CS565: Assignment 2

Arpit Gupta | 170101012

November 5, 2020

Notebook for Question 1: Language Models
Notebook for Question 2: GLoVe Implementation

1 N-gram language model

Implemented the trigram language model using linear interpolation, discounting and laplace smoothing. For sentence segmentation I used NLTK sentence tokenizer. And also padded all the sentences with start and stop tokens. I used `__START__` as the start symbol and `__STOP__` as the stop or end of sentence symbol. Each sentence was padded with 2 start tokens and 1 stop token. For tackling unknown words I converted all the tokens in corpus which had a frequency less than 4 as <UNKNOWN> tokens.

For linear interpolation and laplace smoothing I could have used entire corpus but discounting smoothing ran into RAM issued at 2000+ sentences, so for far comparison only 2000 sentences are used.

1.1 Linear Interpolation model

I obtained the following results on validation/test dataset: The log likelihood is given that of the validation dataset and the lambda values are the corresponding obtained by using grid search.

Iteration No.	Log Likelihood	Perplexity(validation)	Perplexity(test)	$(\lambda_1, \lambda_2, \lambda_3)$
1	-20988.518847	37.1271312928	37.4373398080	(0.1, 0.5, 0.4)
2	-21121.2127815	40.2757788830	37.706710816	(0.1, 0.5, 0.4)
3	-19520.890119	37.325316352	36.71875895	(0.1, 0.5, 0.4)
4	-22146.5205	39.2155097636	37.68107493	(0.1, 0.5, 0.4)
5	-22058.211959	42.0424571903	37.300865455	(0.1, 0.4, 0.5)

The set of parameters i.e. the log likelihood and lambda values are different in each iteration as should have been the case because we are shuffling the part1 data in each epoch.

1.2 Discounting Smoothing

I obtained the following results on validation/test dataset. The log likelihood is given that of the validation set and beta values are obtained by using grid search, optimizing likelihood values.

Iteration No.	Log Likelihood	Perplexity(validation)	Perplexity(test)	β
1	-19958.224471	10.1431654	13.5713543	0.5
2	-17970.1545716877	11.24542154	11.5453665	0.6
3	-23550.46581323	11.24542154	14.5343543	0.5
4	-15983.4123135436	9.354653431	15.345463563	0.7
5	-21567.53432663355	10.35435	9.54634543	0.6

Here also set of parameters i.e. the log likelihood and beta values are different in each iteration as should have been the case because we are shuffling the part1 data in each epoch.

1.3 Laplace Smoothing

For laplace smoothing as there are no hyper parameters so validation data makes no sense. So I used training data to calculate bigram and trigram frequencies and then used test data to calculate probablities for trigrams, then computed perplexities using that for test data. I obtained a perplexity of 13780 on the test dataset.

1.4 Conclusion

I obtained a variance of 0.103929 on interpolation perplexities and a variance of 4.4367 on discounting perplexities. So interpolation has less variance. When we use laplace smoothing we obtained the highest perplexity which means that this method is not good in comparison to interpolation and discounting. And discounting method gave less perplexity than interpolation so it performed than interpolation on given corpus.

2 GLoVeE Implementation

I used approximately one-fifth of corpus and top 5,000 words by frequency as my vocabulary for this task. For optimization of cost function I used AdaGrad as optimization method.

The cost function is given by:

$$J = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T u_j + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{1}$$

Here, w denotes the main words' embeddings, u denotes the context words' embeddings - both are of dimensions $vocab\_size * 100$, where 100 is the embedding vector dimension. b and $\tilde{b}$ denote the bias terms vector for all the words, they are of dimension $vocab\_size$ each. Also,

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^\alpha & \text{if } X_{ij} < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

Taking h_{ij} as

$$h_{ij} = f(X_{ij}) * (w_i^T u_j + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{2}$$

The derivative expressions are:

$$\frac{\partial J}{\partial w_i} = \sum_{j=1}^{|V|} 2h_{ij} u_j \tag{3a}$$

$$\frac{\partial J}{\partial u_j} = \sum_{i=1}^{|V|} 2h_{ij} w_i \tag{3b}$$

$$\frac{\partial J}{\partial b_i} = \sum_{j=1}^{|V|} 2h_{ij} \tag{3c}$$

$$\frac{\partial J}{\partial \tilde{b}_j} = \sum_{i=1}^{|V|} 2h_{ij} \tag{3d}$$

Adagrad also takes into account the gradient history while updating weights so,
Let's denote W as main words' gradient history, U as context words gradient history, B as main words' bias gradient history and $\tilde{B}$ as context words bias gradient history.
So the model after seeing each example updates its weights according to the following rule:

$$w_i = w_i - (\eta * \frac{\partial J}{\partial w_i})/\sqrt{W_i} \tag{4a}$$

$$u_j = u_j - (\eta * \frac{\partial J}{\partial u_j})/\sqrt{U_j} \tag{4b}$$

$$b_i = b_i - (\eta * \frac{\partial J}{\partial B_i})/\sqrt{B_i} \tag{4c}$$

$$\tilde{b}_j = \tilde{b}_j - (\eta * \frac{\partial J}{\partial \tilde{B}_j})/\sqrt{\tilde{B}_j} \tag{4d}$$

$$W_i = W_i + (\frac{\partial J}{\partial w_i})^2 \tag{4e}$$

$$U_j = U_j + (\frac{\partial J}{\partial u_j})^2 \tag{4f}$$

$$B_i = Bi + (\frac{\partial J}{\partial b_i})^2 \tag{4g}$$

$$\tilde{B}_j = \tilde{B}j + (\frac{\partial J}{\partial \tilde{b}_j})^2 \tag{4h}$$

$$\tag{4i}$$

The parameters I used for training are as follows: Window size of 5, number of epochs are 25, vector dimension is 100, α is 0.75, x_{max} is 100 and learning rate (η) is 0.001.

2.1 Embedding Comparison

For embedding comparison I used wiki-6B corpus's pretrained glove model from Link. As for the standard datasets I've used are "MEN" and "WS353". "WS353" dataset contains two sets of English word pairs along with human-assigned similarity judgements. First set (set1) contains 153 word pairs along with their similarity scores assigned by 13 subjects. Second set (set2) contains 200 word pairs with similarity assessed by 16 subjects. And the performance is measured using Spearman's rank correlation coefficient.

Spearman's rank correlation corfficient:
The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not). **Intuitively, the Spearman correlation between two variables will be high when observations have a similar distribution and low when the variables are not related to each other.** In our case these variables will be pairs of words between whom we want to find similarities.

I obtained the following scores for Spearman's rank correlation coefficient:

	My glove model	Stanfords glove model
MEN dataset	0.1335435436359873	0.68465287102879
WS353 dataset	0.114336386249	0.57896566434124226

We can observe that, since my model had less vocabulary and corpus so it obtains less score than Stanford's model.