Summer 2020

Empirical Analysis of CBOW and Skip Gram NLP Models

Tejas Menon
Portland State University

HONORS THESIS

Empirical Analysis of CBOW and Skip Gram NLP Models

Tejas Menon

Tejas2@pdx.edu

Advisor: Dr. Anthony Rhodes

Portland State University

BACKGROUND

Skip Gram and CBOW are two popular bigram NLP models to create word embeddings. They were devised by Mikolov et. al. in their seminal 2013 paper *"Efficient Estimation of Word Representations in Vector Space"* which grew in popularity due to the improved performance and efficiency of studying feature-heavy, multi-million sized word corpora in an unsupervised manner[1]. Previously, N-Gram models were significantly slower to train, and produced less accurate embeddings requiring more space[1]. The development of these models combated these problems, and paved the way for more recent 'GloVe' and state-of-the-art 'Infersent' models that are largely specializations and scalability enhancements over the canonical models[2]. Therefore, Skip Gram and CBOW can be regarded foundational structures of word embedding development in the field of NLP, studying which will help discover patterns that may be applied more universally.

Briefly, the two models are two-layer, fully-connected neural networks with linear activations in the hidden layer and a host of activation choices for the output layer: from SoftMax activation which produces a multinomial posterior distribution, optimizations can be applied such as negative sampling and hierarchical softmax[3]. The former choice is used for this paper, the details of which will be discussed later; in short, the decision was made due to implementation complexities of hierarchical softmax against diminishing returns. Another similarity of both models is its layer configuration: layer 1 is a $v * N$ matrix and layer 2 is a $N * v$ matrix where v is the number of words in the vocabulary and N is the size of the hidden layer. Each training example consists of neighbor context words and a center word. The objective of CBOW is to learn to predict the word in the center from the context words and the objective of Skip Gram is to learn to predict the context from the center word. Accordingly, in Skip Gram, a one hot encoded vector (input) ‘selects’ one row from the hidden layer and this is the vector representation of the word at this index in the corpus. Then, taking the dot product of the vector representation of this word with each column in the output layer and passing it through softmax gives the probability score of this word being in the same context as words in the output layer. In CBOW, learning the opposite results in taking the mean of rows in the hidden layer (context words) and taking its dot product with each column in the output layer to give the probability score of the word in the output layer being the center of the context. The images below signify effectively the inverse nature of both models[4]:

