

School of Engineering and Technology

Capstone 2 Final Report

Building the Constructions Activities Dictionary – Making machine understanding word choices in Construction (NLP)

By

Chua Wen Soong

(18032573)

Bachelor of Science (Honours) Information Systems (Business Analytics)

Supervisor: Prof. Lau Sian Lun

# 

# Table of Contents

[**Abstract**](#_b26pz9omjuf3) **3**

[**Acknowledgements**](#_ogmt2vab3pxq) **4**

[**1.0 Introduction**](#_nfhxiwio7gck) **5**

[**2.0 Literature Review**](#_lm8huhaa89ef) **6**

[**3.0 Methodology**](#_okx61j9bhp3p) **8**

[3.1 Overview](#_djv483bo5hbo) 8

[3.2 Construction Dataset](#_nljjzowlprcg) 8

[3.3 System Description](#_mvmxpg8u6bcx) 9

[3.4 Data Ingestion](#_4ns3t5c6brsh) 9

[3.5 Data Preprocessing of Text](#_u6obeskecm5d) 9

[3.6 Word Embedding/Vectorisation](#_2crc4ayxt4n1) 11

[3.6a Word2Vec](#_y24dsy3n6226) 11

[3.6b Word2vec with Gensim Phraser](#_l03y6wr13pnz) 11

[3.6c FastText](#_wob179sd1vs8) 11

[3.7 Evaluation](#_6lxbhfa6zruq) 12

[3.7a K-means Clustering](#_t0ggykkihhmc) 12

[3.7b Activity words for Comparison](#_j4fdl2u6f4mp) 13

[**4.0 Results and Discussion**](#_h9w316wm4o43) **14**

[4.1 Visualisation](#_u7s82qy86jsy) 14

[4.1a Word2Vec](#_csnr4o3sd7yt) 14

[4.1b Word2Vec with Gensim Phraser](#_lo7zhpkrxo94) 14

[4.1c FastText](#_42ki9rg45kcb) 15

[4.2 Clustering using K-means clustering](#_5nzwve4ft2sc) 15

[4.3 Comparison with Activity words](#_xx51pz6bqrst) 16

[4.4 General Comparison of Word Embedding Techniques](#_20ipga9fsqd5) 17

[4.5 Limitations of the system](#_lx0fupxcjo6c) 18

[**5.0 Conclusion**](#_cjgmgld8dyrt) **19**

[**References**](#_ooruebx3j1p3) **20**

# 

# Abstract

Industries with specialised terminology such as the construction industry either do not have a publicly available lexicon, or have one that is very limited. Without a proper lexicon, it is difficult to employ Natural Language Processing tasks as there is no way to correctly represent the unstructured text in a way that can be understood by machines. In order to solve this issue, this paper uses a few word embedding techniques in an attempt to represent the words in a construction corpus. Specifically, the Word2Vec and FastText for word embeddings. When trying to classify activity words, FastText gave the best results in embedding the construction corpus.

# 

# Acknowledgements

I am very grateful to Prof Sian Lun Lau for giving me this opportunity to conduct this NLP-based project under his care and guidance throughout the entirety of this project. I would also like to thank Mr Ziqing Liew from Sunway Construction Group, for providing the construction data and scoping the problem.

# 

# 1.0 Introduction

Documentation could be considered a core process in nearly all projects for every industry. It is used to describe the project in detail while being easily understood by humans. However, a problem that commonly occurs with project documentation is the amount of documentation that accumulates quickly over time. The large amount of accumulated documentation becomes extremely time consuming for a person to sieve through the entirety of it to glean relevant information.

Technology has played a role in reducing this inefficient use of time by providing simple tools such as text-to-text matching. As an example, the basic search function that is ubiquitously seen in data files can only perform searches which match the exact text to the user’s input. This function, albeit useful for simple searches, can only produce limited results and pale in comparison to modern day search engines like Google Search which can provide much richer results on user searches. If industries could leverage the advanced search engines seen on browsers, they would be able to use their data more effectively for downstream applications.

The disparity between online search engines and the simple search function on a Word document can be mainly attributed to one thing: data. Online search engines can make use of the huge amount of textual data readily available on the net, to train their machine learning models. These models would then allow their search engines to have a language model that can represent and mimic languages better. These models which are pre-trained on billions of words are also transferable to local applications.

However, industries would face a problem when trying to use the same model, that is, the language used by the public is very different from the jargon used in their industry. The language models trained online would represent words differently from the words used in the industry-specific words. For example, if you search the word ‘casting’ online, it would return searches related to ‘fishing’, ‘spellcasting’, ‘games’, ‘actors’, or ‘streamers’. However, in the construction industry, the word ‘casting’ is related to a liquid material being poured into a mold.

