ASSIGNMENT 2 NATURAL LANGUAGE UNDERSTANDING

SURVEY REPORT: WORD EMBEDDING

Elisabeth Putri – 20306250

Similarity between two things can happen in every aspect. For example, in animal world, tiger looks like cat in bigger version or fox looks like dog. Both of this similarity is supported by their context, in this case is they are animals. This context and similarity are also important in words. Words that occur in the same context will usually have meanings that close to each other. This research is developed since the synonym words are tended to occur in the same environment.

Because of this similarity, it can be applied in either business or even daily life. First, review classification either positive or negative regarding to a product. Or even to search for information in search engine, like google, there are several suggestions related to the words that have been typed. All of these are the application of Text Processing, which is processing the text to clustering words, do sentiment analysis, document classification and tagging (Francois Chaubard, Rohit Mundra, Richard Socher, 2016).

Text processing is also useful to consider decisions. It gives suggestion about similar movie which might be attractive to be watched. These suggestions depend on the movie that is usually seen, so the suggestion will not something that is far from the movie viewer expectation. Words have high relation to the texts; it is useful to generate documents using language model.

To begin the text processing, several steps should be done to clean up the data (corpus) which are going to be used (Singh, 2017). There are several options to do the text pre-processing, such as converting word into lowercase, removing stop words, removing punctuation, removing leading and trailing whitespaces, expanding contractions, removing special characters (number, emoji, etc.), removing HTML/ XML tags, replacing accented character, and correcting spelling error. In this process, it is also compulsory to divide the data into smaller chunk of sentence, even a chunk of words. The process of slicing the data into words is called tokenization.

Each of the words which is already tokenized somehow has its own relation between each other. This relation is built by a linguistic study of word meaning. The exact same word can be a lemma, which is root of another word. The word ‘sing’ is the lemma for word sing, sang, and sung. Their relation might be a synonym or antonym or even a similar context and relation. The common relation between words happens if they are in the same semantic field. Semantic field is a group of word which is related into the same domain, then building structural relation to each of them. For example, word ‘hospital’ as a domain, then other words which relate to it are doctors, nurse, surgeon, drugs, anaesthesia, patient, medical record (Daniel Jurafsky, James H. Martin, 2020). Some words also bring connotation meaning, either positive or negative. For example, the word ‘happy’ is relate to a positive connotation while the word ‘sadness’ is more relate to a negative connotation. This positive and negative word could lead to the classification of sentiment in that document.

Presenting the word processing in Natural Language Processing is using mathematical way. Each of the word is detected its meaning, then transposed into a vector which is usually called vector semantics. The idea is presenting the word in the multidimensional semantic space which is derived from the neighbours’ word distribution. This vector is also called embedding. This technique is quite

common to be used because this model can learn automatically without supervision. Sentiment is assigned by the classifier when similar word is detected.

The visualisation of these word vectors can be seen in Figure 1. The visualisation depicts that the vectors can capture general and useful information about the word and its information to each other. It is the reason why the vectors are useful as features for task prediction, like part-of-speech tagging or named entity recognition.

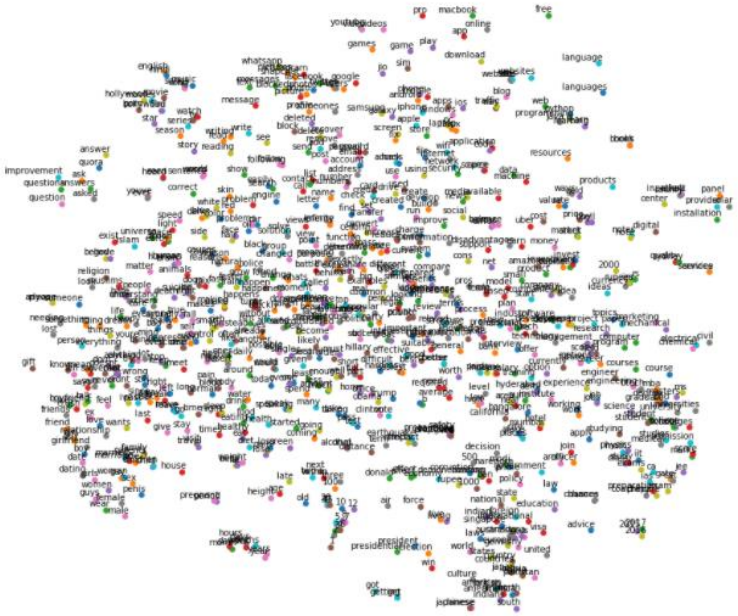

Figure 1. Vectors relation visualisation

The vector models are appended to form a matrix. One of the most common forms of matrix is the co-occurrence matrix, which is representing the amount of word occur either in a document or the relation between term and term. To build this matrix, each word (without repetition) is collected from the documents. This collection of word is called vocabulary. Each row of this matrix represents each word of vocabulary, while the columns represent document. The column of matrix can be changed into word to build a term-term matrix.

