**Name**: Chia Bing Xuan

**Matriculation Number**: A0259419R

**Title**: DSA4213 Assignment 1 – Word Embedding Exploration

1. **Introduction**

In this assignment, I conducted experiments on various word embedding algorithms. I considered three techniques:

* Skip-Gram with Negative Sampling (SGNS)
* Shifted Positive Pointwise Mutual Information - Singular Value Decomposition (SPPMI-SVD)
* GloVe

I implemented each of the aforementioned techniques, utilising both qualitative and quantitative metrics to evaluate and compare their performances on a selected corpus.

1. **Explanation of Algorithms**
   1. **SGNS**

The skip-gram model is a specific model variant of the Word2Vec family. The fundamental working principle of skip-gram models lies in distributional semantics – the idea that the meaning of a word can be deduced from its neighbouring words, within a preset context window. Ultimately, our goal is to ensure that words appearing in similar contexts will have similar vector embeddings.

Suppose we have a large corpus of length . We also set the size of the context window,. For each position , we consider the word at the centre of the window, , as well as the surrounding words (context words) within the window,. For skip-gram models in particular, we aim to maximise the conditional probability of each context word, given the centre word. In other words, if is the concatenation of all word embeddings to be obtained, we want to maximise

which is equivalent to minimising the objective function

Note that each word will either be a context word or a centre word, and this role changes as the window shifts. Thus, we associate each word with two embeddings – for when it is a context word and for when it is a centre word. For a centre word and a context word , the vanilla skip-gram algorithm defines the conditional probability using the softmax function:

where is the set of unique words in the corpus. However, this is computationally expensive to determine, as we would have to sum over all the words in to do so. In my implementation of skip-gram Word2Vec, I made use of negative sampling. For each centre word , this involves selecting words that are outside of the context window, but still within the vocabulary. In essence, we aim to maximise the similarities between actual context words and , whilst minimising the similarities between negative samples and . Hence, instead of minimising in the definition of , we minimise

where is the sigmoid function and is the unigram distribution present in the distribution , from which the negative words are sampled.

Optimisation techniques can then be used to find that corresponds to minimum loss.

* 1. **SPPMI-SVD**

The SPPMI-SVD algorithm makes use of the pointwise mutual information (PMI) metric, in the context of word co-occurrences.

Suppose we have a large corpus of many documents. We also set the size of the context window,, which is to be shifted throughout each document in the corpus. For a word and a context , co-occurs with if appears in the context window centred at . In this way, we can construct a co-occurrence matrix, in which the -th entry corresponds to the number of co-occurrences of word with context , throughout the corpus. With this, each word in the vocabulary corresponds to a single row in the co-occurrence matrix, which is a vector that encodes information about how often other words appear near that word.

Then the PMI of each pair is defined by

where is the sum of all the elements of the co-occurrence matrix and is the number of co-occurrences of word with context .and are respectively the number of times word and word are the centre words.

The shifted positive PMI metric is then defined by

where is a hyperparameter that dictates some shift applied to the PMI value. The SPPMI function can then be applied element-wise to the co-occurrence matrix, obtaining a SPPMI matrix . Singular Value Decomposition can then be applied to for dimensionality reduction:

Keeping the largest singular values, the reduced word embeddings are (based on my implementation):

where corresponds to the leftmost columns of and is the diagonal matrix containing the largest singular values.

* 1. **GloVe**

The intuition for GloVe arises from how a word can be distinguished from another word by considering the ratios of conditional probabilities, with respect to some probe words. For instance, we consider the ratios and for probe words and . As an example, in the case of

we might have and , which would allow us to differentiate between the two target words “ice” and “steam” in that way.

With this in mind, we can make use of the log-bilinear model to define the GloVe word embeddings **,** and, ensuring that the difference can reproduce the ratio of conditional probabilities, thereby encoding meaning components:

Note that the conditional probability(and similarly) can be expressed in terms of co-occurrences, using the co-occurrence matrix :

where is the -th entry of .

