CS5740: Assignment 2
Github Repository (A2-14)

Yujie Shao ys993 | Snigdha Singhania ss4224 | Yiling Sun ys994

1 Introduction (5pt)

In this report, we present results from our experiment of creating vector representations for English words. These word embeddings can accurately represent the syntactic and semantic similarity between words. The skip-gram model is used with the 1 Billion Word Language Model Benchmark corpus for the experiments, which explore, the raw text and the POS-tagged corpus.

In Section 2, we formalise the Skip-gram model used as inspiration to generate these word2vec embeddings. Section 3 discusses the experiments, at length – particularly the training data prepared using linear bag of words contexts and dependency-based contexts. Results show that the linear BOW model outperforms the syntactical, dependency-based model. Section 4 and 5 talk about the data preprocessing and model implementation. Finally, the results and its analysis is conducted in Section 6. Spearman's ρ of **0.547** and **0.2803** are achieved on the development and test data, respectively.

2 Model (10pt)

Skip-gram is the state-of-art word embedding technique introduced by Mikolov [1], where each word is presented by a dense vector $\mathbf{v_w}$ with dimension $\mathbf{d}$ and each word is associated with its contexts $\mathbf{v_c}$, which can be generated from the corpus. The network learns to predict the context given word by maximizing the corpus probability:

$$\underset{\theta}{\text{argmax}} \sum_{w,c \in d} (log e^{v_c \cdot v_w} - log \sum_{c'} e^{v_{c'} \cdot v_w})$$

However, it is intractable to sum over all context, and we introduced negative sampling to approximate it. More specifically, while true samples were labeled as 1, we created unobserved samples with **negative sampling** and labeled them 0. Mikolov et al. propose a negative sampling rate between 2-20, which depends on the training size [2]. For our experiments, we generate 5 negative samples per positive sample, such that for every (w, c) pair, $(w, c_1), ..., (w, c_5)$ are generated where each c_i is chosen from its unigram distribution raised to 3/4.

In our experiments, Skip-gram is implemented as a binary classification problem, where the objective is to minimize the cross-entropy loss between the positive and generated negative samples using a logistic sigmoid loss function:

$$\underset{v_w, v_c}{\text{argmin}} - (log \frac{1}{1 + e^{(v_c \cdot v_w)}} + log \frac{1}{1 + e^{(-v_c \cdot v_w)}})$$

Our model experiments with 2 approaches for generating contexts **Linear Bag-of-Words** and **Syntactical Dependency-based** Contexts.

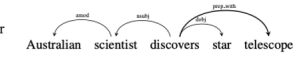
3 Experiments (12pt)

Three experiments are primarily performed to find the most accurate word2vec:

The Australian scientist discovers the star with his telescope.

WORD	CONTEXTS
australian	<start>, scientist, discovers
scientist	<start>, australian, discovers, star
discovers	australian, scientist, star, telescope
star	scientist, discovers, telescope, <end>
telescope	discovers, star, <end>

(a) Linear BOW Context Extraction



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

(b) Dependency-based Context Extraction

Figure 1: Example of Context Extraction

Context: We test the performance using 2 context extraction techniques. In the linear bag-of-words method, we generate (word, context) pairs where k words before and after the target word are its context. Window sizes $k \in \{2, 3, 4\}$ are explored. For the CONLL dependency-based approach, the context is based on structured context dependency tree as explained in [3]. Figure 1 shows an example.

Training Data Size: We experiment with first 200k, 500k and 800k sentences in the benchmark corpus. By limiting the training data the computation speed for the model improve, but comes with a tradeoff of increased frequency of unknown words in the development and test set.

Embedding Dimension: Levy & Goldberg use 300 as the embedding size[3]. We run tests

with 200 and 500 dimensions to find the optimal hyperparameter for our data & objective function.

4 Data

Data (5pt) A subset of the 1 Billion Word Language Model Benchmark corpus is used to train the Skip-gram model for the experiments. Two versions of the dataset are made available: raw textual sentences, and POS-tagged corpus in CONLL format. Some statistics from our dataset are highlighted in Table 1.

