1 INTRODUCTION

This report documents the process and results of training the CBOW word2vec model on a corpus of article-text related to farmer's protests obtained from Times of India (TOI) archives over 2-years (1/1/2020-1/1/2022). First, we plot the initial embeddings (Google News Vectors), followed by training the model and ploting the trained embeddings with the aim to subjectively compare the two and observe any discernible trends in movements of the word vectors.

2 METHOD

The article-text obtained was present as a CSV, with each row pertaining to one article. First, we pre-process this data to make it usable in obtaining word-level vector embeddings.

Next, we use this CSV to prepare the initial embeddings for our corupus, reducing them down from 300 to 2 dimensions, and then plot them onto 2-d scatter-plots.

Finally, we train our word2vec CBOW model (using the gensim library) and plot the trained embeddings onto new scatter-plots.

2.1 Pre-Processing

Pre-processing has a separate codefile from the rest of the analysis, and is included in the GitHub repository as *'preprocessing.ipynb'*.

1. We strip the text of all non-alphabets, including punctuation, numbers, and non-English characters (like â, œ, €) which slipped into the data during the scraping phase.
2. We convert all words to be in lowercase.
3. We remove all stop-words using Gensim's parsing.preprocessing module.
4. We lemmatize all the text, converting each word into its canonical form via the spaCy library.

Each row's article text is pre-processed and stored in an adjacent cell in the same row, as a singular string, with each word separated by a whitespace.

2.2 Initial Embeddings

All the code from here-on is in the second codefile, included in the GitHub repository as *word2vec.ipynb*.

First, we initialize a list in our environment, which includes the entire corpus, each word as an element. There are a total of 7,19,468 elements here. We then get a list of all the unique words in our corpus, which boil down to just 41,131. Out of these, the number of words in the Google News Vectors' Vocabulary is 16,151.

Net, we load in the Google New vectors. We initialize the embeddings for the words that are present in the Google News Vocabulary to those provided. For out-of-vocabulary words (OOVs), we don't have a pre-provided embeddings and need some placeholder vectors to represent them. So, we the initial embeddings for these to be random vectors with values drawn from a normal distribution. The mean and standard deviation for this distribution are set to the values as calculated by taking the embeddings of all our in-vocabulary vectors.

We want a scatter-plot of these embeddings, which entails reduction from 300-dimensional vectors to 2-dimensional vectors. To achieve this, we separately use PCA and tSNE reduction on our

embeddings.

Plotting all 40,000 word embeddings makes the scatter-plot way too dense to read, so instead we plot only the top-n words by frequency. For the results shown below, n is set to 1,000. So, each plot we get contains the 1,000 most words in our corpus.

2.3 Trained Embeddings

Finally, we train the CBOW and Skip-gram models on our corpus to obtain trained-embeddings for all words, and plot both the PCA and tSNE plots for both these trained emebddings.

The training generated embeddings for each vector which occured at least twice in the corpus, the window-size was set to 2, and the number of epochs trained on was 10.

Again, the plots were only generated for the 1,000 most frequent words in the corpus. The results are shown as follows.

3 RESULTS

All the plots shown here are interactive plots (created using the plotly library) and can be inspected further via the GitHub respository.

3.1 PCA

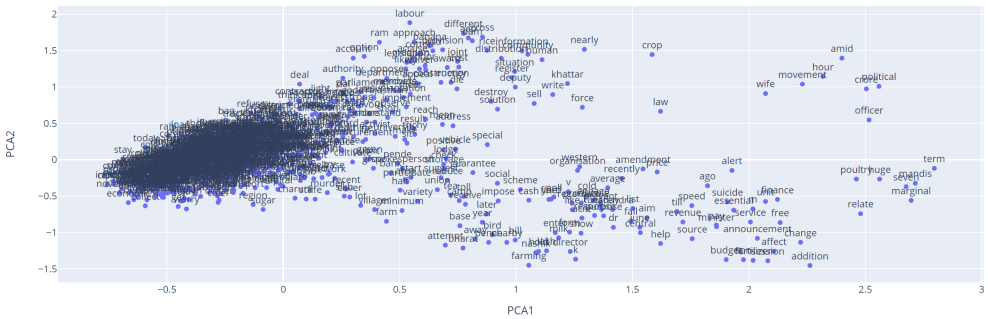

Fig. 1. Pre-Training PCA

Expectedly, the embeddings before training are pretty random since most of the words in our corpus were OOVs. Most of the points are present in a single cluster towards the left, possibly due to the randomized vectors assigned to OOVs around the mean and standard deviation of the in-vocabulary words. While we can make sense of some word-groups (social, scheme, guarantee, cash at (1, -0.5)), a-lot of the groupings seem arbitrary.