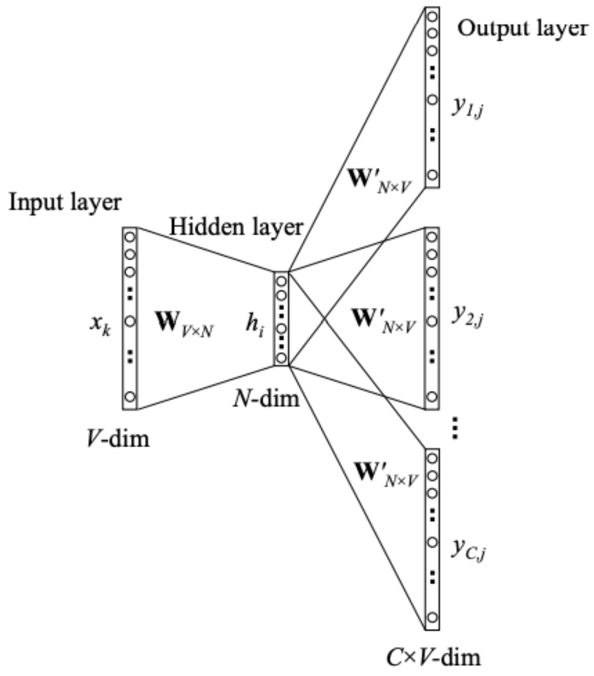

Figure 1 Skip Gram Model

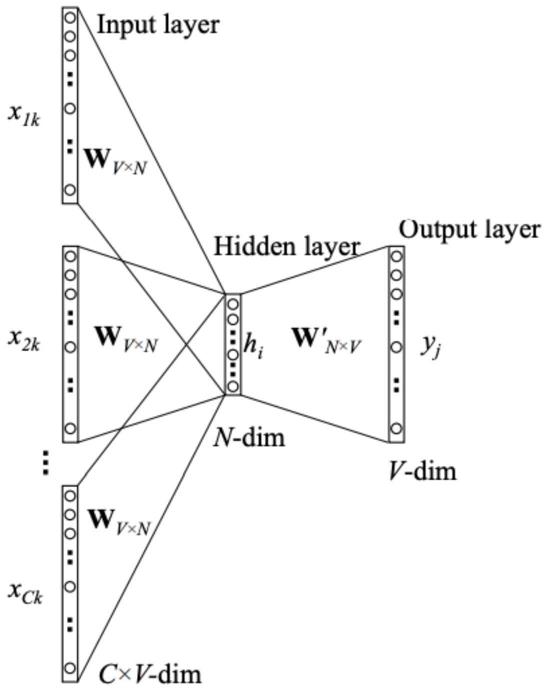

Figure 2 CBOW Model

Training consists of preparing context + center pairs over the corpus and using stochastic gradient descent as the optimization algorithm to learn hidden and output layer weights. It has been shown that Adam/Sparse Adam are good contenders to the traditional algorithm using the same loss function for faster training due to learning optimization – Sparse Adam doesn’t update all the weights during backpropagation and supports a dynamic learning rate. Along with negative sampling which greatly improves performance by order of magnitudes, this dual choice seemed apt and was therefore used for this analysis project.

The usage of negative sampling is crucial for training on multipurpose, (very) large corpora for the simple reason that the hidden and output layer are sized by the number of words in the vocabulary which can be hundreds of thousands of words. Updating all the output layer weights and backpropagating this loss would require millions of weight updates for each training example which is infeasible given hundreds of thousands of training examples[4]. The innovation of negative sampling thus is to sample from a noise (random) distribution a few 'negative' context/center words to train against for each training example which over the training period would propagate to most data[4]. The size of the negative sample per training example affects the training speed and the larger the size of the sample the better the accuracy of the model. The given loss (objective) is advised per the second word2vec paper by the same authors[3]:

$$E = -\log \sigma(\mathbf{v}'_{w_O}{}^T \mathbf{h}) - \sum_{w_i \in \mathcal{W}_{\text{neg}}} \log \sigma(-\mathbf{v}'_{w_i}{}^T \mathbf{h})$$

where $\mathbf{h} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{v}_{w_c}$ for CBOW (mean of context words) and $\mathbf{h} = \mathbf{v}_{w_I}$ for Skip Gram (center word). w_i are the words in negative sample to train against (wrong context words for Skip Gram and wrong centers for CBOW) and w_O is the index of the target word (correct center for CBOW and (one of the) correct context words for Skip Gram). This expression does not produce a well-defined softmax posterior distribution as before but works well to produce high quality word embeddings. A brief insight:

increasing the $\mathbf{v}'_{w_O}{}^T\mathbf{h}$ center/context word inner sum in the first term of the objective decreases E while decreasing the $-\mathbf{v}'_{w_i}{}^T\mathbf{h}$ center/context word negated inner sum in the second term of the objective increases E as expected.

Another optimization strategy for these models is subsampling: reducing the chance of picking a frequent word as either the center or context to train for in a given training example. This reduces overtraining on common articles such as 'the', 'to' & 'a' which skews word embeddings as these words are likely to appear in every context and therefore is highly likely to be predicted as center/context. This can unintentionally perturb subtle relationships between *useful* related word vectors. Training samples are sampled according to the probability[3]:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(w_i)$ is the frequency of a word in the corpus and $P(w_i)$ is the probability of discarding this word. t is a randomly selected constant seeded to some small value such as 0.001. As expected, more frequent words have a higher probability of being discarded from the training sample.

