CS5740: Assignment 2
Github group: Technologic

Chaoping Deng cd446
Yating Zhan yz2237
Ying Zhu yz2354

1 Introduction

In this assignment, our task was to build vector embedding for words. Our training data came from the billion word language model benchmark data-set. We used the first 1 million sentences in the dataset as our small training data and use the whole data set as our large training data. We experimented with different preprocessing strategies, methods to handle unknown words, and skip-gram model with different parameters.

2 Approach

2.1 Preprocessing and cleaning

2.1.1 Clean text

- Syntax Clean
  - remove digits and special characters
  - replace newlines with spaces
  - remove zeros
  - split on 1+ spaces
  - remove single character
  - convert word to lower cases
- Contextual Clean
  - Lemmatize and stem each word

2.1.2 Synonym Processing

Synonym processing is used for unknown words handling. For each unseen word (after loading training data), we use wordnet to find the corresponding dictionary word with the closest similarity to it and treat it as that word. A threshold (=0.8) is set to categorize an unseen word to unknown word.

2.2 Word-context pairs generating

2.2.1 Linear Skip-Gram

To construct linear skip-gram pairs, we create the vocabulary on top of the cleaned data and we use the same vocabulary for both word and context. For each word, we generate (word, context) pairs based on the window size of the context. For example, given a cleaned sentence "support, fake, war, terror, understand, following, orders," and window size 2, we generate ("war", "support"), ("war", "fake"), ("war", "terror"),and ("war", "understand") for the word "war."

2.2.2 Structured Skip-Gram

We experimented with structured skip gram by using the training-data.1m.conll file.The file contains information relating to syntactic dependencies. For a target word, we consider enriching the context pair with modifier information. Namely, for each context-word pair, we add modifier relating relationship/inverse relationship between the word with its head and feed them into our skip gram model, in addition, relationship with a preposition are collapsed which leads to the direct connection between context and its preposition. The snippet of a preprocessed context file is as below: ('Control','Disease/nn_1'), ('Prevention', 'Control/conj'),('Control','Prevention/conj_1'), ('Control', 'Centers/prep_for'),('Centers','Control/prep_for_1').

2.3 Model construction

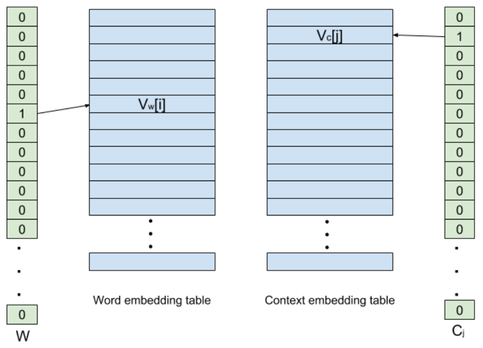


Figure 1: Neural network architecture

We used skip gram model with negative sampling to construct word embedding. To construct our neural network structure, we used two look-up parameter tables with the dimension of vocabulary size ∗ embedding dimension. We used one of the look-up parameter tables to get word embedding and used the other look-up parameter table for context embedding. The loss function to update our embedding table is:

$$-(\log\sigma(V[c]\cdot V[w]) + \sum_{(w,c)\in D'} \log\sigma(-V[c]\cdot V[w]))$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and V[c] and V[w] from the two look-up parameter tables.

2.4 Model Optimization

2.4.1 Negative Sampling

To generate negative samples, we constructed a large array and fill the array with the index of each words multiple times. The number of times each word appears in the array is calculated by the count of the word in training data multiply $P(w)$, where $P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n}(f(w_j)^{\frac{3}{4}})}$. The power $\frac{3}{4}$ is chosen according to reference "Word2Vec Tutorial." During training, for each (word, context) pair, we randomly generate 15 numbers within the range of the length of the array and use the 15 numbers as indices to get the negative sampling words.

2.4.2 Sub-sampling

To down sample the high frequency words in the vocabulary, we calculate the probability of keeping the word in a table with the formula: $P(w_i) = (\sqrt{\frac{z(w_i)}{0.001}} + 1) \cdot (\frac{0.001}{z(w_i)})$ where $z(w_i) = \frac{count(w_i)}{\sum_{j=0}^{n} count(w_j)}$. The sample value 0.001 is chosen according to reference "Word2Vec Tutorial." Each time when we construct a skip-gram (word, context) pair for word i, we randomly generate a float number between 0 and 1, and if the random number is small than $P(w_i)$, we keep the word, otherwise, we skip that word.