After derivation, the loss function for GloVe embeddings is given as

with weighting function . We associate each word with two embeddings – for when it is a target word and for when it is a probe word.

1. **Methodology**
   1. **Selection of Corpus and Data Processing**

For this assignment, I investigated the effectiveness of the aforementioned embedding techniques, on a corpus of IMDB movie reviews offered by the NLTK library. This dataset consists of 1000 positive reviews and 1000 negative reviews compiled by Bo Pang and Lillian Lee. Note that each review can be loaded as a list of raw tokens, along with its associated sentiment label (either positive or negative). Throughout the workflow, a seed of 42 was set for reproducibility.

A round of data pre-processing was carried out on this dataset:

1. Convert the class labels “pos” and “neg” into their corresponding integer labels (1 and 0 respectively)
2. For each review (list of raw tokens):
   1. Convert the tokens to lowercase
   2. Remove tokens that do not consist entirely of alphabets (eg. punctuation marks and numerals)
   3. Remove stop words (eg. function words such as “a”, “the”, “it”, etc.)
   4. Lemmatise the remaining tokens (convert to root word / base form)

The most common words in the processed dataset are the following:

|  |  |
| --- | --- |
| **Word** | **Count** |
| film | 11053 |
| movie | 6977 |
| one | 6028 |
| character | 3879 |
| like | 3789 |
| time | 2979 |
| get | 2814 |
| scene | 2671 |
| make | 2634 |
| even | 2568 |

* 1. **Experimentation**

I started by performing a train-test split of 80%-20% on the processed dataset, before using the train set for my embedding exploration.

Firstly, I applied each embedding technique on the train set using the following set of parameters. Note that the set of parameters used are consistent across the techniques (the shift is analogous to the number of negative samples in SGNS).

|  |  |
| --- | --- |
| **Embedding Technique** | **Parameters** |
| SGNS | Number of epochs: 20  Minimum frequency of words to consider: 1  Vector size: 50  Window size: 3  Number of negative samples: 5 |
| SPPMI-SVD | Minimum frequency of words to consider: 1  Vector size: 50  Window size: 3  Shift (): 5 |
| GloVe | Number of epochs: 20  Minimum frequency of words to consider: 1  Vector size: 50  Window size: 3  Number of negative samples: 5 |

I subsequently carried out hyperparameter tuning by iterating through various sets of hyperparameters, attempting to find the one that leads to optimal performance for each embedding technique. To quantify performance, I made use of the Spearman correlation coefficient. This was calculated between human-labelled similarity scores in the WordSim-353 dataset, as well as the corresponding cosine similarity values produced by each fitted model.

The following are the hyperparameter values that were experimented with, which were consistent across the three techniques:

|  |  |
| --- | --- |
| **Parameter** | **Values Considered** |
| Vector size | 50, 100, 150 |
| Window size | 3, 5, 10 |
| Number of negative samples (SGNS, GloVe) / Shift () (SPPMI-SVD) | 3, 5, 10 |

For each embedding technique, I then compared the quality of the new embeddings with the original embeddings. This was done qualitatively, by analysing the nearest neighbours of a list of selected words. If hyperparameter tuning did not lead to any significant improvement in the quality of the nearest neighbours, the original embeddings were chosen – otherwise, I selected the embeddings produced from hyperparameter tuning.

* 1. **Sentiment Analysis**

After selecting a set of word embeddings for each technique, I conducted sentiment analysis on the movie reviews dataset, as part of a downstream task. Using a set of word embeddings , it is possible to convert each review in the dataset (both train and test) to a feature vector. The steps taken were as follows:

1. For each review:
   1. Ignore words that are not present in the vocabulary of
   2. Convert all remaining words to their corresponding vector embeddings
   3. Determine the mean of all these vector embeddings element-wise to obtain a feature vector for the review
2. Using the embeddings and class labels in the train set, train a classifier to identify the sentiment of a given review.
   1. Models used: Random Forests, Support Vector Machines (SVMs), XGBoost
3. Evaluate the classifier on the embeddings and class labels in the test set
4. **Evaluation**
   1. **SGNS**
      1. **Fixed Set of Parameters**

A Word2Vec model was fitted on the train set with the gensim library in Python. A fixed set of parameters was used:

* Number of epochs: 20
* Minimum frequency of words to consider: 1
* Vector size: 50
* Window size: 3
* Number of negative samples: 5

Upon obtaining the word embeddings, principal component analysis (PCA) was used to reduce them to vectors of size 2. The corresponding PCA scatter plot of the 50 most common words in the corpus is shown below.