METHOD

Primarily, the goal of this research is to examine the usefulness of word embeddings created using CBOW and Skip Gram under two varied corpus genres – technical (non-fictional) and creative (fiction) literature – to probe the influence of corpus style on word embeddings use cases and applications. I conducted a toy sentiment analysis using a manually curated dataset of phrases mapped to sentiments and also a subject analysis which maps words to concepts/subjects instead. Specifically, the 'Aristo-mini' 18-million-word sized corpus which consists of science, geographical and historical facts was used to represent a technical genre and a 26 million word sized collection of Sci-Fi novels to represent a creative, fictional genre.

EXP 1: SENTIMENT ANALYSIS

Common spoken and written language usually has sentimental value attached to them efficacious to learn given the enormous applications in computing. My analysis consists of simple, unambiguous phrases mapped to three buckets of similar sentiments—positive, negative and neutral—which will be tested on trained CBOW models from each corpus genre. It is hypothesized that the creative, fiction literature with rich emotive contexts would outperform the technical corpus model due to the largely neutral nature of a technical corpus. Training examples consist of phrases which form the context vectors of the first layer of CBOW and are averaged to be passed into different sentiment words in the output layer. If the predicted sentiment is found in the bucket of the target sentiment, the example is predicted correctly and the accuracy score is incremented. As there are three total buckets with an equal number of test examples for each bucket, better than chance prediction would be indicated by accuracy above 33%.

EXP 2: SUBJECT ANALYSIS

Prediction of a topical subject given a word is the logical counterpart to sentiment analysis. The greater volume of factual, conceptual data within technical corpus may allow trained models to perform better in this segment, and since this experiment uses individual words instead of phrases as test input, we can directly use the hidden layer word embeddings of the Skip Gram model to calculate cosine similarity between the input word and subject words in each bucket. If the subject of the closest word matches the target subject, the example passes and accuracy score can be incremented. The choice of cosine similarity instead of other similarity measures such as L2 norm is used since high dimensional vectors may greatly differ in vector magnitudes during training. The optimization step also persuades similar word vectors closer together in terms of angular separation instead of proximal separation which is another reason to use cosine similarity[4]. As in sentiment analysis, there are four subject buckets—academics, world, nature & lifestyle-- common to both corpora and each of which have 6 subject words to predict from. Unlike before, an accuracy score >25% can be considered better than chance.

IMPLEMENTATION

The models, test environment and data reader were created using the PyTorch library with its Autograd features for tensor weight calculation and backpropagation given loss equation and optimization algorithm. Each layer was developed manually however, along with all the forward propagation steps. Each model is created by extending the `torch.nn.Module` base class with all the interface methods implemented such as `Module.forward()` and `Module.save_embedding()`. The data reader is an extension of the `torch.dataset` class and uses the build-in auto batch sampler by implementing `__getitem__ ()` for the batch iterator. This function reads the next line in the corpus and attempts to match words in this line with the corpus vocabulary generated during the initial passthrough of the corpus. The initial passthrough is responsible for creating the discard list for subsampling, the negatives list for negative sampling and also a min-count filter which force excludes words from the vocabulary if their frequency is below a certain value. This prevents words spelled incorrectly & stray characters from making it into the vocabulary. For Skip Gram which is benefited by CUDA parallelization, this built-in PyTorch feature is also turned on for faster training.