2.4.3 Handling of Unknown words

We define a word that is not included in the training data as Unknown. There are various ways of handling unknown words. The easiest way is to throw out any unknown words from dev and test vocabulary. This would inevitably result in a significant loss of information especially the case where there are a large number of such words. We explored the following two ways of Unknown words handling.

- UNK key
  Under this setting, we count the total appearances of each word and use a threshold to label words with a count less than the threshold as Unknown. The objective here is to capture the average value of unknown that would produce the best testing accuracy. The benefit of this approach is that we could reduce the training samples efficiently. The downside is that every single unknown word are handled the same way. When the threshold is large enough, many information will be lost and each unknown words will be treated the same.

- WordNet Processing
  To further differentiate each Unknown words, we used wordnet to find synonyms from the dictionary for each unseen words. If a synonym is found for a certain unseen word in dev/test set, the unseen word is replaced by that synonym from the training set. We believe we can use words existing in the dictionary to approximate a word we have not encountered before. Our further analysis justifies this finding.The following is an example of synonym pairs as processed by the wordnet file corresponding to (target word, dictionary word, similarity) we then load the preprocessed file based on a threshold of choice (0.8 for our experiments)
  [antecedent, grandparents] 0.9
  [squishing, squished] 1.0
  [undatable, occurred] 0.5
  [provincialism, parochialism] 0.875
  [narrow-mindedness, parochialism] 0.9333

3 Experimental Setup

3.1 Amounts of training data

We experimented with different amounts of training data. For large data, we used the billion word language model benchmark data set. And We used the first 1 million sentences in the data-set as our small training data. We cleaned the raw text on both data-sets as described in section 2.1.1 and used linear skip-gram to construct the (word, context) pairs with threshold 25 for low-frequency words. For large data-set, we have a vocabulary size of 123122 and 267600000 pairs, and for the small data set, we have a vocabulary size of 18625 and 12600000 pairs. We run 1 epoch of the model described in section 2.3 with window size 5 and 500 embedding dimension. We did negative sampling size 15 and sub-sampling on both data-sets, and we used UNK token to handle both unknown words.

3.2 Dimensions for the embeddings

We experimented with different dimensions for the embeddings (dim = 300, 400, 500, 600). For all the trials, we used small data-set (first 1 million sentences in the billion word language model benchmark dataset) as the training data. We cleaned the raw text on both data-sets as described in section 2.1.1 and used linear skip-gram to construct the (word, context) pairs with threshold 10 for low-frequency words. We run 1 epoch of the model described in section 2.3 with window size 2, negative sampling size 15 and sub-sampling, and used UNK token to handle both unknown words on all the trials.

3.3 Window size

We experimented with window size 3 and 5 for (word, context) pair construction on small data-set. Both trials used synonym processing described in section 2.1.2 and used linear skip-gram to construct the pairs with the threshold for low-frequency words. We run 1 epoch on both trials with the model described in section 2.3 with negative sampling size 15 and sub-sampling.

3.4 Contexts (linear, syntactic)

With parameters(window size 500) consistent with the linear model. We experimented with syntactic skip gram using the training-data.1m.conll file which contains structured information regarding the 1 million sentences. We first processed the conll data file into structured pairs and then feed the data into the same skip gram model as used in the linear case. We generated 238904 unique words and repeated the training for 3 epochs. We also used a threshold of 0 to handle unknown words.

3.5 Stop Criteria

We stop the training by identifying saturation for test correlation score. Specifically, for every 10 million context-word pair window, if the training correlation score fluctuates no more than +=0.01 from its running mean, then we stop the training process.

4 Results and Analysis

4.1 Experiment Analysis

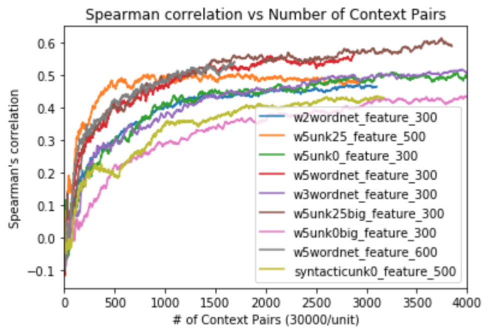


Figure 2: Result Analysis

In Figure 2, we plot correlation score progressing graphs for different models that we tried. We will break it down with detailed analysis in this section.