A diagram of a number of dots

AI-generated content may be incorrect.

It can be seen that the SGNS model was able to separate out words of certain categories. For example, words related to individual cast members – such as “character”, “actor”, “role” and “performance” are significantly further away from the other word clusters. Moreover, words describing the segments of a movie (in part or whole) – such as “movie”, “film”, “scene” and “plot” are located near one another, although “story” is positioned further away. Words describing quantity – such as “little” and “many” – are also within close proximity of one another. However, this may not be ideal since “little” and “many” are opposite in meaning. Generally, the embeddings are part of one large cluster, with smaller clusters located further away from the main one. Some of these smaller clusters – such as the one containing “new”, “go” and “take” – are less reasonable, given that these words do not have similar meanings.

I also analysed the nearest neighbours for a list of ten common words (“film”, “like”, “good”, “time”, “story”, “character”, “life” and “scene”), checking if they are valid or otherwise.

|  |  |  |
| --- | --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** | **General Semantic Meaning** |
| film | movie, unsatisfactory, expanded, mant, emphatically |  |
| like | synch, embarrassed, sake, kinship, xerox |  |
| good | decent, eas, great, terrible, paled | Adjectives that evaluate quality |
| time | booted, waaaay, heartbreaker, percent, scarce |  |
| story | parallel, storyline, overlap, analogy, linear | Describes the plot of a movie |
| character | personality, tangential, role, incorrectly, attachment | Describes the traits of individuals in a movie (eg. personality, role) |
| life | comfort, harmony, miracle, live, sacrificing | Intangible aspects of life |
| scene | moment, sequence, confrontation, straw, gunfight | Various movie scenes |

As detailed in the “General Semantic Meaning” column, the nearest neighbours of some words are logical. For example, the word most similar to “film” to “movie”, as expected. The word “sacrificing” is similar to “life”, since they are likely to appear in similar contexts, such as in the phrase “sacrificing [one’s] life”. The word “good” is most similar to other words with positive connotations, such as “decent” and “great”. While it makes less sense for “good” to be similar to “terrible”, this can be explained by how these two words have the same function – both are used to evaluate the quality of something or someone. However, the closest neighbours for other words are less reasonable, such as the words “like” and “time”.

* + 1. **Hyperparameter Tuning**

After hyperparameter tuning with Spearman correlation coefficient (using WordSim-353 dataset), the optimal parameters of vector size = 150, context window = 10 and negative samples = 5 were obtained. The results are shown below.

A white and blue chart with black text

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** |
| film | movie, rejuvenates, godforsaken, unconventionally, horor |
| like | ewwwww, unflushed, interferred, glisten, cagney |
| good | expended, commensurate, imaganitive, bregman, faulted |
| time | wayyy, bullsh\*tting, rewound, buddying, lamanna |
| story | ascribe, comprehendably, brining, unsurprising, tangentially |
| character | tangential, logistical, recaptured, unfetching, thorougly |
| life | fullest, bottlecap, predetermined, overtaken, touchingly |
| scene | impart, storyboarded, choppily, humping, unusable |

The nearest neighbours here are generally less logical. From this qualitative analysis, the performance of the word embeddings appears to have worsened after hyperparameter tuning. Thus, the original embeddings were chosen for SGNS.

* 1. **SPPMI-SVD**
     1. **Fixed Set of Parameters**

SPPMI-SVD was conducted on the train set. A fixed set of parameters was used:

* Minimum frequency of words to consider: 1
* Vector size: 50
* Window size: 3
* Shift (): 5

The results are shown below.

