	2
0.22	0.24	0.20	0.20	0.25	0.20	30	2
0.21	0.24	0.20	0.21	0.23	0.20	40	2
0.21	0.24	0.19	0.20	0.23	0.20	50	2
0.21	0.23	0.20	0.21	0.24	0.20	60	2
0.21	0.24	0.20	0.21	0.25	0.20	70	2
0.22	0.24	0.21	0.21	0.24	0.20	80	2
0.23	0.21	0.22	0.21	0.24	0.20	90	2

Discount amount is the amount discounted from each count of the bigram and trigram combinations. This amount was set as a flat 0.75 as shown to be optimal by church and gale (Church & Gale, 1991). Having this as a fixed amount greatly reduces the number of tests need to be performed.

Amount of training files used is varied throughout multiple tests. This setting was varied for the purposes of exploring how the accuracies achievable with the models with access to varied amounts of training data. This allows for further analysis on model accuracy under varying circumstances.

Evaluation

The evaluation of model performance will replicate my technique used in labs. Each model is scored between 0 and 1. 1 meaning each question was answered correctly. 0.25 meaning 25% where answered correctly. This score is based purely on the percentage of correct answer.

Tables 1, 2 and 3 show the initial testing based on optimising and investigating the use of the unknown token. These tables showed overall decreases in performance with small amounts of training data but with the differences evening out as training data sizes where increased. These tests also showed an initial baseline of the unigram model which while performing well with low amounts of training data, did not show the same increase as other methods showed. Table 4 shows these results.

Table 4: results

Smoothed Bigram Score	Smoothed Trigram Score	Bigram Score	Trigram Score	Unigram Score	Training File Amount
0.20	0.22	0.18	0.20	0.21	10
0.21	0.23	0.20	0.20	0.23	20
0.22	0.23	0.21	0.20	0.25	30
0.21	0.23	0.21	0.21	0.23	40
0.21	0.24	0.20	0.21	0.24	50
0.21	0.24	0.20	0.22	0.25	60
0.21	0.24	0.21	0.22	0.24	70
0.22	0.23	0.22	0.21	0.23	80
0.23	0.23	0.22	0.21	0.23	90
0.27	0.26	0.27	0.24	0.24	522

Initially, the smoothed versions of the models had a higher accuracy than their unsmoothed counterparts with the smoothed trigram model performing best bellow 90 training files used. The unsmoothed bigram later achieved the same accuracy of 27%. This might be due to the implementation of unknown tokens achieving similar results as the smoothing at lower frequencies. While the bigram models achieved the highest results, the difference is only 1% however the initial results with low training data is noticeably higher when smoothed than either bigram or unsmoothed trigram model. The strong initial increase in accuracy followed by a non trivial with large increases in data is a previously shown trait of the use of n-grams with the same trend being demonstrated by Lesher and Moulton. (Lesher & Moulton, n.d.).

The Microsoft challenge n-gram baseline managed to achieve 36% with a smoothed trigram and 39% with a smoothed quad-gram (Zweig & Burges, 2011). The difference in training data size being them using 126k unique words used over 5 times and my models using 152k unique words being used over 1 time. This difference in sparsity could likely increase the performance of my models with the same training data used. Similarly, to the continual but gradual increase shown in (Lesher & Moulton, n.d.).

My second trigram model with the combined probabilities is shown in table 5 compared to the initial trigram model. Table 5 shows that while there was a significant increase in performance, the 2nd approach appeared to have a significant increase on the non-smoothed version.

Table 5: 2nd combined probabilities model compared against first

Smoothed Trigram Score	Trigram Score	Smoothed trigram 2nd	trigram 2nd
0.263	0.243	0.264	0.263

All models managed to achieve significantly higher than the baseline for random guessing of 20%(0.2).

Conclusions

In conclusion the n-gram models appear to be highly dependent on training dataset size as shown in my investigations and through others (Lesher & Moulton, n.d.) (Zweig & Burges, 2011). While smoothing the data increased performance, the performance increase was less prominent when larger training sets where used.

Further Work

This report leaves room for further work. The potential for the n-gram models' effectiveness to increase with an increase in training data could be explored (Lesher & Moulton, n.d.). There is potential to attempt to use out of domain training data with topic modelling and domain adaption which is shown to improve success rates of n-grams for other problems (Bacchiani & Roark, 2003).

While my smoothing technique increased performance initially, it did not provide increases as the dataset scaled up, this leaves options for testing of other smoothing techniques to supplement the dataset, adding better scaling performance increases. Different smoothing techniques have been shown to have varying effects on n-gram models dependant on the domain used (Dumoulin, 2012).

The probability calculations can be altered to see if using a non-stationary n-gram model increases performance (Xiao & Liu, 2007).

Bibliography

Bacchiani, M. & Roark, B., 2003. *Unsupervised language model adaptation.* Hong Kong, IEEE.

Church, K. W. & Gale, W. A., 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language,* Volume 5, pp. 19-54.

Dumoulin, J., 2012. *Smoothing of ngram language models of human chats.* s.l., IEEE.

Lesher, G. & Moulton, B., n.d. *EFFECTS OF NGRAM ORDER AND TRAINING TEXT SIZE ON WORD PREDICTION,* New York: Department of Communication Disorders and Sciences.

Weeds, J., 2020. *Language Modelling 1,* Brighton: s.n.

Xiao, J. & Liu, B., 2007. An Empirical Study of Non-Stationary Ngram Model and its Smoothing Techniques. *Computational Linguistics and Chinese Language Processing,* 12(2), pp. 127-154.

Zweig, G. & Burges, C. J., 2011. *The Microsoft Research Sentence Completion Challenge,* s.l.: Microsoft Research Technical Report MSR-TR-2011-129.