N-grams Model Analysis: Unigram, Bigram, and Trigram

CSE 143 Natural Language Processing: Assignment 1

Alexandra Luther Abirami Patchaiyappan Mythili Karra

I. Introduction

In this assignment, we experiment with unigram, bigram, and trigram models of natural language processing to better understand n-gram concepts, smoothing and perplexity. We programmed the models that we later used to find different values for a variety of variables that affect the models. Throughout this report, we evaluate the differences in our data and examine any correlations. Overall, we aim for a better understanding of all the components that make up a n-grams model.

II. Model Description

Our program imitates the n-grams language model. Essentially, we take in a training data file and create the unigram, bigram and trigram models by using the corresponding equations (see Fig. 1). These models are represented as dictionaries with key value pairs of the token and the calculated probability for each token per n-gram model. In our model, we account for OOV words in the training data set by replacing any word that appears less than three times with the "UNK" token as stated in the assignment description. We then utilize the unigram, bigram, and trigram dictionaries that were created to create corresponding unigram, bigram, and trigram models for our development data and afterwards, for our test data. Using these baseline dictionaries, we analyzed perplexities and implemented smoothing through our experimentation described below. It should be noted that when calculating perplexities for our development and test sets, we needed to implement smoothing so that the bigram and trigram tokens with a probability of zero could still be parsed. For this instance of smoothing we used the random weight values: 0.3, 0.1, 0.6.

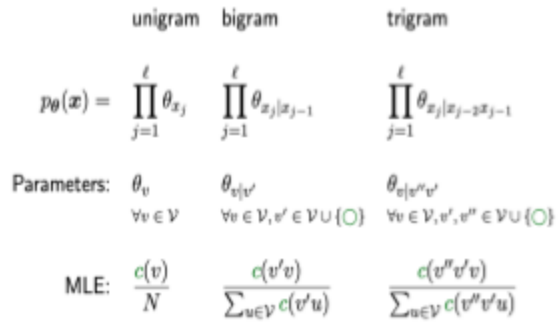


Fig. 1: equations, src: class slides presented by Jeffrey Flanigan (borrowed from Noah Smith)

III. Experimental Procedures

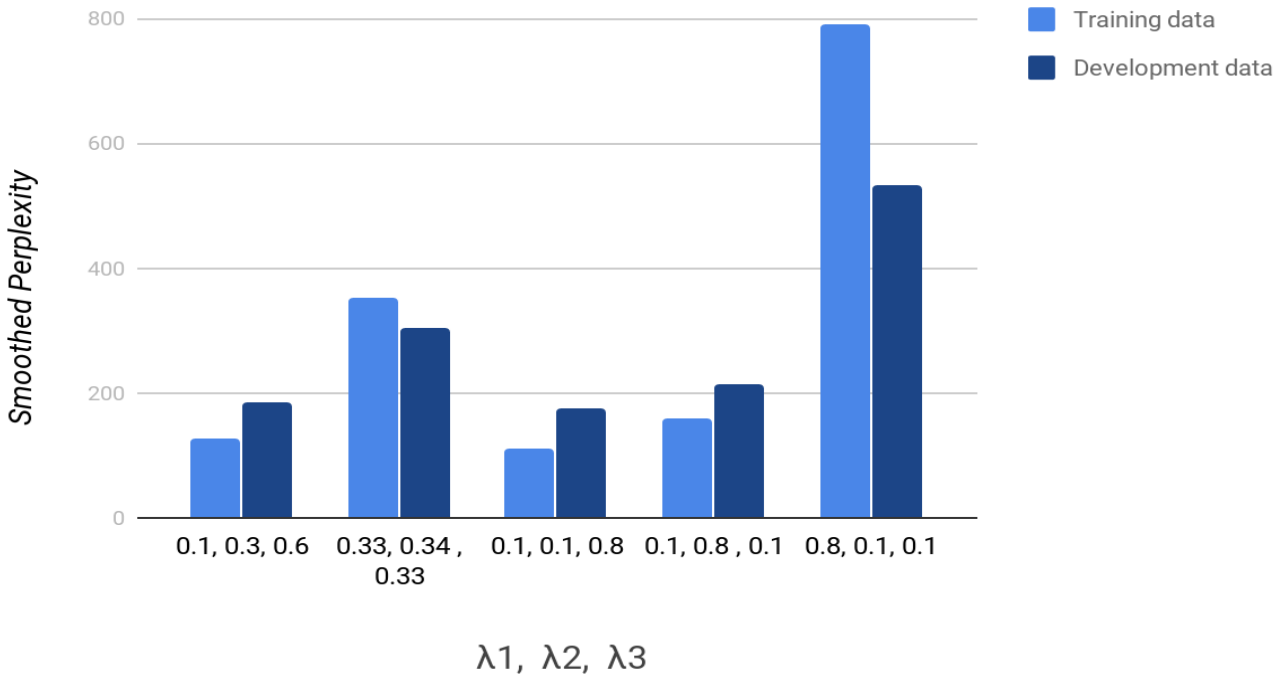
We examined four key groups of data which correspond with the four questions listed in the assignment description. The first portion examines the perplexities with different weight (lambda) values for the training set and the development set. The second part

