An analysis and comparison of *word2vec* and *GloVe* embeddings for fake news text classification

Team 6: Malcolm Smith Fraser, Han Gong, Wilson Huang, Zoe Zhu, Tommy Tseng

Abstract

In this report we explore global vectors for word representation (*GloVe* vectors), and discuss what needs they were created to solve, how they are constructed and how the method compares to *word2vec*. As part of the comparison, we explore the differences in the types of similar words the two methods produce and briefly discuss the mathematical reasoning behind that. We also train two *LSTM* classification models to identify fake news with both a *word2vec* and a *GloVe* embedding layer – which perform identically.

Word2vec and *Glove* are built using different methodologies which is clear when examining the words these vectors generate when given a target word, however, these differences were not found to effect performance of a binary classification task using and *LSTM* neural model.

Contents

Introduction

In natural language, words carry meaning through symbols. However, while symbols are easy for humans to understand, such comprehension has been proven very difficult for computational systems. To address this problem, researchers introduced word embedding methods. Word embedding, also known as distributed word representation, represents any given word in vector form. The resulting vector of numbers, known as a word vector, is associated with the meaning of that given word in a mathematical form.

Word embedding can capture both the semantic and syntactic information of words. With word vectors, researchers can analyze the similarities between words directly (word similarity and word analogies). Furthermore, word vectors can be used as input features in various models for high-level tasks including sentiment analysis, topic modeling, and topic classification.

In this report, we evaluate two word embedding methods – *word2vec* and *GloVe* – and compare their performance in an *LSTM* driven fake news detection model.

Background

Related work

Before *word2Vec*, words were encoded through statistics. One commonly used technique was *SVD* (Singular Value Decomposition), which is a dimensionality reduction method on a document by term matrix. *SVD* is then used to process *LSI* (Latent Semantic Indexing) which can produce term to term similarities, document to document similarities, and term to document similarities.

However, since *SVD* is based on matrix decomposition, the dimensions of the matrix change when the dictionary changes, and the whole decomposition must be re-calculated when we add a word. It is also very sensitive to word frequency imbalance, so we often must pre-process the documents to remove stop-words and normalize the corpus (through lemmatization or stemming). As a result, *SVD* is computationally expensive and not suitable for large dictionaries or corpora.

Word2vec

To address the need for a simpler methodology, Mikolov et al. introduced *word2vec*, which is an unsupervised learning method to develop embeddings that capture relationships between words and documents. *Word2vec* follows one of two main methods, either predicting a target word based on surrounding words, known as continuous bag-of-words (*CBOW*), or predicting surrounding context words based on a given target word, known as skip-gram (*SG*). To start, *word2vec* initializes all focus word vectors and context word vectors with random weights. Then *word2vec* extracts a word window (focus word in the middle, certain number of context words before and after the focus word) and calculates the vector values to predict the pattern of words in the word window. As a result, words that share context, and contexts that share many words lay near each other in the vector space.

The key to the enhanced training efficiency of *word2vec* lies in the fact that there is only one hidden layer required for both *CBOW* and *SG* models. Therefore, it is fast to train and can easily scale to massive datasets. Unlike methods like *SVD*, *word2vec* also does not build or store the actual matrix in memory. In

addition, techniques such as sub-sampling of frequent words and negative sampling further enhance the efficiency of *word2vec* by bringing down frequent terms and bringing up infrequent terms.

However, because *word2vec* is a window-based model, it can only obtain information from a given window-length, leading to limited predicting performance. Also, *word2vec* cannot handle out-of-sample words. For words that did not appear in the training set, *word2vec* will not be able to predict since those words were not vectorized beforehand.

GloVe

Based on *word2vec*, Pennington et al. developed an unsupervised learning algorithm called Global vector for word representation (*GloVe*) in 2014 [7]. The team's aim was to improve word prediction performance with word statistics over the entire corpus. The main concept in *GloVe* involves building a counting-based co-occurrence matrix for all unique words, and then calculating the probability ratios for two given words (*i* and *j*) with respect to a probe word k.

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} where\ X_i = \sum_k X_{ik}$$

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 1. This example shows how the co-occurrence matrix is used to compute whether two words are highly correlated. Taking k = solid as an example, the probability for k=solid given the word ice is 1.9 x 10-4. This conditional probability is calculated by number of times word ice and word solid show up together divided by number of times word ice appears in the entire corpus. If the word solid often shows up with the word ice, the conditional probability will approach to 1, and vice versa.