4.1.1 Amounts of training data results

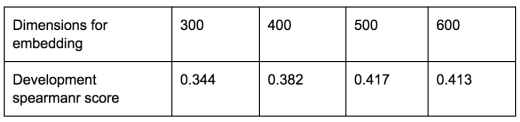
Dimensions for embeddings	300	400	500	600
Development spearmanr score	0.344	0.382	0.417	0.413

Figure 3: Development spearmanr result

As we can see from Table 3, the big dataset produces better correlation score. By utilizing the big data set, there are two advantages: First, we can encounter more context regarding a certain target word and produce higher correlation score for them. Second, there are less unseen words. Given more relevant data in the corpus, we can produce better result even with a less superior model.

4.1.2 Dimensions for Embeddings results

As feature size increases, the learning rate significantly increases but it has a tendency of overfitting training data. By increasing feature size, we are essentially increasing model complexity and therefore more likely to produce models that overfit.



Dimensions for embedding	300	400	500	600
Development spearmanr score	0.344	0.382	0.417	0.413

Figure 4: Development spearmanr scores with different embedding dimension size

4.1.3 Window size results

As window size increases, the correlation score increases. As we produce more context around a certain target word, we build more associations for that word to be learned. The downside of this, however, is that if the window size increases to a certain number we must include an unassociated word from a different sentence. This will have a negative impact on model performance.

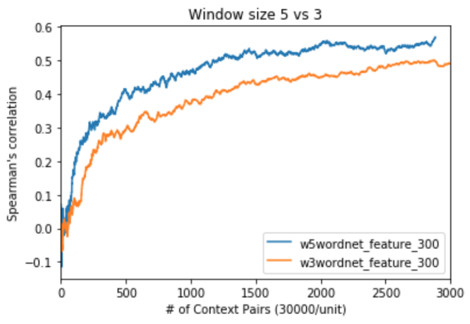


Figure 5: Widow size change results

4.1.4 Context results

It is interesting to see that the syntactic model performs relatively poorly. Our hypothesis is that it could be due to the sparsity of the data so that it is harder to learn.

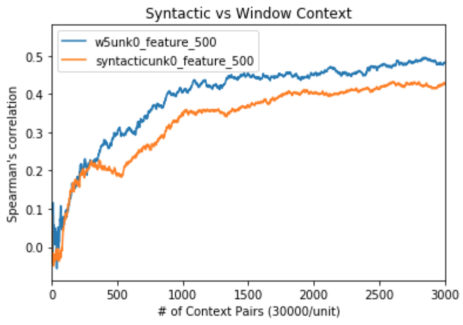


Figure 6: Context Comparison Result

4.1.5 Unknown words handling result

Unknown words handling plays an important part in model performance. Wordnet handling is superior to simply categorizing word below a certain frequency as unknown. This is expected since we are able to associate unseen words with words that we already know. Designing a threshold above 0 to categorize low-frequency words as unknown produces models with much better performance than ones simply keeping all encountered words. The reason is that we cleanse the data by getting rid of low-frequency noisy information.

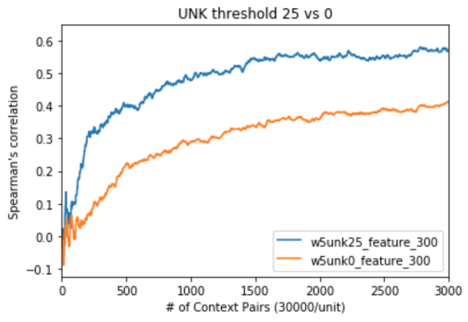


Figure 7: Unknown words handling results

4.2 Model Comparison

We hand pick several examples from test_x to compare the result produced by several models (unk_25, use an unknown token with cutoff threshold 25, wordnet: synonym

id	word1	word2	unk_25	wordnet	syntactic
41	antifeminism	sexism	-0.051079009	6.838058949	-0.055562947
310	planners	notebook	1.864174724	1.135543585	3.72661972
412	headship	position	1.987221837	1.553877234	-0.077157438
540	crudeness	wild	1.935014009	1.724774361	0.214393869

Figure 8: Model Comparison Examples

processing, syntactic: structured skip gram). Id 41 corresponds to a row where unknown words are handled. Wordnet produces a much better result because it can handle words with similar meanings but far different syntaxes. Id 310, 412,530 corresponds to the cases where syntactic model gives a higher similarity score between word pairs that are functionally close and windows context model assigns a higher score to pairs that are topically close. Planners and notebook are functionally close to each other where headship-position, crudeness-wild are topically related.