The co-occurrence matrix is less powerful association measurement. The developed technique is by giving weight into each vector. The first technique is tf-idf (term frequency-inverse document frequency). It is a combination between the number of terms appear in the document multiply by inverse of document frequency. The detail equation is shown below.

$$w_{t,d} = tf_{t,d} \times idf_t$$

Equation 1. Tf-idf equation

Another idea is PMI which stands for positive pointwise mutual information. This one is used in term-term matrix. It measures how often two words are observed together divided by the expectation of two words to co-occur with assumption that both independent. This ratio is an estimation of two words expected more to co-occur than we expect by chance.

$$PMI(w,c) = log_2 \frac{P(w,c)}{P(w)P(c)}$$

Equation 2. PMI equation

The similarity between two words can be assessed by cosine similarity. To calculate this, the vector for each word is needed. These vectors should contain same dimensionality. The formula is shown below. Higher the cosine similarity result, the more those two words are closer to each other. Each of the matrix above, either the co-occurrence matrix, tfidf matrix, or PMI matrix, the relation between two words is still calculated by cosine similarity as below. From those matrices, we gain vectors for each words.

$$cosine(v, w) = \frac{v \cdot w}{|v||w|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

Equation 3. Cosine similarity equation

To decide the similarity between two documents, tfidf matrix can be used to decide whether two documents are similar. The document representation is taken by the word vector in each document, then compute the centroid of the vectors. The application for tfidf matrix and PPMI model is computing word similarity. Several application tasks are finding the word paraphrase, tracking word meaning changes, or discover word's meaning from different corpora automatically.

Another matrix which is used for analysis is embeddings, the short dense vectors. The value in this vector is a real-valued, not zeros, and the dimension is just 50-1000. To compute the embeddings, it uses the skip-gram with negative sampling (SGNS) which is contained in the software package called word2vec. This embedding technique is kind of static embedding which learn one fixed embedding for each word in the vocabulary. The idea behind this is training a classifier on a binary prediction. Although this method is using neural network, it just uses the next word as signal to be used as learning embedding representation for each word, then predict it. It is much simpler because this word2vec use binary classification and using logistic regression (Daniel Jurafsky, James H. Martin, 2020).

Skip-gram technique emphasizes the probabilistic classifier, given a test target word w and the context. Thus, it could assign the probability of similarity between the context window and the targeted word. The probability calculation depends on application of logistic (sigmoid) function to the dot product of the embeddings of the target word with each context word. To depict this algorithm, the scheme is shown below. This skip-gram is better to be done by using large dataset because it keeps doing observation by using target word and its context. The steps of building skip-gram technique will be described below.

Assumed W_i is the weight matrix between input layer and hidden layer of size [V*N]; W_2 is the weight matrix between hidden layer and output layer of size [N*V]. The skip-gram steps are:

1. Generate the one-hot encoding vectors for the input which have size [1*V]
2. Multiply the vector in number one by matrix W_i to get the embedded word with size [1*N].
3. Calculate the hidden layer output by multiply the result in number 2 with matrix W_j. (the obtained result is called z(i)).
4. Turn the scores into probabilities by y(i) = softmax(z(i)).
5. Compared the generated probability in number 4 with the true probability, y(i).