Data Size	200K	500K	800k	CoNLL
Subsampling	S_{freq}	S_{freq}	S_{prob}	S_{freq}
Training Set	28M	37M	55M	35M
Dev Set	353	353	353	353
Test Set	2034	2034	2034	2034
Vocabulary	69K	108K	139K	40K
Doc. Length	25.37	25.38	25.38	25.14

Table 1: Dataset Distribution
(S_{freq}: by freq(w), S_{prob}: by $P_{drop}(w)$)

The vocabulary is constructed after **subsampling** and pruning out low-frequency words (count(w) $<$ 5). Context repetition for words which occur very frequently in the corpus is not informative and tremendously increases the volume of training data. Two subsampling techniques were experimented with using the frequency and counts of the word in the training data. In the first technique, we cap every word in the training set to occur at most f times, where all f samples are randomly picked. The second subsampling method is based on a probability, where every word w, is dropped with a probability, $P_{drop}(w) = 1 - \sqrt{\alpha/\text{freq}(w)}$, where α is a hyperparameter for the threshold. We use $f = 2000$ and $\alpha = 10^{-5}$, for the subsampling methods, respectively [2].

Pre-processing (10pt): For raw text, (**A**) and (**Z**) is added to mark the start and end of sentences, after which they are tokenized to words. Words in both raw and CONLL dataset are lowercased, and stop words and unit length words are removed. We also remove non-alphabetic characters as the development and test set do not contain special characters/numbers. Further, as different inflections and derivations of the same word should have similar vector representations, words are stemmed using `SnowballStemmer`. Although stemming is expensive, it reduces the vocabulary size (from 145K to 110K in 500K sentences), which can speed up computation. However, stemming introduces some ambiguity which can cause information lost in some words. For example, both *university* and *universe* are stemmed to *univers*.

Handling Unknown words: The test-set has 256 unknown words. The following hierarchical approach estimates their representation:

- **Synonyms** for every unknown word is retrieved from WordNet. The mean across embeddings of synonyms in the vocabulary is used to depict the word. This method is effective in identifying similar words, but has the risk of slowing down the training model. The embeddings might be less accurate if the word has multiple senses.
- Unknown words are tokenised into **Character *n*-grams**, where $n \in [3, \text{wordlength})$, and the mean across all n-grams which are in the vocabulary is used to approximate the word embeddings. If these character n-grams have semantic meaning, they can add some meaning to the unknown word in the absence of synonyms (eg: *witchcraft* = *witch* + *craft*), with the downside that the n-gram of a word might not have similar meaning with the word.
- **Average across rare-word** embeddings handles the remaining unknown words after the previous steps. We classify rare words as those with less than 20 occurrences in the training corpus. This method is straightforward, but might overgeneralize different low frequency words. However, we see that for training data size $>$ 200K, the first 2 conditions suffice to handling of all unknown words, and the average embedding is never required. It is a fallback when a less data is used.

5 Implementation Details (3pt)

The Skip-gram model discussed in Section 2, adopts a binary classification approach and is implemented using PyTorch. `BCEWithLogitsLoss` is used to calculate the Binary Cross Entropy Loss after passing the results through a Sigmoid layer. We experiment with different embedding dimensions to represent the words and achieve the best result for **300**. The ADAM optimizer with a learning rate of 0.001 is used to the train the model for 20 epochs, where the training data is divided into batches of 8192 points. Further, 5 negative samples are generated per positive (word, context) pair. We split the data into train and validation in the ratio 9 : 1, to check the efficacy of our model while training.

