CSE 143 Assignment 2

Sanjay Shrikanth, Matthew Daxner, Collin McColl, Soren Larsen

5/17/2022

1 N-gram Language Modeling Using Base MLE

Model Description: We implemented basic unigram, bigram, and trigram language models that use Maximum Likelihood estimation to quantify the probability of a word (or sequence of words) occurring in the sentence. In our probability calculations, we ignore all words that are not in the $n-$gram models after training.

Unigram: a `Python` dictionary that maps words to their respective count. We pooled all words with a count less than 3 as $<UNK>$.

Bigram: Similar to unigram, we have a dictionary that maps tuples (of size 2) to their respective counts. We keep an instance of unigram in our bigram model to ensure that all the words that are processed are actually part of the overall corpus

Trigram: Similar to the previous, we store 3−tuple mappings. We store an instance of Bigram (and therefore Unigram) to make sure no errors occur and avoid any non-vocabulary words being processed.

Experimental Procedure: We followed the general equations for the Maximum Likelihood calculations. Using the starter input and corresponding perplexities as a foundation, we coded each model in a way that it first passes the sample input test before evaluating the overall model perplexity. Our model initially "trains" using the training set to determine the counts, and thus the respective probabilities. We feed in the dev and test input after training to determine their respective perplexities.

Perplexities of the data sets across the three models

	Train	Dev	Test
Unigram	976.54	892.25	896.499
Bigram	55.11	19.33	19.43
Trigram	6.52	2.40	2.41

Results: We observe that the perplexities decrease as we go from training to test set. This is likely because the train set is the largest set of the three, thus having the most words for the model to account for. Furthermore, we notice that the perplexities decreased as n-gram increased. This is because with more information (prior words), the perplexity must decrease due to there being less uncertainty to account for.

2 Additive Smoothing

2.1 Smoothing with $\alpha = 1$

For this part, we implemented Laplace smoothing with $\alpha = 1$

Model Description: We used the same model structure established in the previous part and modified our probability computation to include the smoothing. For example, in bigram we used the form:

$$\frac{c(w_i, w_{i-1}) + \alpha}{c(w_{i-1}) + V \cdot \alpha} \tag{1}$$

where V is the size of the vocabulary. We implemented this similar addition for all other probabilities before taking the log to be used for perplexity calculations.

Experimental Procedure: First we tried to implement additive smoothing with $\alpha = 1$. After getting something that runs remotely well, we noticed that implementing this caused some functionality errors that needed to be fixed, which ended up being our consideration of *<UNK>* characters. After doing so, we modularized the smoothing addition so its easier to simply specify it on the command line and observe the outputs efficiently. We then repeated the process with multiple values of α.

Perplexities of the data sets across the three models

		Train	Dev	Test
$\alpha = 1$	Unigram	977.51	894.39	898.55
	Bigram	923.31	118.507	119.46
	Trigram	7372.53	13.99	14.18

2.2 Testing with different values of α

Perplexities of the data sets across the three models with different configurations of α

		Train	Dev	Test
$\alpha = 0.5$	Unigram	976.80	893.15	897.35
	Bigram	634.45	88.89	89.56
	Trigram	4725.55	11.83	11.98
$\alpha = 3$	Unigram	983.34	901.70	905.76
	Bigram	1640.474	190.31	192.00
	Trigram	12415.55	17.65	17.90

2.3 Best Hyperparameters and Results

Our best perplexity values with out any smoothing at all. So:

Perplexities of the data sets across the three models

	Train	Dev	Test
Unigram	976.54	892.25	896.499
Bigram	55.11	19.33	19.43
Trigram	6.52	2.40	2.41

Results: We observe that as expected, the perplexity of the dev and train set decrease substantially as n increases in the language model. Interestingly, our trigram model had extremely high perplexity in the training set but the lowest perplexity in the other data-sets by a considerable amount. This could be because due to there being more chance for low-probability sequences of 3 words, the model's perplexity would be high with the large training set. We also noticed that higher values of α did not improve the model's performance. This could be because alpha normalizes the probability distribution, so unseen or low probability data can be considered and thus lowering MLE values for higher probability sequences.

3 Interpolation

Model Description: For our model we implemented the given equation

$$\theta'_{x_j}|_{x_{j-2},x_{j-1}} = \lambda_1\theta_{x_j} + \lambda_2\theta_{x_j}|_{x_{j-1}} + \lambda_3\theta_{x_j}|_{x_{j-2},x_{j-1}} \tag{2}$$

Our model involves an instance of the unigram, bigram and trigram models. We adapted the probability calculation and Incorporated the weighted sum to find $P(w_i|w_{i-1}, w_{i-2})$ before computing the log likelihood.

Experimental Procedure: We started with dissecting the equation and understanding how each of the models contribute to the overall probability calculation. Once we matched the sample output, we tested different configurations of λ to see its effect on the model's perplexity.

3.1 Testing Different Values of λ

(*Question 1*)

Set Used	Train Set	Dev Set
0.3,0.3,0.4	11.68	2.81
0.6, 0.3, 0.1	29.817	3.51
0.1, 0.4, 0.5	9.51	2.63
0.1,0.3,0.6	12.81	2.76

(*Question 2*) We chose the parameters that reported the lowest perplexities on the train and development data-sets, those parameters were:

$$\lambda_1 = 0.1,\ \lambda_2 = 0.4,\ \lambda = 0.5$$

When the testing set was applied we found a perplexity of 2.64

3.2 Perplexity and Unseen Data

(*Question 3*) Removing half of the training data will *decrease* the perplexity on previously unseen data because with a smaller training set, its likely that the models will have a smaller vocabulary size due to the smaller entropy. So, when the model considers unseen data, there will be more words considered out-of-vocabulary in the MLE calculations. This pools together all unseen words into a singular class, thus decreasing perplexity as the model has more certainty to predict those words.

Perplexities after cutting the dataset in half

	Train	Dev	Test
Unigram	769.455	723.12	725.55
Bigram	13.31	13.27	13.21
Trigram	3.19	2.00	1.99

Removing half the training set had a noticable decrease on perplexity on all language models.

(*Question 4*) Converting all tokens that appeared less than 5 times to $<UNK>$ would decrease the perplexity on previously unseen data when compared to only converting tokens that appeared less than 3 times. This happens because when we add more words to the $<UNK>$ label we decreasing the vocabulary size of the training set. When the model goes to evaluate unseen data there will be more words that it skips over because of increase of words encapsulated in the $<UNK>$ label, though the model will now have a greater understanding for all other words outside of the $<UNK>$ label as we have increased how often they must occur in the training data.