6. The error between output and target is calculated and propagated back to re-adjust the weights.

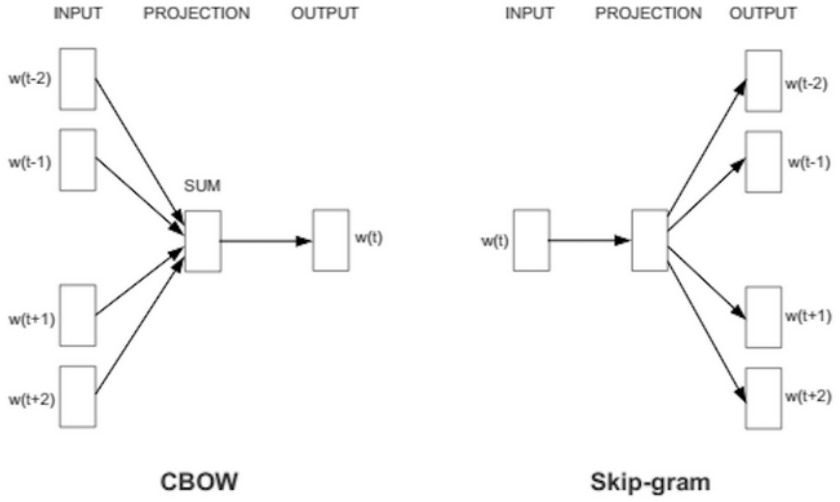

Figure 2. CBOW and Skip-gram scheme

In the figure 1, there is the inverse technique of skip-gram, which is called Continuous Bag-of-Word (CBOW). Both techniques are classified as prediction method. The CBOW technique will predict the target word from the surrounding context. It is suggested using in the smaller dataset as this technique using the whole context as one observation.

Assumed W_i is the weight matrix between input layer and hidden layer of size [V*N]; W_2 is the weight matrix between hidden layer and output layer of size [N*V]. The skip-gram steps are:

1. Generate the one-hot encoding vectors for the input context of size m. We have C(=2m) for the one-hot vector size [1*V]. Thus, the input layer size is [C*V].
2. Multiply the vector in number one by matrix W_i to get the embedded word with size [1*N].
3. Take the mean of 2m[1*N] vectors.
4. Calculate the hidden layer output by multiply the result in number 3 with matrix W_j. (the obtained result is called z).
5. Turn the scores into probabilities by y = softmax(z).
6. Compared the generated probability in number 4 with the true probability, y(i).
7. The error between output and target is calculated and propagated back to re-adjust the weights.

The word2vector itself is taking the large input vector, compressed into smaller dense vector then the neural network will process it to be a probabilities of target words as output. It can not use the string word directly, thus, we must process the data into the one-hot vectors. The vector will have same length as the vocabulary, then filling the vector with zeros except the index which represent the word. The word index will be filled with number '1'. Because it is using the neural network, this method is computed using hidden layer. This technique only uses one hidden layer to operate the data.

The idea of one-hot encoding is like this. When we are going to create the word 'python' as one-hot representation, we need to have a look on the vocabulary. In this case, the vocabulary contains five words, which are nlp, python, word, ruby, and one-hot. The word python contains number 1 while the rest of the vocabulary get zero.

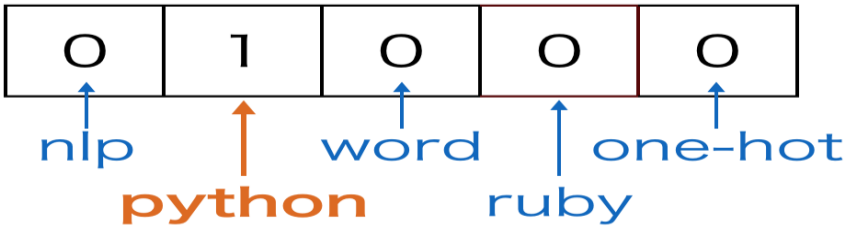

Figure 3. One-hot encoding vocabulary representation

The same rule of one-hot encoding happens for every word in the vocabulary. The word that is vectorized has value 1, and the rest are zero. For comparison, the figure 3 below visualises the difference between vector for python and vector for ruby.