Dividing the conditional probabilities for our context words gives us a ratio probability. If the target word, *k,* is disproportionately correlated to word *i* (in the numerator) than word *j* (in the denominator), the probability ratio will be large (>>1). On the other hand, if word *k* correlates more strongly with to word *j* in the denominator than word *i,* the probability ratio will be smaller (<<1). However, if the conditional probabilities for word *k* – given words *i* and *j* – are equally high or low, the probability ratio will be close to 1. As such, given a target word, *k, and* the two other words *i,* and *j, GloVe* vectors can effectively identify relevant words (words that have a unique affinity of a specific word) and disregard irrelevant words (words that are universally common, or that never show up in the target word's context). In the example, the word ice is more correlated with the word solid than to steam is with solid, which aligns with our common understanding to ice and steam as different state of water.

The global co-occurrence aspect of *GloVe* has been shown to carry two unique advantages, better performance on analogy tasks than word2vec, and having more context information for each trained word. However, *GloVe* is limited in that storing the co-occurrence matrix requires significantly more memory than *word2vec*.

Approach/Methodology

Data Collection and Pre-processing

The dataset we are using is the *Liar* dataset published by William Yang in July 2017 containing over 10k+ labelled data of fake news content. The target feature has six levels including true, mostly true, half-true, barely true, false and pants-fire.

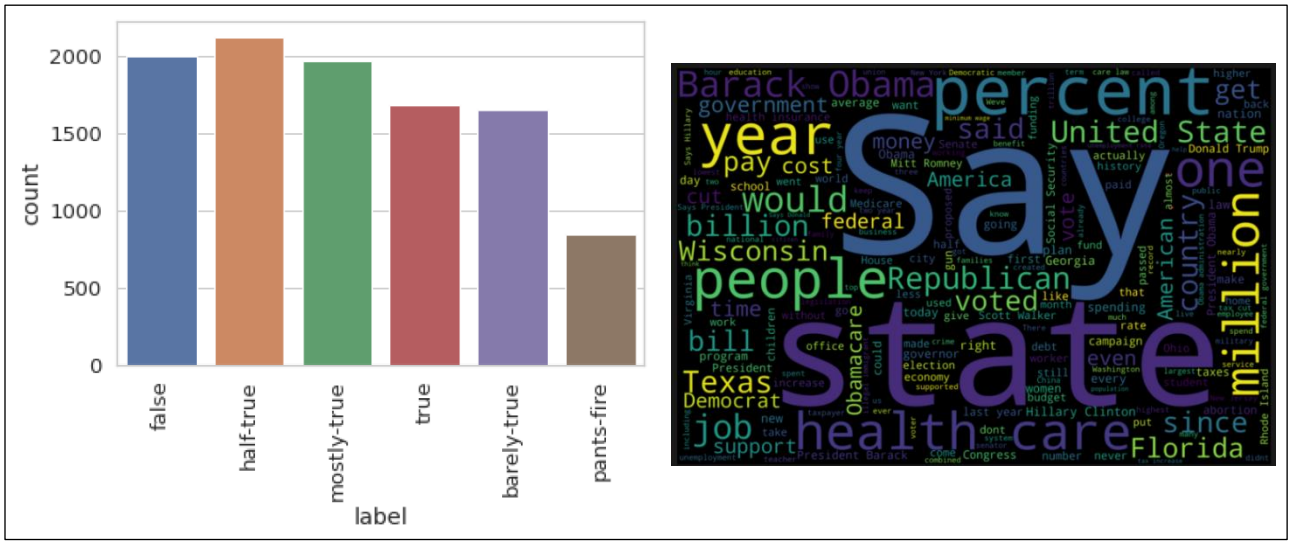

Figure 2. Shows the distribution of the label categories in the Liar dataset and a word cloud displaying the most common words in the corpus.

For the NLP implementation and word embedding comparison, the 14 features in the original dataset were trimmed to two – the class labels and the news statement. To improve the embedding performance and classification accuracy, square brackets and stopwords were removed using the NLTK package, as these words/symbols do not carry any significant amount of meaning.

Word Embeddings

Testing the efficiency of the two word embedding methods on the news content involved two major tasks.

In the first task, *word2vec* and *GloVe* were fit to the training data from the *Liar* dataset. To fit the word embedding models, news statements from training dataset were tokenized using NLTK package and organized into a "list of lists". The glove embedding was trained using the *glove_python* package created by *maciejcula* [4], following the co-occurrence ratio calculation defined by *GloVe*. The *word2vec* model was trained using the *gensim* library using the same tokenized training words as for *GloVe*. After creating both embedding vectors, similar words were generated for various target word examples, such as "president". The results of the similarity task were used to examine the difference in the matrix construction methods and predicted the outcomes of embedding layers in the classification model.