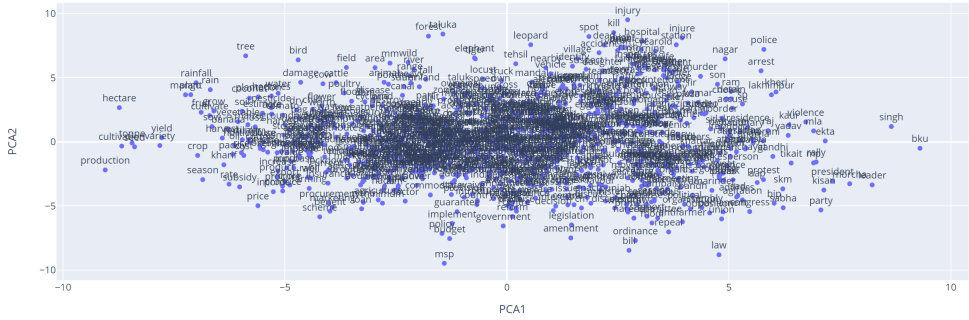

Fig. 2. CBOW: PCA

These results make a-lot more sense. For each point, the neighbouring points represent some word which could be reasonably understood as relevant given the context of the farmer's protest. For instance,

1. leader, kisan, president, morcha towards the right (7.5, -3)
2. seed, yield, tonne, production towards the left (-8, 0)

However, the central cluster still encapsulates majority of the data points, with no discernible pattern.

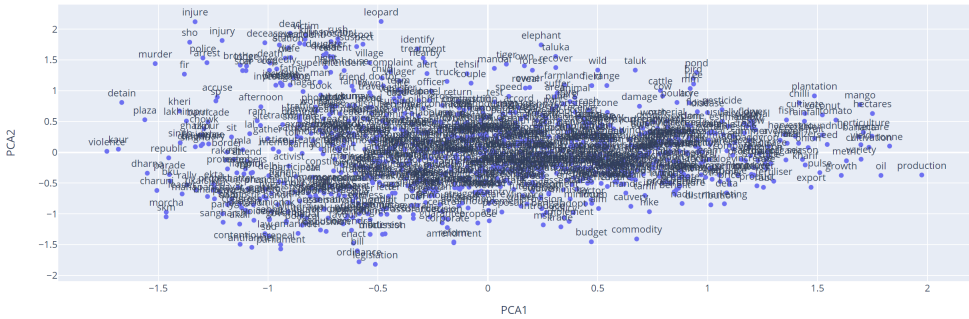

Fig. 3. Skip-gram: PCA

We get similar results for the skip-gram PCA run as well.

3.2 tSNE

As before, before training the results are random due to the divide between in-vocbulary words and OOV words. However, the tSNE results still are more distributed, with alot more clusters visible than was the case with PCA before training.

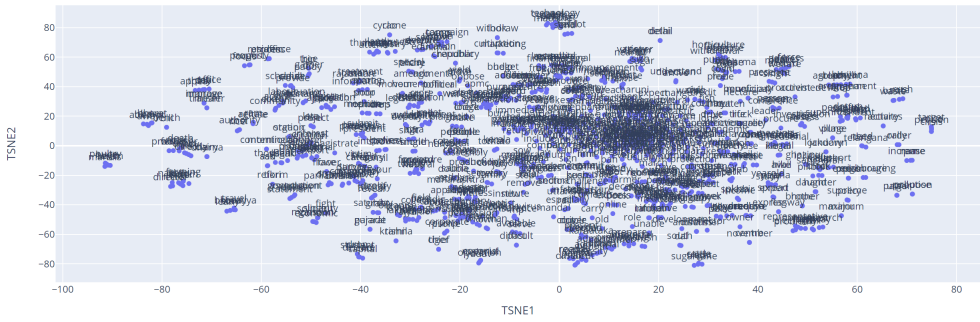

Fig. 4. Pre-Training tSNE

Fig. 5. CBOW: tSNE

These results are again much more distributed than their PCA counter-part and seem more sensible in their clusters. Some particular clusters of interest can be

1. worry, destroy, coronavirus, lockdown (-17, 10)
2. suicide, debt, marginal, small (-30, -12)

Each cluster captures words very apparently related to each other given the context of farmer's protests.

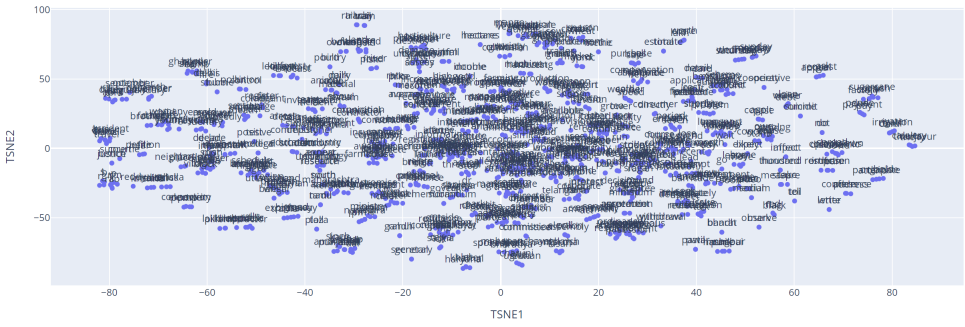

Fig. 6. Skip-gram: tSNE

These results are similar to the CBOW run-results but the clusters towards the center are a little more concentrated. Moreover, on subjective judgement, each cluster does not contain words as handedly related to each other in the given context as it did in the CBOW-run.

Without some form of computational evaluation, it's hard to do further analysis on the obtain scatter-plots. However, the CBOW run for the tSNE-reduced embeddings seems to produce clusters which align best with subjective interpretations than other runs.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009