To try and tackle this problem, this paper applies several Natural Language Processing (NLP) techniques in an attempt to model the construction industry’s language for further downstream applications. Three word embedding techniques – Word2Vec, Word2Vec with Gensim Phraser, and FastText – were used to create the word representation for the construction corpus. They were then compared based on their ability to capture the semantics of words by clustering and average cosine similarity to a self-curated ground-truth pair of words.

# 

# 2.0 Literature Review

NLP is a subfield, combining the areas of linguistics, computer science, and artificial intelligence that aims toward learning, understanding, recognizing, and producing human language content (Hirschberg & Manning, 2015). It is a widely studied topic that has produced several main applications such as word sense disambiguation, machine translation, summarization, syntactic annotation, and named entity recognition (Zeroual & Lakhouaja, 2018).

Indurkhya et al. (2010) states “It is universally acknowledged that ordinary language use involves a more or less seamless integration of linguistic knowledge, cultural conventions, and real-world knowledge” (p. 94). As humans, we easily understand common speech and writings because we understand the context behind the words. An approach taken by the NLP community to analyse the meanings of words by its context is through Semantic Analysis. Semantic Analysis refers to analyzing the meanings of words, fixed expressions, whole sentences and utterances in context (Indurkhya et al., 2010, p. 94). This is crucial for two reasons, first, to reflect the cognitive reality of ordinary speakers, and second, to achieve comparative and interoperability between different systems of semantic description (Indurkhya et al., 2010, p. 93). Essentially, Semantic Analysis is one of the core principles of NLP in representing words in different senses.

Current NLP language modeling is mostly focused around the issues of word polysemy and homonymy that are prevalent in natural language. Word polysemy is the multiple meanings that can be associated with a word. For example, ‘hot’ could mean differently in context with ‘heat’ or ‘another person’, or even phrases like ‘hot cakes’. Word homonymy refers to words that are spelt the same or sound the same but mean different things. Examples would be ‘a bright light’ versus ‘light weights’ and ‘night’ versus ‘knight’.

To solve these issues, a publicly available lexical resource could be used such as WordNet, FrameNet, and Levin Classes to help distinguish the different word senses and offer major improvements in downstream tasks (Hanks & Pustejovsky, 2005). However, Rashid et al. (2020) finds that across different institutions, their lexical databases do not adhere to a common metadata standard and lack support for formal semantics. Looking at the larger picture, institutions in specialised industries would not have cross-compatible lexical databases. Thus, the limited amount of lexical resources in specialised industries lead to the limited use of NLP applications in those industries.

Fortunately, there are other ways to model the language used in specialised industries without the need to build a lexical database of jargon used in that industry. One of them is using machine learning methods for word representations. Word representations in NLP have been studied extensively by researchers in literature. In the past, words were represented through the simple one-hot encoding word representation. It simply represents a word as a matrix with only 1 non-zero entry. However, there were many criticisms upon its ability to capture word semantics between words and it faces huge data sparsity (Wang et al., 2019). The ability for word representations to be able to capture word semantics between words came about with the popular use of the distributional hypothesis, which states: ‘linguistic items with similar distributions have similar meanings’. It provided a basis for the idea of integrating semantics into word representations, and has garnered use in the subsequent development of word representation models (Wang et al., 2019).

Today, Neural Network-based Word embeddings have become mainstream in word representation and building downstream NLP tasks. Since it was popularised by the famous Word2Vec in 2013(Mikolov et al., 2013), the field has made large advancements by building more efficient and better models in capturing word semantics in raw texts such as Global Vectors (GloVe) (Pennington et al., 2014) which use global statistical information. ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) are newer models that have dynamic vector representation of words in different contexts. These new models try to improve upon Word2Vec in these three main aspects: Out Of Vocabulary (OOV) words, Textual representation in different contexts, and Word Embeddings of different languages. However, only Word2Vec’s algorithm was used in this paper.

Similar works that utilise word representations for building downstream NLP tasks like text classification or information retrieval can also be found in other industries. The medical industry for example, has publicly available medical datasets and working models working on text classification (Weng et al., 2017). Further specialised industry focused works are information extraction on regulation for construction management systems (Zhang & El-Gohary, 2016) and article classification for dengue research (Li et al., 2020).

# 

# 3.0 Methodology

## 3.1 Overview

This paper is focused on improving the comprehension of machines in human language, in the construction domain. To that end, NLP is integrated with machine learning tools to develop a system that learns on unlabeled construction data provided by Sunway Construction Group. The data was provided by Sunway Construction Group so that a preliminary model can be trained on and tested. The system was built using Python 3.8.5. Within Python, the packages utilised were NLTK, Numpy, Pandas, Gensim, Sklearn and Matplotlib.

