CS565: Assignment-2 Report

Umang(170101074)

Link to colab notebook for language models: Assignment2(CS565)_first.ipynb
Link to colab notebook for GloVe: https://colab.research.google.com/drive/1Ian3mI3mHJIB30EWjYdJR49hVdQiyJSf?usp=sharing

1. N-gram language models

For all the trigram models, only a small subset of the given corpus(about 1400 sentences) is used as for larger corpus, the RAM for google-colab notebook crashed.
For mapping of unknown words, a frequency threshold of 6 is used.

1. Interpolation Smoothing
   For interpolation smoothing, perplexity and log-likelihood on the validation set are obtained as follows:

Iteration Number	Perplexity	Log-likelihood
1	1.004	-0.0058
2	1.021	-0.030
3	1.020	-0.0289
4	1.016	-0.023
5	1.010	-0.0157

   Parameters lambda1, lambda2 and lambda3 are 0.1, 0.5 and 0.4 respectively. They are the same in each iteration.
   For the test set, perplexity is 1.004 and log-likelihood is -0.035.

2. Discounting Smoothing
   For discounting smoothing, perplexity and log-likelihood on the validation set are obtained as follows:

Iteration Number	Perplexity	Log-likelihood

1	15.214	-3.927
2	14.542	-3.862
3	14.683	-3.876
4	15.122	-3.918
5	14.284	-3.836

Parameter beta for bigram probability distribution and trigram probability distribution are both 0.9.
They are the same in each iteration.
For the test set, perplexity is around 13.5 and log-likelihood is around -3.7.

3. Laplace Smoothing

If only Laplace smoothing is used, perplexity and log-likelihood on the validation set are obtained as follows:

Iteration Number	Perplexity	Log-likelihood
1	39.429	-5.301
2	45.256	-5.500
3	38.975	-5.284
4	40.326	-5.333
5	38.048	-5.249

For the test set, perplexity is around 38 and log-likelihood is around -5.3.

For the selected subset from the given corpora, both interpolation and discounting smoothings performs better than Laplace smoothing. The best out of three is interpolation as it has the least perplexity(close to 1) and simultaneously highest log-likelihood. Interpolation smoothing had the least variation.

1.3.2 Vector Semantics: GloVE implementation

GloVe is implemented using AdaGrad as an optimization method. The model is trained on one-fifth of the given corpora.
The cost reduction with each iteration can be seen here: Link to cell
Comparison of embeddings is done using pre-trained embeddings on WS353 and MEN datasets. The *web* mounted using google drive can be found on Drive Link.

The Spearman correlation of scores for WS353 are as follows:
For implemented model, correlation is 0.022
For pre-trained model, correlation is 0.531
Similarly, the Spearman correlation of scores for MEN are as follows:
For implemented model, correlation is 0.029
For pre-trained model, correlation is 0.696

Formulation of adagrad derivative expression is as follows:

$$x(t+1,i) = x(t,i) - (\eta/\sqrt{\sum_{\tau=1}^{t-1} g(\tau,i)^2})g(t,i)$$

where η is the learning rate, x(t+1,i) is value to feature at step t+1, $\sqrt{\sum_{\tau=1}^{t-1} g(\tau,i)^2}$ is sum of squares of past gradients and g(t,i) is the partial derivative of the cost function w.r.t. to the parameter x(i) at step t.
This expression for optimization algorithm is used as follows:

word_vector.dot(context_vector) + word_bias[0] + context_bias[0] - np.log(x_ij)
cost =f(x_ij)*[dot_product(word_vector, context_vector) + word_bias + context_bias -log(x_ij)
]^2

g(word) = [dot_product(word_vector, context_vector) + word_bias + context_bias -log(x_ij)]*
context_vector
g(context) = [dot_product(word_vector, context_vector) + word_bias + context_bias -log(x_ij)] *
word_vector
g(word_bias) = [dot_product(word_vector, context_vector) + word_bias + context_bias -log(x_ij)]
g(context_bias) = [dot_product(word_vector, context_vector) + word_bias + context_bias -log(x_ij)]

word_vector = word_vector - (learning_rate*g(word))/sqrt(G(word_vector))
context_vector = context_vector - (learning_rate*g(word))/sqrt(G(context_vector))
word_bias = word_bias - (learning_rate*g(word_bias)/sqrt(G(word_bias))
context_bias = context_bias - (learning_rate*g(context_bias))/sqrt(G(context_bias))