In the second task, pre-trained embeddings were selected, imported, and transformed to be used in the final classification model. Although the self-trained embeddings are sufficient for similarity task, more advanced embeddings trained by large corpus such as Wikipedia, Twitter and Google News can increase the performance of the embedding layers in the complex classification model, especially for neutral networks. To ensure the fairness of the comparison, the pretrained vectors were all trained on Wikipedia

and had 100 dimensions. The pretrained *GloVe* vectors were pulled from Stanford's *GloVe* website [7], and the pretrained *word2Vec* vectors were pulled from Wikipedia2Vec. In the transformation process, a dictionary mapping word indexes to the corresponding embeddings is constructed and used to compute the embedding matrix by looping over the training corpus.

- ***Word2vecWiki**:* Pertained *word2vec* embeddings of 100 dimensions obtained from Wikipedia2Vec. These are the embeddings used to vectorize our fake news statements to generate the embedding matrix for our *word2vec LSTM* model.
- ***GloVeWiki**:* Pretrained *GloVe* embeddings of 100 dimensions obtained from Stanford glove project website. These are the embeddings used to vectorize our fake news statements to generate the embedding matrix for our *GloVe LSTM* model.
- ***Word2vecFN**:* In addition to the *word2vecWiki* vectors, we also trained our own set of word vectors on the corpus of fake news using the *Gensim wor2vec* package. These vectors were used to analyze the top similar words in the corpus.
- ***GloVeFN**:* In addition to the *GloVeWiki* vectors, we also trained our own set of word vectors on the corpus of fake news using the *glove_python [4]* package. These vectors were used to analyze the top similar words in the corpus.

Modeling

The Keras/Tensorflow implementation of a sequential neural network was used to classify our news headlines. *LSTM*, or Long short-term memory, is a recurrent neural network proven effective on features that are ordered in a sequence. Since the meaning of a word depends on the ones that preceded it, *LSTM* is a particularly good tool for text classification tasks. The model was trained on a binary classification task which combined the labels *Pants-Fire, False, Barely-True,* and *Half-True* together in to a single *False* label; and combined *Mostly-True* and *True* into a single *True* label. A model was also trained for multilevel classification, but the results were very poor and hence will not be included in this report. We assume that the poor results in the multilevel setting were due to our limited sample size, but this is something that we hope to explore further in future work to arrive at a definitive conclusion.

Model Training – The *LSTM* model was trained with a batch size of 256, over 3 epochs, and using the 100-dimension *word2vecWiki* and *GloVeWiki* vectors. In addition to the embedding layer, the model uses two *LSTM* layers with 128 units and then 64 units; and two *Dense* layers which includes the output later which unitizes a sigmoid activation function. See *figure 2* for the complete set of model hyperparameters.

Model Analysis – The model was assessed using the overall accuracy and loss across all epochs. Precision, recall, and F1-score were also generated.

MODEL LAYERS	LAYER PARAMETERS
EMBEDDING LAYER	dimensions = 100
1ST LSTM LAYER	units = 128 return_sequences = True recurrent_dropout = 0.25 dropout = 0.2
2ND LSTM LAYER	Units = 64 recurrent_dropout = 0.1 dropout = 0.1
1ST DENSE LAYER	units = 32

	activation = relu
2ND DENSE LAYER (OUTPUT)	units = 2 activation = softmax
OTHER TRAINING PARAMETERS	network_type = sequential loss = sparse_categorical_crossentropy optimizer = rmsprop optimization metric = accuracy batch_size = 256
VALIDATION PARAMETERS	learning rate reduction method = reduceLROnPlateau monitor = val_accuracy patience = 2

Figure 3. Parameters for the classification Neural Network implemented with Keras/Tensorflow

Results

GloVe and *word2vec* yield different outputs from the similarity task. Take the word "president" as an example, the results of *GloVe* yields "Obama", "Barack", and "administration". Judged by human brain, these words are quite specific to the given word. In contrast, the output from Word2Vec is reasonable but those words could also be fit into another scenario or another paragraph.

```
glove.most_similar('president')

     [('Obama', 0.9744995769918297),
      ('President', 0.9737992560839709),
      ('Barack', 0.9727454789390149),
      ('administration', 0.9711692727036643)]
```

```
W2V.wv.most_similar('president')

     [('senator', 0.9984863996505737),
      ('became', 0.9983881711959839),
      ('declared', 0.9983291625976562),
      ('modern', 0.9982132911682129),
      ('doubled', 0.998191237449646),
```

Figure 4. Similar words generated by from the GloVeFN and word2vecFN. Results from glove show words that co-occur together while word2vec results show words that would appear in the target word's context.