examines examines the same hyperparameter values as the first part, but this time on the test data set. The third part examines the perplexities given only half the training data and looks for a relationship between the size of the training data and resulting perplexities in the development and test data sets. The last part examines the UNK tokens by changing the amount of UNKs in the training set and analyzing correlations with the perplexities.

i. Effect of Hyperparameter on the Perplexity of Training and Development Data

These perplexity scores demonstrate that with increased emphasis on the trigram, the smoothed perplexity decreases. This confirms that the trigram has a significantly lower perplexity score in comparison to the bigram and unigram. While this holds true for both the training and the development data the proportions are not as drastic for the development data as the test data. This is because although the test data accurately predicts the trend it does so much more drastically and since that algorithm is based of it this is more relevant the more words are tested together. When dealing with new data, in this context the development data, the unigram and bigram play a more significant role.

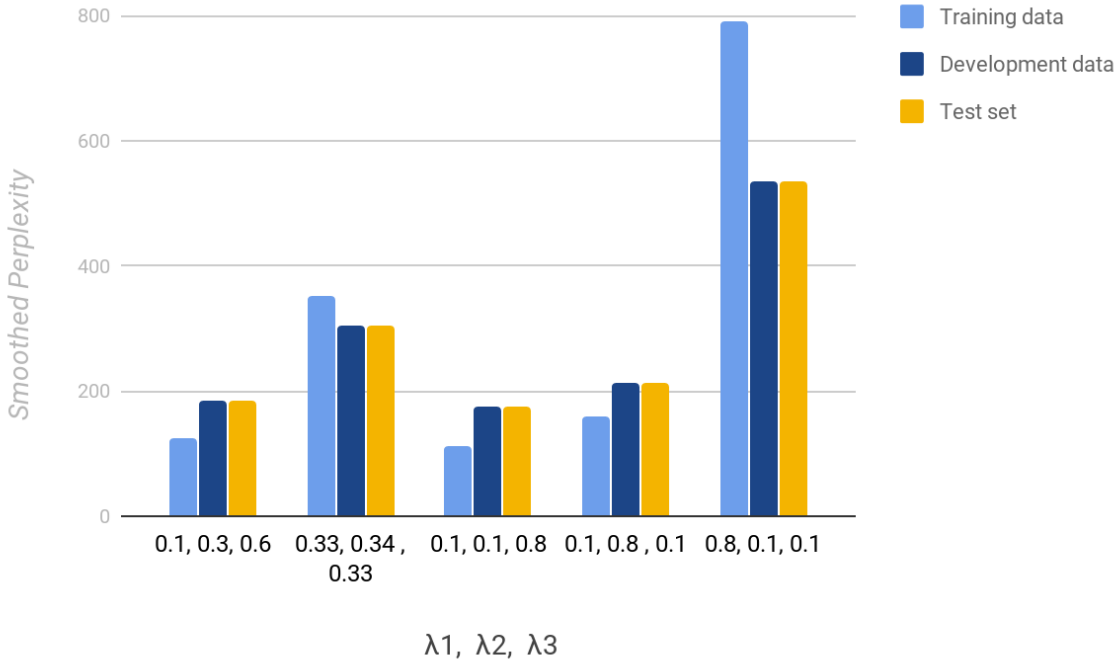
λ1	λ2	λ3	Perplexity for training	Perplexity for Dev
.1	.3	.6	125.4906	185.4612
.33	.34	.33	351.0572	305.7011
.1	.1	.8	111.6474	174.2993
.1	.8	.1	160.0988	213.3658
.8	.1	.1	789.7280	533.4527