4.3 Discrepancy analysis

We found that there exists a large difference between our final model's accuracy on development set and test set. One possible explanation is that test set has more low frequency words than development set. In order to validate this hypothesis, we did UNK word analysis. After calculating, we found test set has 16% unknown words with 0 threshold while forming dictionary. The following is the graph describes how the portion of UNK word changes in development with the change of threshold of dropping low frequency word.The red line indicates that we need to increase our low frequency cut-off threshold to 430 to obtain 16 % of UNK word in development set.After we set the UNK word percentage to 16%, our model performs similary on both test set and dev set.

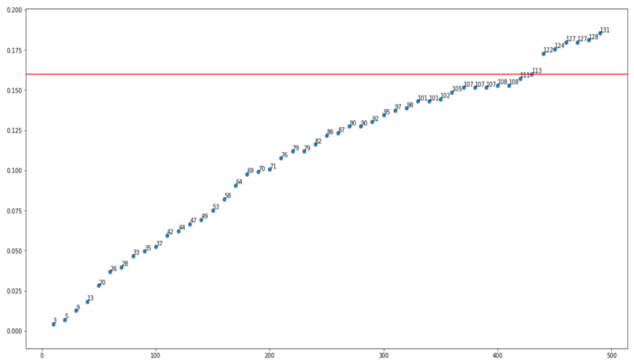


Figure 9: Low frequency cut-off threshold and UNK percentage graph

4.4 Error Analysis

Shown on Figure 10, we list the 8 worst word pairs our model produces on the development set. The Dif column represents the absolute percentage difference between the predicted and actual values. It seems that our model tend to overestimate similarities between certain word pairs. Our hypothesis is that such word pairs appeared in our context many times but they are together just by randomness without a direct meaning association.

	id	predicted	actual	word1	word2	Dif
3	3	1.172968	0.54	noon	string	1.172163
67	67	2.480081	0.92	stock	jaguar	1.695740
102	102	2.165888	0.92	monk	slave	1.354226
200	200	1.641157	0.62	rooster	voyage	1.647027
217	217	1.488560	0.23	king	cabbage	5.472001
249	249	1.322745	0.31	professor	cucumber	3.266918
311	311	2.086663	0.54	chord	smile	2.864190
332	332	2.833160	1.31	stock	CD	1.162718

Figure 10: Word pairs with unreasonable correlation score

marriage	sex bishop abuse law rabbi
network	Internet viewer software infrastructure video
cup	Championship tournament victory Maradona defeating
grocery	bread food index popcorn seafood
Yale	Harvard psychiatry professor journal science
Arafat	Palestinian Jerusalem Israel peace rabbi

Figure 11: Example of words and their 5 associated words in development set

Figure 11 shows some examples words with their top 5 associated words in development set with the embedding trained with large data-set, using 500 dimension of embedding, window size of 5, linear skip-gram, and UNK 25.

5 Final Performance

The following are the five results we submitted on leaderboard.We reported the last row as our final submission.

Data	Epoch	Features	window	Lem/Stem	Unknown	Model	Dev	Test
1 million sentences	10	300	5	No	UNK 0	Linear	0.53	0.2184
1 million sentences	3	300	10	Yes	UNK 0	Linear	0.54	0.1671
1 million sentences	3	300	10	Yes	UNK 0	Linear	0.50	0.1593
1 million sentences	1	300	5	Yes	WordNet	Linear	0.48	0.0971
1 billion sentences	1-2	500	5	Yes	UNK 25	Linear	0.63	0.2954

Figure 12: Model performance

6 Conclusion

We believe that we can achieve higher correlation scores by iterating through more epochs of the large data set. We also believe that synonym processing can assist us in that direction. Given more time, we would also be able submit better results with further unknown words processing and well trained results from syntactic skip gram model.

7 Reference

Levy, O., Goldberg, Y. (2014). Dependency-Based Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). doi:10.3115/v1/p14-2050
Word2Vec Tutorial - The Skip-Gram Model. (n.d.). Retrieved March 18, 2018, from http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/