The accuracy for both the *word2vecWiki* and the *GloVeWiki* trained models were both right around .645 with *word2vec* edging out a negligibly higher accuracy. An analysis of the loss of the neural network shows the loss begin to plateau after one epoch at around .652

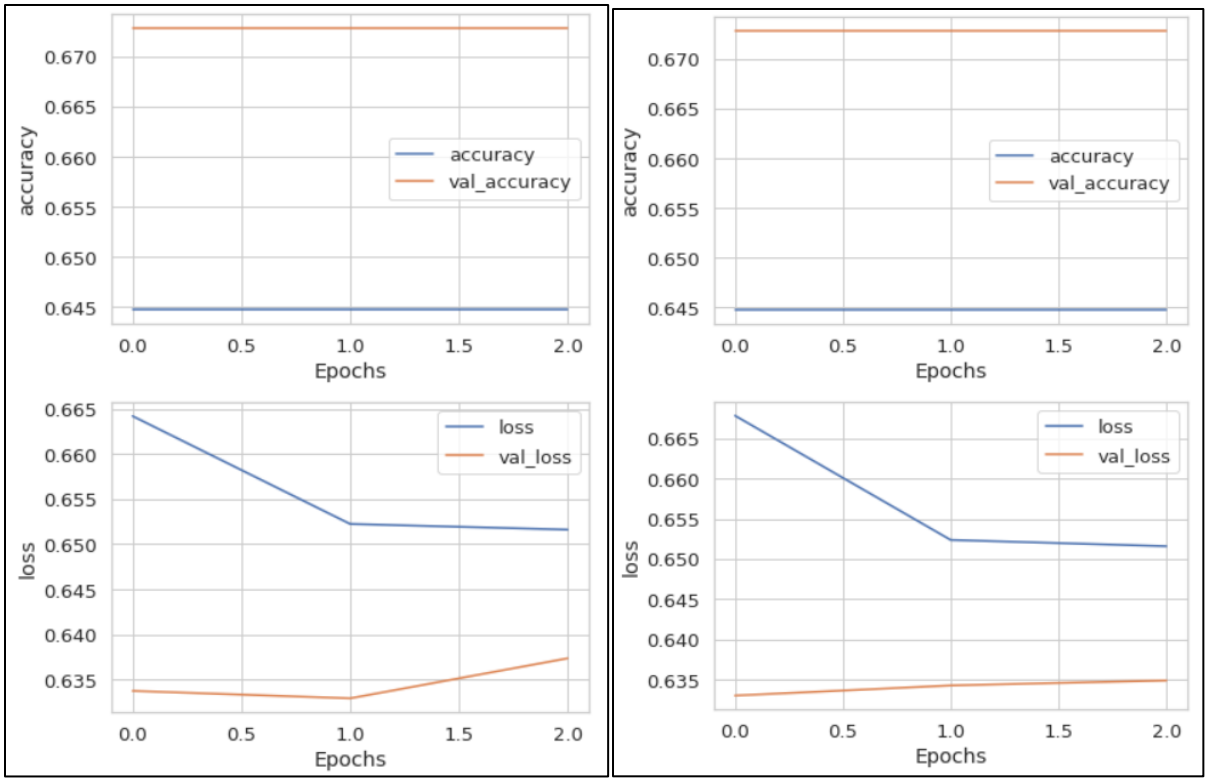

Figure 5. Accuracy and loss plots for the GloVe LSTM model (left) and word2vec LSTM model (right). The two plots are nearly identical and show both the accuracy and loss plateaung after 1 epoch at roughly 0.673 and 0.652 respectively.

	precision	recall	f1-score	support
0.0	0.65	1.00	0.78	818
1.0	1.00	0.00	0.00	449
accuracy			0.65	1267
macro avg	0.82	0.50	0.39	1267
weighted avg	0.77	0.65	0.51	1267

	precision	recall	f1-score	support
0.0	0.65	1.00	0.78	818
1.0	0.00	0.00	0.00	449
accuracy			0.65	1267
macro avg	0.32	0.50	0.39	1267
weighted avg	0.42	0.65	0.51	1267

Figure 6. Evaluation metrics for the GloVe LSTM model (left) and word2vec LSTM model (right). Both models performed identically with a max accuracy of 0.65 and a max F1-score of 0.78.