## 3.2 Construction Dataset

The dataset consists of construction data from 8 different construction projects. The data is stored under several sheets in an excel file. Each sheet represents a project and is listed as follows: ‘Parcel F’, ‘SIS’, ‘LRT3’, ‘Belfield’, ‘CP2’, ‘TNB’, ‘PNLC’, and ‘SMC4’. The words to be embedded are stored in the column ‘task\_name’ for each project sheet.

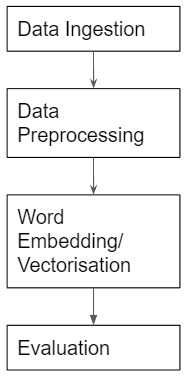
## 

## 3.3 System Description

A system to automatically embed words with vectors is proposed in this paper. As the data is unlabeled, the system uses several unsupervised methods to embed the words. Then, they are compared against each other through clustering and visualisation. The system consists of four main parts which are, data ingestion, data preprocessing, word embedding/vectorisation, and evaluation. The system flow is shown [Figure 1](#vdjffi74fv3s).

**Figure 1**

*System Flow*



## 3.4 Data Ingestion

Data ingestion is needed to input the raw construction data into a desired format to be ready for data preprocessing. The data which is found on ProjectsData.xlsx excel file is aggregated into one Pandas dataframe. This is done by using Pandas package to concatenate each project excel sheet into a single dataframe. Combining them together, the length of the dataframe is 50347.

The main corpus that the system is trained on will only be using the data in column ‘task\_name’. The column ‘task\_name’ consists of strings and has a total of 50347 observations in it. The observations had a maximum length of 16 words and a minimum length of 1 word.

## 3.5 Data Preprocessing of Text

Data preprocessing of text is the next part in the system. It consists of several steps in transforming the text to fit downstream techniques and algorithms. The steps in data preprocessing for the first corpus is done as follows: removing punctuations, special characters, and stand-alone numbers, lowercasing text, lemmatizing, and tokenization.

The first step is removing all punctuation and stand-alone numbers in the main corpus. This step is performed as punctuations and special characters in the corpus are not important to the word embedding models used later. Stand-alone numbers in the corpus are also removed because they have too many word disambiguations associated with it. For example, a number ‘5’, could be associated with number of items (5 cars), time (5 pm), number of activities (5 drawings), or location (level 5).

The second step involves lowercasing text. All text in the corpus are lowercased so that the system is case insensitive. This helps to count similar words as one rather than being two separate words due to their casing. As an example, ‘Drawing’, and ‘drawing’ are both lowercased to become ‘drawing’.

The third step is lemmatizing. The lemmatization process converts words to their base form. The word ‘batches’ for example, will be converted to ‘batch’ through this process. It uses the ‘wordnet’ lexical database to identify words to their base forms. Lemmatization is used instead of stemming as it provides a better conversion of words at an increase in processing resources needed. This increase is negligible as the corpus is quite small.

The fourth step is tokenization. Tokenization breaks down the text into individual elements called tokens which would then be used in downstream modeling techniques. In this case, each token represents a word.

The end result is a list of tokens which has a vocabulary size of 1752 and 256556 tokens (word occurrences). This corpus is called “Corpus A”.

In addition to that, a step was added to remove all numbers to create a second corpus, “Corpus B”, for comparison later. This is done as there are certain words such as ‘f2’, ‘f3’, ‘f4’, and so on that can be reduced to a single ‘f’, denoting ‘floor’. Using this corpus may give better results in providing more occurrences for these groups of words as the corpus is already small. This corpus also reduces the vocabulary size to 1118.

## 

## 3.6 Word Embedding/Vectorisation

As the corpus is unlabeled, several unsupervised word embedding techniques were trained on each corpus to create vector representations of each word. The word embedding techniques used are Word2vec, Word2vec with Gensim Phraser, and FastText. The vector representation of each word would then allow the system to mathematically calculate the similarities of words through the use of cosine similarity on the vectors. Cosine similarity is the dot product of two vectors divided by the product of their magnitudes. The cosine similarity formula is shown as:

Cosine similarity:

### 3.6a Word2Vec

Word2vec is an algorithm that learns word representations through a shallow neural network architecture consisting of 1 hidden layer. The output of the word representation is a vector representation of the word with n-dimensions. The algorithm has two architectures to capture the semantics of words: CBOW or Skip-gram. CBOW architecture uses the surrounding words to predict the middle word while Skip-gram architecture uses the middle word to predict the surrounding words (Mikolov et al., 2013). Between the two, Skip-gram was chosen. The number of dimensions used is 200 and the number of iterations is 10.

