# CS5740: Assignment 2

## https://github.com/cornell-cs5740-21sp/a2--rmh

Rashmi Sinha rs2584 | Malcolm Yang gy86 | Haoran Yin hy522

## 1 Introduction (5pt)

In this assignment, we study famous embedding models to transform the English words to vectors to understand the similarity between two given words.

We adopt the first 1M sentences for "1 Billion Word Language Model Benchmark" (1B) to train our model and evaluate the results by calculating the word similarity against human annotation. After trying various models and fine-tuning hyperparameters, we decided to use skip-gram model with negative resampling. Our best development score is 0.503. And our best test score on leaderboard was 0.132 (This is a version of dev score 0.4). Even though test score was not as good as we expected, there was still fair amount of evidence that our model generated meaningful embedding via a simple skip gram model.

## 2 Model (10pt)

### 2.1 Naive Skip-Gram Model(SG-1)

One classic approach for generating pair of words is examining neighboring words by assuming the high semantic similarity. For example, if we denote the center word is $w_n$ where n is index of word in the context and the window size is $d$, then generated set of word-context pairs should be in the range $(w_n, w_{n-d})...(w_n, w_{n+d})$. Therefore, the task is maximize the likelihood of pair existing in observed data denoted as $D$, which implies the objective function

$$\arg\max_\theta \prod_{w,c \in D} p(D = 1 \mid w, c), \text{ where } p(D = 1 \mid w, c) = \frac{1}{1+e^{-v_c \cdot v_w}}$$

which is a binary sigmoid function, and $v_c, v_w$ is the embedded vectors corresponding to the pair of words.

### 2.2 Skip-Gram Model with Unigram Distribution Negative Sampling(SG-2)

One problem of naive model is only considering positive samples and fails to consider the negative cases about which pair should not exist, leading to trivial results. In this method we sample from 0.75 of the unigram frequency distribution to get negative samples. This is under the assumption that fetching words from long tail distribution will give examples not associated with the current word. Derived from the 2.1, the objective function is now minimizing the negative log of

$$\arg\max_\theta \prod_{(w,c) \in D} p(D = 1 \mid w, c) \prod_{(w,c) \in D'} p(D = 0 \mid w, c) \text{ where } p(D = 1 \mid w, c) = \frac{1}{1+e^{-v_c \cdot v_w}}.$$

To form the negative set, we simply pull out the complement from $D$ denoted as $D'$. The negative samples were sampled following the distribution $p^{\text{NS}}(w) = \frac{f(w)^{\frac{3}{4}}}{\sum_{w'} f(w)^{\frac{3}{4}}}$ in $D'$.

### 2.3 Linear Skip-Gram Model with Rejection Negative Sampling(SG-3)

This is similar to the above model for the most cases except for the method of sampling the negative cases. This gets rid of the assumption made in the previous model and checks for every negative sample generated whether it belongs to the $(input, context)$ pairs present. If a negative sample is present in any input-context pair, it resamples the negative example from the vocabulary.

### 2.4 Continuous Bag-Of-Word Model

This model uses a window similar to what is explained in the Naive Skip Gram Model. This model is very similar to the Naive Skip Gram Model(SG-1) apart from the fact that the distributed representations of the context are combined to predict the word in the middle i.e input word, while in Skip-Gram model the distributed representation of the input is used to predict the context.

## 3 Experiments (12pt)

We conducted our experiments across different models explained in the previous section. For SG-1 model, we tuned the following hyperparameters - window size, learning rate, embedding dimensions, number of epochs, batch size and gamma for schedular. We also experimented with different sizes of data and combinations of data from the 1 Billion dataset provided. For the SG-2 and SG-3 models, in addition to the above experiments, we also tuned the number of negative samples. For the CBOW model, we only experimented with window size for a fixed set of data we found working well for the other models. Apart from this, we also experimented with other subtleties which were not model specific such as applying unit normalization to the final embedding matrix and various techniques of handling unknown words. For the unknown words embedding we experimented with three techniques: (1) Conditionally adding more data from 1B dataset containing unknowns, (2)coming up with synonyms (including hypernyms and hyponyms) of all unknown words and looking if these words existed in our vocabulary. If it existed, the embedding of the unknown word was replaced with its synonym present in the vocabulary and (3) Replacing rare words with 'UNK' token and using this to fetch unknown word embeddings.

## 4 Data

**Data (5pt)** We mainly used first 1M sentences of 1B billion word language model benchmark dataset in Conll format. We used 100K subset of it to tune parameters such as window size, negative sample size and batch size and then included 1M sentences and conditional sampling from remaining dataset. There are 706 words pairs in development set and 2034 words in the test set. The vocabulary size is 262,952 without prepossessing in 1M training set. We computed these words counts by maintaining a dictionary where key word string and value is the word count when traversing the dataset.

**Pre-processing and Handling Unknowns (10pt)** For preprocessing, we changed every word in training, development and test set to lowercase. We also replaced non-alphabet characters with empty string using regular expression. We used NLTK package to remove English stop words and get stemming of the word in the training set. We also ignored sentences containing only single word after stop word removal as they had no associated context. We also tried to remove uncommon words which only under certain frequencies in the dataset although not improved our model performance. We first tried to use the subset of data to train our model since it's more fast to run. But entire 1M was used for training as it gave better results.
We used three approaches to handle unknown words. First we tried to use a larger dataset, we traverse the entire 1 B dataset and conditionally getting sentences containing unknown words. Second, we used NLTK to replace some words with their synonyms in existing vocabulary. Third, we tried to create an "unknown" token in vocabulary and replaced rare words in training set with this token and got it's embedding.