6 Results and Analysis

Test Results (3pt) Spearman's Rank Correlation is used to measure the relation between the predicted and human-rated scores. Table 2 shows

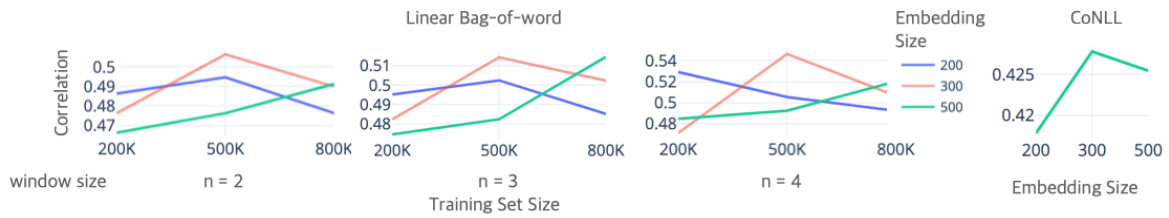


Figure 2: Experimental Results with varying embedding dimensions, train and window size for linear BOW context (Left 3) & Dependency-Parsed CONLL dataset (Right)

the best results obtained for different training sizes on the development and test set. The test set was not run with 800k training sentences because the results were worse on the development set. Results on the test data is considerably poor as compared to dev as the data is 6 times larger and comprises of many unknown and rare words.

Train Size	200k	500k	800k
Development Set	0.539	0.547	0.513
Test Set	0.204	0.2803	N/A

Table 2: Spearman's Correlation best results

Development Results (15pt) Figure 2 shows the results from experiments discussed in Section 3. As suggested in [3], an embedding dimension of 300 works best for the model. It is suspected that the smaller dimensions are unable to capture sufficient semantic information while a larger value is too general. The performances increase as the window sizes increase, since larger window sizes capture more context information. The larger train size does not guaranteed better performance. The results on CONLL-parsed data produce very different results from BOW, as the context generated is different (semantic and syntact, respectively). For the given dev and test data, it is our assumption that the true similarity scores are not created from a functional view, as the parsed data performs considerably terribly. The model is also very sensitive to epochs, and performs well in the range of 15-20, above which it begins overfitting. The best results from these experiments are also summarized in Table 2.

Qualitative Analysis (10pt) In Table 3, the normalised score generated by the skip-gram model is compared with the human-rated score. The difference between these scores are ranked in decreasing order. The table shows top 5 word pairs with greatest difference in estimated values. To further analyze these results, one word is randomly selected from each word pair, and and their vector representations are to other words in the vocabulary using cosine similarity. Table 4 shows

Word Pair	**Similarity Difference**
(Fuck, Sex)	0.792270
(Journey, Voyage)	0.773552
(Dollar, Buck)	0.737723
(Magician, Wizard)	0.735387
(Hotel, Reservation)	0.714395

Table 3: Largest Errors in Prediction

Target Word	**Top 3 Similar Words**
Fuck	Bellowing, Noncivilized, Cock
Voyage	Breather, Dwarfish, Sailing
Buck	Lease, Predetermination, Point
Magician	Conjurors, Rascality, Calcify
Hotel	Airport, Luxuriance, Seven

Table 4: Top Similar Words to Erroneous Pairs

the top 3 words with closest embeddings of the chosen word. It is interesting that word pairs with large difference in estimated scores are nouns with similar meanings. As linear BOW is used for context generation, synonyms rarely occur within the same context, causing their similarity score significantly lower than expected. It is also possible that some of these words have implicit meanings that are seldom mentioned in regular sentences, but can be quickly identified as similar by humans. Further, we observed that words with big differences is similarity have multiple meanings and can be used with different POS. For example, *buck* can be used as noun, verb and adjective and with different meanings, making its embedding less similar to the embedding of *dollar*.

7 Conclusion (3pt)

Our findings show that using training size of 500K, embedding dimension of 300, and linear bag-of-words context generation with window size of 4 yields the best development and test results. The model is simple that the moderate size of datasize gives better correlation score. We observe that nouns pairs with similar meanings receive lower similarity score than expected. Words with multiple senses represent different POS and are likely to generate less accurate embeddings.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[3] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302–308, 2014.