CSE 143 Assignment 2

Liam Xu, Jaisuraj Kaleeswaran, Vishwa Vijayasankar

July 28, 2024

1 Programming: n-gram language modeling

Training Set		Dev Set		Test Set	
Model	Perplexity	Model	Perplexity	Model	Perplexity
Unigram	976.54	Unigram	892.25	Unigram	896.49
Bigram	77.07	Bigram	28.29	Bigram	28.31
Trigram	7.87	Trigram	2.99	Trigram	2.99

The perplexities of the unigram model on the different data sets are the highest, while the bigram perplexities are much lower, and the trigram perplexities are the lowest. Also, the perplexities of the models on the debug file with HDTV . were $658, 63.7, 39.5$, respectively. This is what was expected from the assignment document against the sample tests. Furthermore, the behavior of the perplexities of the models were also expected to be what they were. The unigram models consider each word independently of its context, leading to less accurate predictions. The bigram model, which considers the previous word, significantly reduces perplexity. The trigram model further improves performance by considering the previous two words, capturing more context and yielding the lowest perplexity.

2 Programming: additive smoothing

$\alpha = 1$

Training Set		Dev Set	
Model	Perplexity	Model	Perplexity
Unigram	977.51	Unigram	894.39
Bigram	1442.31	Bigram	1669.66
Trigram	6244.42	Trigram	9676.65

$\alpha = 0.5$
Training Set

Model	Perplexity
Unigram	976.80
Bigram	971.66
Trigram	3964.85

Dev Set

Model	Perplexity
Unigram	893.15
Bigram	1241.95
Trigram	7905.41

$\alpha = 0.1$
Training Set

Model	Perplexity
Unigram	976.55
Bigram	407.84
Trigram	1115.69

Dev Set

Model	Perplexity
Unigram	892.39
Bigram	701.73
Trigram	4899.49

Using additive smoothing resulted in worse perplexities than before, so the best hyperparameter would be $\alpha = 0$, which is the same as not using any smoothing. This suggests that additive smoothing may allocate too much probability mass to unseen events, reducing the model's accuracy. Specifically, by adding a constant α to all counts, we effectively distribute some of the probability mass away from frequently occurring events to those that were never observed in the training data. This can lead to a situation where the model becomes less certain about frequent events, which negatively impacts its overall performance. Essentially, because additive smoothing shifts too much of the probability mass away from commonly seen tokens, it results in a less accurate model overall.

3 Programming: smoothing with linear interpolation

3.1

Training Set

Hyperparameters	Perplexity
$\lambda_1 = 0.1, \lambda_2 = 0.3, \lambda_3 = 0.6$	11.15
$\lambda_1 = 0.2, \lambda_2 = 0.2, \lambda_3 = 0.6$	11.53
$\lambda_1 = 0.1, \lambda_2 = 0.4, \lambda_3 = 0.5$	12.44
$\lambda_1 = 0.05, \lambda_2 = 0.4, \lambda_3 = 0.55$	11.57
$\lambda_1 = 0.01, \lambda_2 = 0.4, \lambda_3 = 0.59$	10.95
$\lambda_1 = 0.6, \lambda_2 = 0.3, \lambda_3 = 0.1$	38.33

Dev Set

Hyperparameters	Perplexity
$\lambda_1 = 0.1, \lambda_2 = 0.3, \lambda_3 = 0.6$	3.25
$\lambda_1 = 0.2, \lambda_2 = 0.2, \lambda_3 = 0.6$	3.33
$\lambda_1 = 0.1, \lambda_2 = 0.4, \lambda_3 = 0.5$	3.32
$\lambda_1 = 0.05, \lambda_2 = 0.4, \lambda_3 = 0.55$	3.24
$\lambda_1 = 0.01, \lambda_2 = 0.4, \lambda_3 = 0.59$	3.19
$\lambda_1 = 0.6, \lambda_2 = 0.3, \lambda_3 = 0.1$	4.57

3.2

Test Set

Hyperparameters	Perplexity
$\lambda_1 = 0.01, \lambda_2 = 0.4, \lambda_3 = 0.59$	3.19

3.3

If you used half the training data, the perplexity on unseen data would increase, because the model would have less data to learn from and become less capable of predicting the next word correctly. When the training data is reduced by half, the model encounters fewer examples of word sequences, leading to less reliable probability estimates for those sequences. This is because they lack the extensive context provided by a larger dataset, which is essential for making accurate predictions. With less data, the model's exposure to rare words and phrases decreases, increasing the likelihood of errors when encountering such words in unseen data. Consequently, the model becomes less confident in its predictions, resulting in higher perplexity.

3.4

If you converted all tokens that appeared less than 5 times to ' < UNK > ', the perplexity on unseen data would decrease compared to an approach that converts only tokens that appeared just once to ' < UNK > '. This is because reducing the number of unique tokens in the vocabulary helps the model to generalize better by focusing on more common and representative words. By converting infrequent tokens to ' < UNK > ', we achieve a smaller vocabulary, allowing the model to allocate more probability mass to common words, which are more likely to appear in the unseen data. This concentration of probability mass leads to more accurate predictions. Additionally, treating rare words as ' < UNK > ' improves generalization by preventing the model from overfitting to rare events that might not appear in the validation or test sets. A smaller and more focused vocabulary also reduces the computational complexity of the model, leading to faster and potentially more stable training and evaluation processes. Empirical evidence suggests that handling rare words by converting them to ' ⟨UNK > ' enhances the model's ability to handle unseen data, preventing it from assigning significant probability to infrequent and potentially noisy words. Thus, converting tokens that appear less than 5 times to ' < UNK > ' improves the model's robustness and decreases the perplexity on previously unseen data.