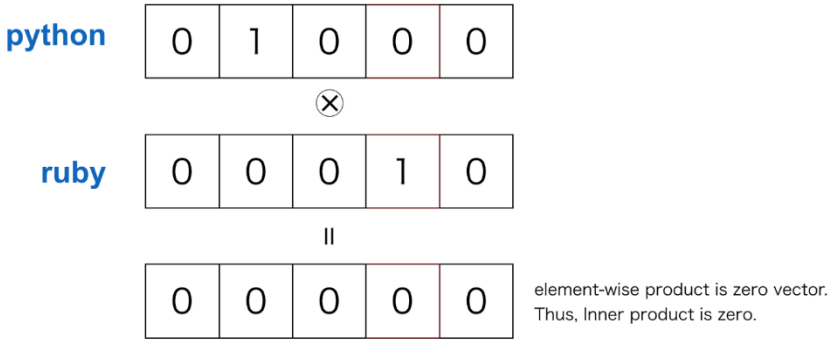

Figure 4. One-hot encoding comparison for each word in vocabulary

In the one-hot encoding technique, there are two disadvantages. It is not possible to get the meaningful result by arithmetic in between vectors because each of the one value in each vector is in the different place. Calculating similarity between two words becomes all zero. Because all the word in vocabulary is mentioned in the vector with almost all has zero value, the vector dimension becomes high. As the increasing number of vocabulary, the number of dimension is also increasing.

Word2vector can give similar output when the different input words have similar context by learning similar word vectors. This technique is powerful as it captures multiple different degrees of similarity between words by using vector arithmetic. It is trained by hierarchical softmax and/ or negative sampling. The hierarchical softmax uses Huffman tree to reduce calculation. This technique is basically approximation calculation the conditional log-likelihood that the model is going to maximise. The hierarchical softmax is recommended to work with the infrequent word, but it is not useful in high number of epochs. The second training for word2vector is negative sampling. This technique is reducing the computation by using negative environment surround the target word. It ignores the zero value in the one-hot label vector and then updating the target's weight. Negative samples are chosen by unigram distribution, which relate to the frequency. Higher the word frequency, those words are more likely to be chosen as negative samples.

The factors that affect in word2vec methods are the model choice (either CBOW or skip-gram), increasing the training set, increasing the number of vector dimensions, and increasing the window size of words. The key parameter of successful method is observed by the complexity and time processing. Overall, when the used number of words increase, followed by the increasing of dimensions, it brings the increasing overall accuracy. To maximize the calculation, the words which appear more than the threshold will be subsampled as it usually provides only little information. The common word pairs or phrases will be combined into single word to increase the productivity(Gilyadov, n.d.).

The skip-gram and CBOW have advantages and disadvantages. Both are using probabilistic, thus both of the methods are generally supposed performing superior deterministic method. Both does not a huge RAM as co-occurrence matrix because the matrix is not that big, thus these methods are low on memory. CBOW method is targeting one word by using many contexts of words, thus it is essentially smooth out over the distribution. It is like regularization; thus, it performs better in small amount of data. On the other hand, skip-gram is used to extract more information because the model is finer grained. Therefore, the embedding results are more accurate with large dataset as the large data is the best regularization. Generally, the combination between skip-gram and negative sub-sampling is outperform from other methods.

Another well-known model than word2vec is Global Vectors (GloVe). This method is a ount based model. This model does not use neural networks. The loss function is calculated by extract the difference between the word embeddings product and log probability of co-occurrence matrix. This technique also has an explicit global information embedded by default. It creates co-occurence matrix globally by estimate probability for a given word to be co-occur with other words. The large matrix contains the information on how frequently each word, which is the matrix row, is seen in some context. It is necessary to use large number of context as this will be a combinatorial in size. After that, it will factorize this matrix to yield a lower-dimensional matrix of word and features, which the row is the vector representation of each word. This global information makes GloVe works better ideally because it also allows a parallel implementation to train the data.

Glove takes the occurring of maximum probability of context word to predict the surrounding word by using dynamic logistic regression. It is able to capture global statistic, simultaneously captures the meaningful linear substructure. Because of its ability, GloVe is the global log-bilinear regression model for unsupervised learning in modelling natural language processing (Sciforce, n.d.).

References

Daniel Jurafsky, James H. Martin, 2020. *Speech and Language Processing.* U.S.: Pearson Education.

Francois Chaubard, Rohit Mundra, Richard Socher, 2016. *CS224d: Deep Learning for Natural Language Processing.* [Online]
Available at: https://cs224d.stanford.edu/

Gilyadov, J., n.d. *2021.* [Online]
Available at: https://israelg99.github.io/2017-03-23-Word2Vec-Explained/

Sciforce, n.d. *Towards Data Science.* [Online]
Available at: https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db89639

Singh, M., 2017. *Word Embedding.* [Online]
Available at: https://medium.com/data-science-group-iitr/word-embedding-2d05d270b285