RESULTS

Firstly, good hyperparameter values for learning rate, hidden layer dimensions, batch size, and window size were to be chosen prior to fully training a model over either corpus. This was since the datasets were very large and consumed 8 hours per epoch for CBOW and 3 hours per epoch for Skip Gram. For this initial spot-checking, 9 CBOW algorithm models with different hyperparameter values were trained for 5 minutes each and the hyperparameter configuration with the lowest training error at the end of the time period were chosen for full training. Listed below are training loss curves for different hyperparameter values:

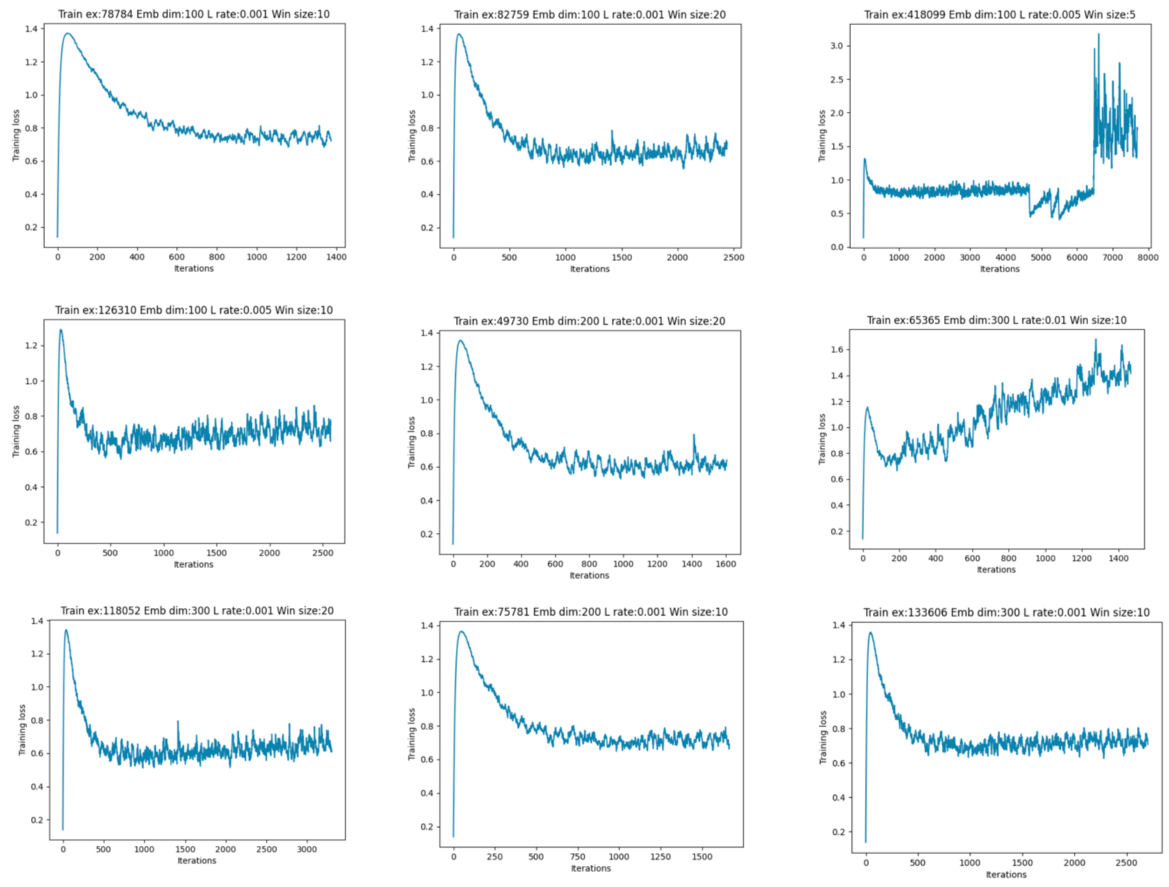

Figure 3 CBOW hyperparameter analysis to select values for longer training

As expected, the recommended hyperparameter values of embedding size = 100, η = 0.001, window size = 10, batch size = 8 was a good compromise between error loss and performance. These were chosen for final training on both models over the whole corpus and persisted to disk for later experimentation with sentiment and subject analysis. The final models trained on each corpus recorded the following error curves:

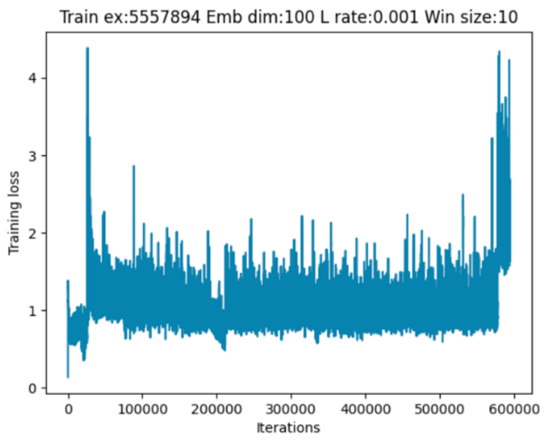

Figure 4 CBOW Aristo-mini Non-Fiction

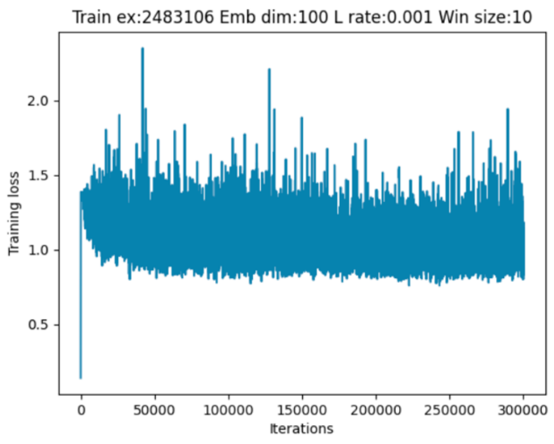

Figure 5 CBOW Sci-Fi Fiction

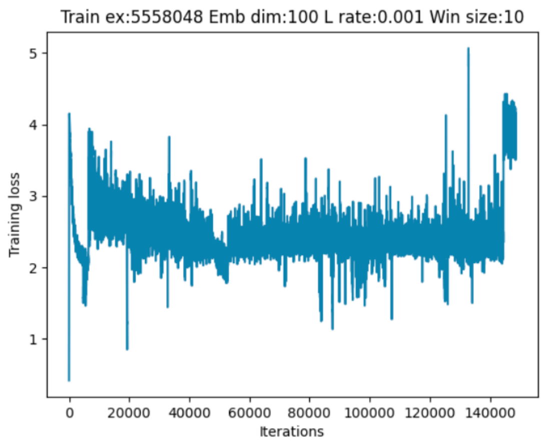

Figure 6 Skip Gram Aristo-mini Non-Fiction

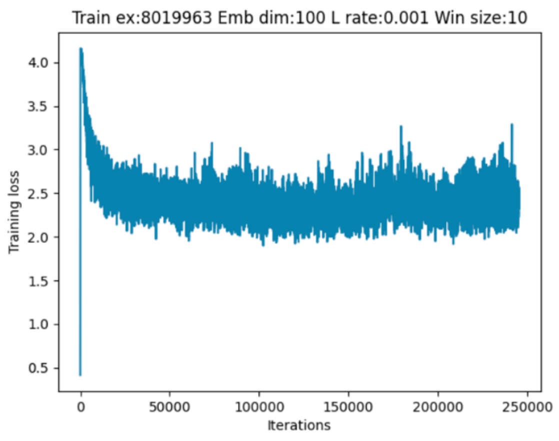

Figure 7 Skip Gram Sci-Fi Fiction

The lower training error recorded for CBOW models are due to the different error loss function and therefore not comparable. The variability of the error curves for the Aristo-mini dataset can be attributed to the topical nature of the corpus—different regions are very disparate and therefore learnings from one region do not transfer over well to the other. This is also a side effect of technical subject matter in different domains which do not generalize word embeddings well. For Skip Gram, a smooth curve in training indicates the uniformity of Sci-Fi novel topics, subjects and literary patterns; the model is able to apply word embeddings from one part of the corpus to the other successfully. The importance of continuing to train over the entire corpus even after observing no significant improvements to training loss is for improving the variance over the model across different topics—though the error may not improve, better generalized performance can be observed given novel input.