A diagram of a number of dots

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** |
| film | interferred, aspired, script, movie, look |
| like | come, think, watching, get, see |
| good | time, bad, much, little, make |
| time | much, even, really, better, could |
| story | time, little, made, interesting, give |
| character | never, even, good, interesting, really |
| life | story, come, actually, much, even |
| scene | much, never, time, audience, enough |

Note that the nearest neighbours for SPPMI-SVD are less reasonable. For example, “time”, “little” and “made” do not have much in common with the word “story”. Hence, I looked to improve this performance through hyperparameter tuning.

* + 1. **Hyperparameter Tuning**

After hyperparameter tuning with Spearman correlation coefficient, the optimal parameters of vector size = 150, context window = 3 and shift = 5 were obtained. The results are shown below.

A white and black graph

AI-generated content may be incorrect.

The positions of the SPPMI-SVD word embeddings are much more varied (over a larger area), compared to that of SGNS. Note that words describing the segments of a movie (in part or whole) – such as “movie”, “film” and “plot” are correctly located near one another, much like in SGNS. Words related to individual cast members – such as “actor”, “star”, “role” and “performance” are also detached from the main word cluster. For SPPMI-SVD, the words that are positioned outside of the main cluster (eg. “play”, “director”, “role”, “man”, “love” and “performance”) were also singled out in SGNS.

|  |  |  |
| --- | --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** | **General Semantic Meaning** |
| film | movie, made, many, could, much |  |
| like | one, really, know, look, even |  |
| good | well, one, really, time, much |  |
| time | see, one, know, much, even |  |
| story | plot, character, many, time, however | Describes the plot of a movie |
| character | much, even, however, good, one |  |
| life | people, even, real, one, time |  |
| scene | one, see, even, well, film |  |

Based on the nearest neighbour analysis, the quality of SPPMI-SVD embeddings has improved after hyperparameter tuning. For instance, the nearest neighbours for “story” now better reflect the plot of a movie (eg. “plot” and “character”). The nearest neighbour to “film” is now more reasonable as well (“movie” instead of “interferred”). Some pairs of similar words also make sense, as they tend to be used together (eg. “good time” and “real life”). This is characteristic of the SPPMI-SVD algorithm, which is based on pairwise word co-occurrences. However, the nearest neighbours generally do not reflect a common semantic meaning related to each test word, with the exception of “story”. This pales in comparison to SGNS, where the nearest neighbours of “good”, “story”, “character”, “life” and “scene” each collectively describe a general idea relevant to the word in question. Since there is greater value in gathering words which convey similar meanings (rather than merely identifying which words are used alongside one another), it can be concluded that the SPPMI-SVD algorithm performs worse than SGNS.

Regardless, since hyperparameter tuning has improved the quality of the SPPMI-SVD word embeddings, the new embeddings were chosen for SPPMI-SVD.

* 1. **GloVe**
     1. **Fixed Set of Parameters**

A GloVe model was fitted on the train set with the text2vec library in R. A fixed set of parameters was used:

* Number of epochs: 20
* Minimum frequency of words to consider: 1
* Vector size: 50
* Window size: 3
* Number of negative samples: 5

The results are shown below.

A graph with black dots

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** |
| film | movie, one, however, since, even |
| like | movie, look, kind, actually, even |
| good | bad, also, making, make, look |
| time | long, since, movie, one, much |
| story | plot, rather, tell, character, way |
| character | seems, main, also, rather, interesting |
| life | real, find, world, take, way |
| scene | sequence, one, moment, particularly, also |

Note that the nearest neighbours for GloVe have some room for improvement. For example, “movie” and “look” do not have much in common with the word “like”. Hence, I looked to enhance this performance through hyperparameter tuning.

* + 1. **Hyperparameter Tuning**

After hyperparameter tuning with Spearman correlation coefficient, the optimal parameters of vector size = 50, context window = 10 and negative samples = 10 were obtained. The results are shown below.