Conclusions

Both the *LSTM* models trained with *word2vec* and *GloVe* embeddings performed nearly identical on classifying fake news. While the differences in how the two methods embed words is clearly visible when observing with the similar word generation task, it does not appear that these differences have a significant effect on the performance of an *LSTM* model in a binary classification task.

One key takeaway from this experiment are that a binary classification task may not be the clearest way to expose the strengths and weaknesses of *word2vec* and *GloVe*. Other tasks may need to be more closely related to similarity tasks, like analogy tasks – which build off word similarities.

Future investigations into how *word2vec* and *GloVe* vectors effect the performance of prediction models would also expand the chosen classification model beyond *LSTM* to other methods like random forest or even a simpler logistic regression. Also, as mentioned before, to make this problem solvable in our context, the original four multiclass labels were converted into binary levels. If a more robust dataset is

found it would be interesting to explore if *word2vec* and *GloVe* vectors show varying effects on the performance of a multi-class classifier. To constrain our evaluation to the word vectors alone, the *LSTM* model used in our tests was trained using the baseline parameters only. However, to further question which embedding to choose, future work should evaluate the most optimal performance of each model, which would involve optimizing each model's parameters separately. Not only might this uncover that a specific model could be optimized to a greater performance, but it would also allow for analysis of how the different vectors effect model performance through by analyzing which hyperparameters needed to be changed to reach that model's optimal performance.

References

[1] Antoniak, M., & Mimno, D. (2018). Evaluating the Stability of Embedding-based Word Similarities. *Transactions of the Association for Computational Linguistics, 6*, 107-119. doi:10.1162/tacl_a_00008

[2] Berardi, G., Esuli, A., & Marcheggiani, D. (2015). Word Embeddings Go to Italy: A Comparison of Models and Training Datasets. IIR.

[3] Cothenet, C. (2020, June 02). Short technical information about Word2Vec, GloVe and Fasttext. Retrieved November 17, 2020, from https://towardsdatascience.com/short-technical-information-about-word2vec-glove-and-fasttext-d38e4f529ca8?gi=e7e766c118ed

[4] Kula, M. (2014, November). Maciejkula/glove-python. Retrieved November 1, 2020, from https://github.com/maciejkula/glove-python

[5] Lai, S., Liu, K., He, S., & Zhao, J. (2016, May 25). How to Generate a Good Word Embedding. *IEEE Intelligent Systems, 31*(6), 5-14. doi:10.1109/mis.2016.45

[6] vLi, H. (2018). Comparison of Word Embeddings and Sentence Encodings as Generalized Representations for Crisis Tweet Classification Tasks.

[7] 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

[8] Rong, X. (2014). word2vec Parameter Learning Explained. ArXiv, abs/1411.2738.

[9] Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. J. (2019, July 8). Evaluating word embedding models: Methods and experimental results. *APSIPA Transactions on Signal and Information Processing, 8*. doi:10.1017/atsip.2019.12

YouTube videos:

[10] Understanding Word2Vec
https://www.youtube.com/watch?v=QyrUentbkvw

[11] Lecture 2 | Word Vector Representations: word2vec
https://www.youtube.com/watch?v=ERibwqs9p38

Summary of Team Roles

For each team member, provide a clear description of their role and contribution.

Wilson Huang

- Figured out how to use GloVe package to train with our own data and do similarity tasks
- Worked together with Han to
  - Converted word vectors into embedding matrix
  - Loaded pretrained word vectors
  - Debugged LSTM model
- Made the introduction part of the video presentation as a person having experienced fake news

Tommy Tseng

- Understood and explained the theory for GloVe
- Responsible for background section of the final report
- Participated in scriptwriting and video-recording for the video presentation

Malcolm Smith Fraser

- Organized, drafted, and edited report document
- Planned and edited the video project
- Background research

Han Gong

- Loaded and cleaned the training dataset for model fitting
- Work with Wilson to transform the pre-trained word embeddings and train the LSTM classification model
- Explain and present the result of embedding task in the group video

Zoe Zhu

- Conducted literature review on word embedding methods
- Responsible for introduction and word2vec section in the final report
- Wrote the script for video presentation (primarily word embeddings and word2vec section) and participated in videorecording