ii. Effect of Hyperparameter on Perplexity of Test perplexity

The perplexities of the test set, while smoothing with the same hyperparameters as the development data, show very similar values, with differences in the decimal range. Since these are both different sets of new data and their perplexity scores are so similar, in comparison to the test data, it can be reasoned that they show the perplexity of most new data put into this algorithm.

λ1	λ2	λ3	Perplexity for training	Perplexity for Dev	Perplexity for Test
.1	.3	.6	125.4906	185.4612	185.3443
.33	.34	.33	351.0572	305.7011	305.6840
.1	.1	.8	111.6474	174.2993	174.2709
.1	.8	.1	160.0988	213.3658	213.0278
.8	.1	.1	789.7280	533.4527	533.7817



iii. Effect of Size of Training Data on Perplexity

If only half the training data is used, it would increase the perplexity on previously unseen data. The UNK token is a representation of an equivalence class of tokens, and this equivalence class expands as training data is removed. Removing data would rapidly increase the model probability of the UNK, causing the training data perplexity of the model distribution to shrink. Intuitively speaking, halving the training data would make more words appear less than the minimum number of times (causing them to be labeled as UNK), which would increase the probability of UNK and in turn decrease the overall perplexity. However, the perplexities for the development and training dataset are expected to increase as there is less training data for the development and training dataset to compare to. This is also proven through our data, where in the training data, the unigram perplexity went from 976.54 with the full data set to 816.48 with the half data

set, 77.07 to 63.08 for bigram perplexity, and 7.85 to 6.53 for trigram perplexity. In the development data, the unigram perplexity went from 630.48 to 582.21, the bigram perplexity went from 173.22 to 180.4 and the trigram perplexity went from 117.41 to 126.11 when going from using the full data set to using half of the data set. In the case of the test data, the unigram perplexity went from 630.96 to 582.38, the bigram perplexity went from 172.74 to 180.31 and the trigram perplexity went from 117.37 to 126.23. It is clearly shown in our results how the training data perplexities had decreased, but the development and test dataset had increased.

iv. Effect of UNK Tokens on Perplexity

If the tokens that appeared less than 5 times were converted to UNK and this were to be compared to when tokens that appear only once are converted to UNK, we predicted the perplexity on the previously unseen data would decrease. We predicted so because we believed that a model should get better as the number of UNK tokens decreased, as the models would be less surprised by the sample since it is more familiar with the data. However, in our data, we noticed that as the number of UNK tokens decreased, the perplexity increased. Tokens appearing less than 5 times are more likely to appear and be assigned to the UNK token than tokens appearing in the data set only once. As the data contradicted our initial prediction, we came up with a possible explanation. Perhaps if there are more UNK tokens, the perplexity would decrease since there is a higher probability for OOV words. In order to figure out which explanation would make the most sense, we would definitely need further experimentation on perhaps a different program in case the data from our program does not match that of others. This experiment has plenty of room for future

	UNK	1	< 3	< 5
Train	Unigram	1128.363248	816.4896196	803.4852488
	Bigram	77.1531828	63.08437004	75.63312296
	Trigram	7.357370392	6.535799371	8.580996675
Dev	Unigram	804.3560744	582.2131446	450.59957
	Bigram	217.524074	180.4042348	128.4887518
	Trigram	143.4328728	126.1160071	89.64888589
Test	Unigram	808.1140436	582.389905	445.7069448
	Bigram	217.7517856	180.3150081	126.8554027
	Trigram	143.8600023	126.2363182	88.67348183

IV. Conclusion

After creating, examining and smoothing unigram, bigram and trigram models which were used to calculate data perplexity of training, development and test data, we were able to perfect a model that demonstrated n-gram concepts while minimizing perplexity and then able to draw results and conclusions. We were able to develop and test our code, try different models and enhance them, and tune the hyperparameters.