A diagram of a movie review

AI-generated content may be incorrect.

Note that the scatter plot after hyperparameter tuning contains more distinct clusters, which is an improvement. The GloVe embeddings of “actor”, “performance” and “role” – all of which pertain to individual cast members – are located further away from the main cluster, much like in SGNS and SPPMI-SVD. In addition, the words “movie”, “film” and “action” (as in “action movie”) are also singled out, positioned within close proximity of each other. Notably, words with positive connotations – such as “good”, “best”, “well” and “star” are also grouped more closely together, which is an enhancement from the previous two embedding techniques. Another new development is that “character”, “director” and “star” (as in “movie star”) are all located near each other – which can be explained by the fact that these words are related to the people involved in a movie. However, the pair of “man” and “love” is less reasonable, as these words are not very similar to each other from a semantic standpoint.

|  |  |  |
| --- | --- | --- |
| **Test Word** | **Nearest Neighbours (in descending order of cosine similarity)** | **General Semantic Meaning** |
| film | movie, one, many, made, even |  |
| like | one, movie, even, good, really |  |
| good | bad, even, like, also, really |  |
| time | one, first, long, two, however | Elements of time |
| story | plot, film, character, way, also | Describes the plot of a movie |
| character | also, main, one, story, interesting |  |
| life | real, world, people, find, come | Intangible aspects of life |
| scene | moment, sequence, another, one, also | Sections of a movie |

Based on the nearest neighbour analysis, the quality of GloVe embeddings has slightly improved after hyperparameter tuning. For instance, the word “good” is now one of the nearest neighbours for “like” (and vice versa), which better reflects the idea of a preference for something or someone. Like the previous techniques, GloVe correctly identifies “movie” as the most similar word to “film”. As mentioned previously, it is also somewhat reasonable for “bad” to be similar to “good”, despite them being polar opposites. Furthermore, like SGNS, certain test words have nearest neighbours that accurately reflect a collective semantic meaning. For example, “first” and “long” possess temporal elements (test word: “time”), while “sequence” and “moment” both describe segments of a movie (test word: “scene”). Some pairs of similar words are also valid, as they tend to be used together (eg. “main character”). This is unsurprising as the GloVe algorithm is based on pairwise word co-occurrences, like SPPMI-SVD. However, there are several cases where the nearest neighbours are less representative of the given word’s meaning. Examples include some of the neighbours for “film” – such as “one”, “many”, “made” and “even”.

Since hyperparameter tuning has improved the quality of the GloVe word embeddings, the new embeddings were chosen for GloVe.

* 1. **Sentiment Analysis**

Using the chosen word embeddings from SGNS (without hyperparameter tuning), SPPMI-SVD (with hyperparameter tuning) and GloVe (with hyperparameter tuning), I conducted sentiment analysis on the movie reviews dataset. This was done by training various scikit-learn classifiers on the train set, before evaluating their performances on the test set. The evaluation results, in terms of accuracy, are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RandomForestClassifier** | **SVM** | **XGBoost** |
| **SGNS** | 73.0 | **77.5** | 74.8 |
| **SPPMI-SVD** | 72.0 | **76.0** | 74.8 |
| **GloVe** | 66.0 | **70.0** | 65.5 |

For this sentiment classification task, SVMs consistently performs better across all the embedding techniques. Among all <embedding, classifier> pairs, SGNS with SVM attains the strongest performance, with an accuracy of 77.5%.

1. **Conclusion**

An investigation was conducted on three embedding techniques – SGNS, SPPMI-SVD and GloVe – using the NLTK movie reviews dataset. The embedding performance was evaluated in two ways. Firstly, I analysed the nearest neighbours of a list of test words. Furthermore, I also used the embeddings for a downstream sentiment analysis task (extrinsic word vector evaluation). In sum, SGNS is the algorithm that achieves the best performance on this corpus.

1. **Appendix**

The repository for this assignment can be found at <https://github.com/chiabingxuan/Word-Embeddings-Exploration>.

Attached below are the code snippets used.