## 5 Implementation Details (3pt)

To implement above models, we used the Pytorch framework. We used batching and GPU to quicken the training. We use Adam for optimization in SG models. For SG-3 model, to make training more efficient, we generated negative samples during training instead of beforehand to consume less RAM. We also split the vocabulary into four portions and randomly selected one portion to fetch our negative samples to save time. In SG-3 Model, we constraint the negative samples to not be same as the input word. For the CBOW model, we used a hidden layer same as the size of the input layer and used Relu as objective function.

## 6 Results and Analysis

**Test Results (3pt)** The best test results obtained was 0.132 on the Skip Gram Model with Rejection Sampling(SG-3) having number of epochs = 3, window size(d) = 2, number of embedding = 300, learning rate = 0.003, batch size = 512, schedular gamma = 0.1

**Development Results (15pt)** Due to constraint of space, we only focus on ablation of 4 major parameters which had the most impact on the development scores: amount of data, number of negative samples, number of training epochs and window size. Table 1 contains a synopsis of the development results where "ep" denotes number of epochs, "neg" denotes number of negative

| System | Correlation |
|---|---|
| **Development Results** | |
| *100K Connl Data, ep = 2* | |
| SG-1 (d = 2) | 0.014 |
| CBOW (d = 2) | 0.023 |
| SG-2 (neg = 5, d = 2) | 0.054 |
| SG-3 (neg = 5, d = 2) | 0.193 |
| *100K Connl Data, ep = 3* | |
| SG-2 (neg = 5, d = 2) | 0.094 |
| SG-3 (neg = 5, d = 2) | 0.254 |
| SG-2 (neg = 10, d = 2) | 0.156 |
| SG-3 (neg = 10, d = 2) | 0.253 |
| *1M Connl Data + Sentences from 1B Data, ep = 3* | |
| SG-2 (neg = 10, d = 2) | 0.301 |
| SG-3 (neg = 10, d = 2) | 0.41 |
| SG-3 (neg = 10, d = 2) + unit norm | 0.343 |
| SG-3 (neg = 10, d = 3) | 0.48 |
| SG-3 (neg = 10, d = 4) | 0.5 |
| **Test Results** | |
| SG-2 (neg = 5, d = 2, ep = 3) | 0.067 |
| SG-3 (neg = 10, d = 2, ep = 3) | 0.132 |

Table 1: Correlation Scores for various models

samples and "d" denotes window size. Since the data set provided in this project was large, we performed ablation analysis of four models on a smaller subset of the data, i.e 100K entries of Conll data. We observed that SG-2 and SG-3 gave relatively higher results and decided to go ahead with those. We tried increasing the number of epochs and saw that it marginally increased the performance. We believed that adding more data will further improve the performance. Hence we decided to go ahead and train on 1M Conll dataset. We also scraped some data from the 1B dataset to handle the unknown words presented in the test set. We observed a significant improvement in performance on doing so. Then we went on to perform unit normalization on the data as we believed it should help improve the correlation. But we saw that the performance on the test set decreased and hence rejected this assumption. Then we tweaked around with the window size. The reason we decided to do this towards the end was because this was an expensive process and increased the data multifold. We wanted to keep our earlier iterations quick and decide on the other hyperparameters before tuning this one. The best final development model was SG-3 with the following parameters - number of epochs = 3, window size(d) = 4, number of embedding = 300, learning rate = 0.003, batch size = 512, schedular gamma = 0.1.

**Qualitative Analysis (10pt)** To better understand where errors come from, we analyzed the result from development result. We denote the misalignment error for each pair by $|Similarity_{prediction} - Similarity_{groundtruth}|$, in fashion of L1 norm and list the 5 worst predictions:

| first | second | Error |
|---|---|---|
| tiger(813) | tiger(813) | 50.28 |
| benchmark(545) | index(2473) | 19.94 |
| bread(315) | butter(249) | 18.49 |
| psychology(187) | psychiatry(84) | 18.09 |
| chord(68) | smile(403) | 16.28 |

comparing the 5 best prediction:

| first | second | Error |
|---|---|---|
| smart(813) | stupid(410) | 0.0061 |
| planet(814) | astronomer(51) | 0.0158 |
| concert(823) | virtuoso(28) | 0.0212 |
| morality(106) | importance(797) | 0.0586 |
| game(10468) | team(10505) | 0.0669 |

the number in brackets implies the frequency appearing in training set. Based on our evaluation metric, we address 2 major issues: (1)we intentionally excluded comparing word by itself in training process because we believe it is trivial. However, the development data requires self-comparing. For example, ("tiger","tiger"). If we consider this case we should have improvement. (2)The training data is sufficient. Even if some words such as "astronomer" and "virtuoso" only appeared very few, our model still successfully handle them via narrative training.

## 7 Conclusion (3pt)

In this assignment learned how to deal with large scale data for training our models and had to carefully perform ablation analysis and use our resources in an efficient manner to obtain maximum output in limited time. From the numerical result, we observed that negative sampling model with larger window size, larger embedding size, more negative samples and smaller batch size gave us a better development set score. We also found that negative samples with constraint and prepossessing except stemming improved model performance. Larger dataset could solve some of the unknown words problem while synonyms and unknown token had limited effect. From qualitative analysis we could conclude that our model successfully handled some rare words however didn't take the similarity between word and it self into consideration. If we included this constraint in our model we could get even better performance. This assignment gave us insight into learning about various models and modelled real world challenges of NLP.