Next, results from sentiment analysis are as follows—these are sentiment word embeddings projected down to three dimensions using PCA in Tensorboard:

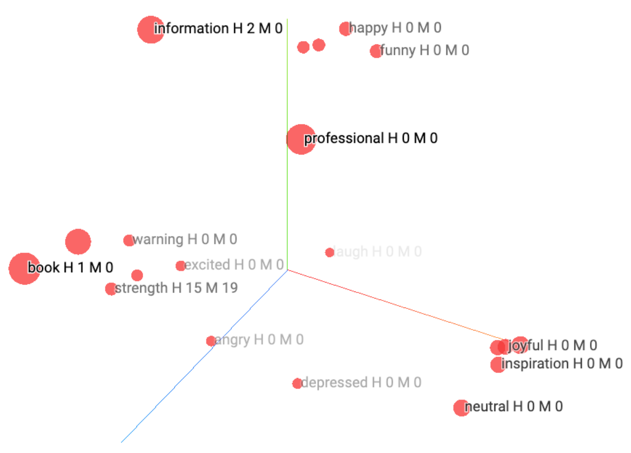

Figure 8 Sci-Fi Corpus; H(hits),M(misses) Accuracy:56%

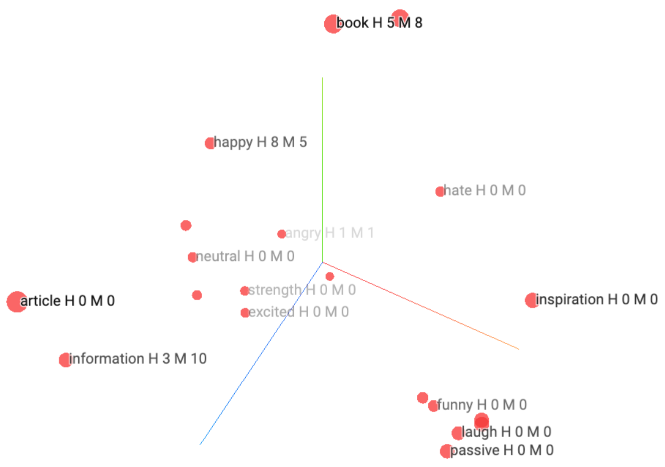

Figure 9 Technical Corpus; H(hits),M(misses) Accuracy:39%

Sci-Fi corpus performed much better than chance (2x) while the technical corpus slightly better than chance at classifying phrases into the three buckets. The Sci-Fi corpus is also skewed towards predicting positive sentiments (‘strength’) significantly, while the technical corpus predicts neutral sentiments much more often (‘book’, ‘information’). While corpus differences can be attributed this result, it must be emphasized that a toy analysis with few phrases isn’t conclusive. A more generic and multi-authored testing sample may be needed for eliminating biases in testing data and classifying a wider range of sentiments. Interestingly, noticeable here is also the clustering of sentiments into different areas of the plot—emotions from the different buckets seem to be more or less together with few exceptions which can be a negative side effect of dimensionality reduction.

Next, results from subject analysis report the opposite trend as expected—a Skip Gram model trained on a fictional corpus performs worse at predicting word subjects than a non-fictional corpus:

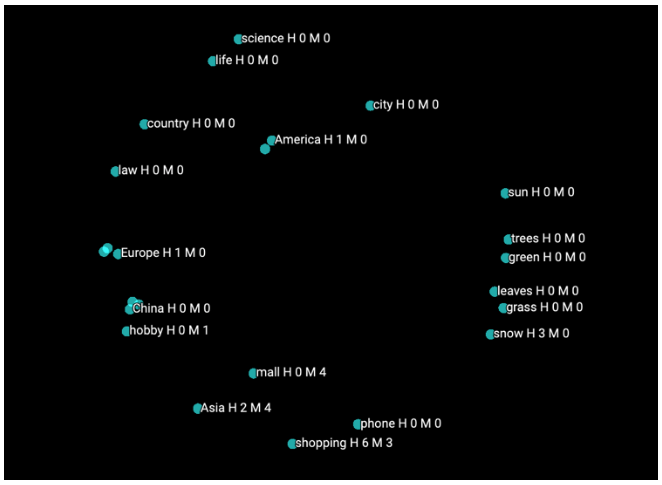

Figure 10 Sci-Fi Corpus; H(hits),M(misses) Accuracy:56%

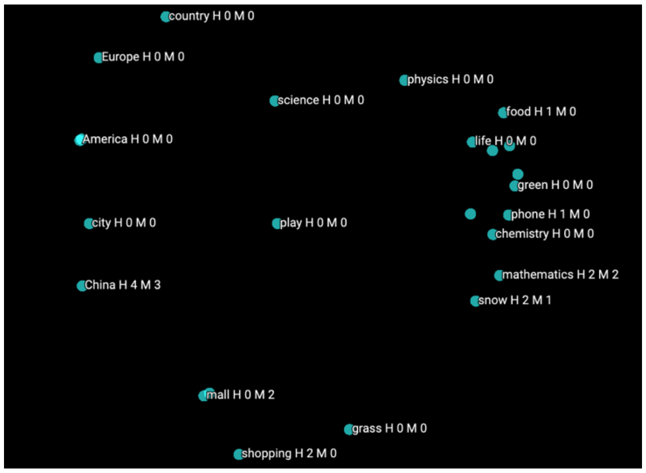

Figure 11 Technical Corpus; H(hits),M(misses) Accuracy:66%

Considerable dispersion of word embeddings in 3d projection meant that 2d projection needed to be used instead as the labels were blank in Tensorboard. From the three subjects – lifestyle, world and nature – the fictional corpus predicts lifestyle more often (‘Shopping’) while the fictional corpus predicts world (‘China’). The accuracy of the models are also significantly better than chance (2x for Sc-Fi and 2.5x for Technical) which is expected as word input instead of entire phrases are easier to classify and use for cosine similarity analysis. As before, a larger variety of subjects and input words will need to be tested at larger scale to reliably rate each corpus strengths and weaknesses. Both projections also show a much better clustering of words in similar subjects—the Sci-Fi corpus concentrates nature data towards the right and world data towards the left while the technical corpus concentrates academic data towards the right and world data towards the left. The clustering is imperfect due to dimensionality reduction losses in 2d.

CONCLUSION

This project confirms good word embedding results for default algorithm and hyperparameter choices for CBOW and Skip Gram. The accuracy scores for Sentiment and Subject Analysis paint an incomplete picture of the efficacy of the models due to various reasons:

- State-of-the-art sentiment analysis models train on labelled data, and very large volumes of it
- Unsupervised NLP has also observed many developments that are unutilized in this paper such as GloVe and Infersent[2]
- The training & test data sizes are inadequate for good performance on a wide variety of literary structures, sentiments & concepts

The findings of the research suggest that corpus style and content has a large influence on the quality of word embeddings and are in essence, specialized to the language corpus. This gives motivation for different applications to use different corpora for training—generating a general model that excels at all applications may require training on a very large volume of data or highly specialized models. The findings also hint at better performance of technical corpus on subject based classification and creative corpus on sentiment classification although further research with larger test datasets and more examples of technical and creative corpora is definitely necessary to make firm conclusions.

REFERENCES

[1] Mikolov, et al. “Efficient Estimation of Word Representations in Vector Space.” ArXiv.org, 7 Sept. 2013, arxiv.org/abs/1301.3781.

[2] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1162

[3] Mikolov, et al. “Distributed Representations of Words and Phrases and Their Compositionality.” ArXiv.org, 16 Oct. 2013, arxiv.org/abs/1310.4546.

[4] Rong, and Xin. “word2vec Parameter Learning Explained.” ArXiv.org, 5 June 2016, arxiv.org/abs/1411.2738.