### 3.6b Word2vec with Gensim Phraser

This technique is similar to using Word2vec, but it adds a Gensim phraser to allow the model to embed phrases instead of single words only. The Gensim phraser is inspired by Mikolov et. al. (2013), for learning meaningful phrases. They were able to do this by finding words that appear frequently together but also infrequently in other contexts to avoid meaningless bigrams of phrases such as “this is”.

The minimum count set for words to appear together to be considered a phrase is 5 and the minimum threshold for the score is 3.

Due to introducing new vocabulary by identifying phrases, the vocabulary size of “Corpus A” increased to 2183 whereas “Corpus B” increased to 1362.

### 3.6c FastText

FastText is another implementation of Word2vec developed by Facebook, that takes into account subword information (Mikolov et. al. 2017). This is an extension of the Word2vec Skip-gram model by modelling word morphology as a sum of its character n-grams.

## 3.7 Evaluation

Evaluating the quality of word embeddings is often difficult (Quality evaluating). Thus, only a subset of words in the corpus is evaluated. They are evaluated first by seeing if K-means clustering could give insight on how the word embedding techniques represent words in different clusters, and second, by using a ground truth word pairs and the average cosine similarity in each word embedding technique.

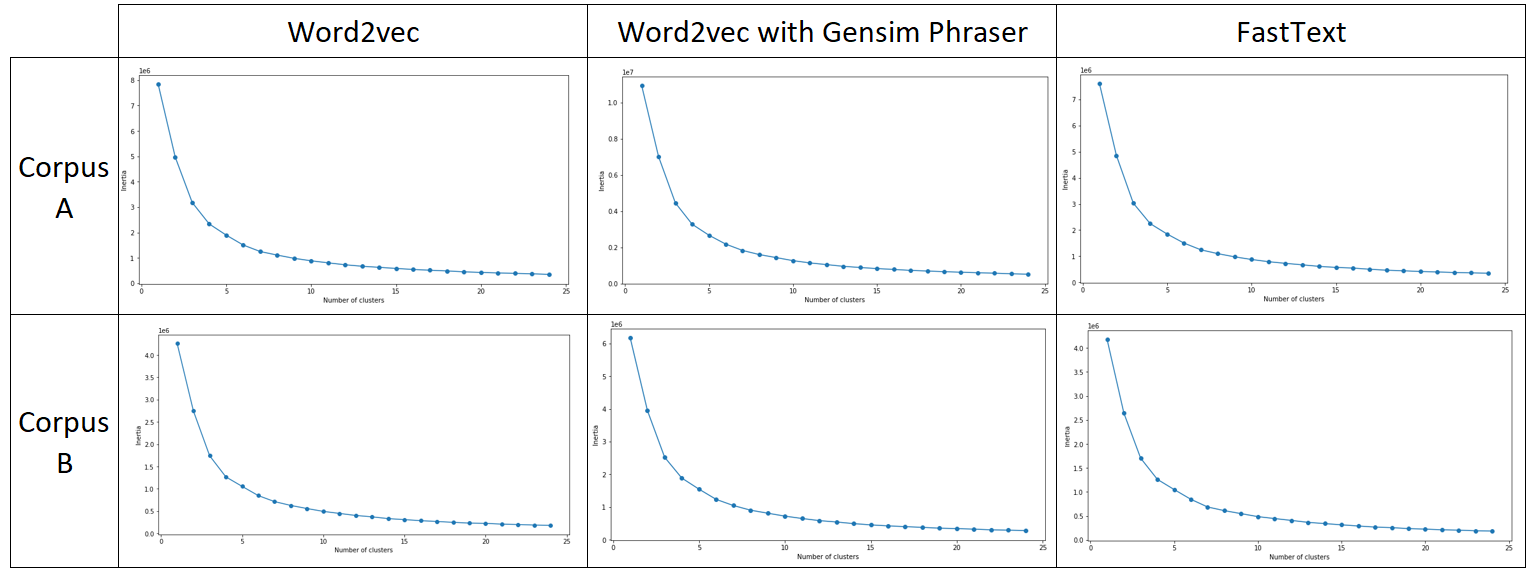
### 3.7a K-means Clustering

In literature, clustering can be used to aid downstream text classification or to discover the kind of structure in the training examples (Thomas & Resmipriya, 2016). The latter is more important for viewing the efficacy of word representations in this paper. The K-means clustering technique was used to cluster the words by their vector representations.

To measure the appropriate number of clusters to use, we used the metric inertia. Inertia, sometimes called within-cluster sum-of-squares criterion, is a measure of how internally coherent clusters are. The metric allows us to create elbow plots, shown in [Figure 2](#5h8pt2g58rnv), which show the optimal number of clusters to be used. The optimal number of clusters for each word embedding technique and their respective corpus was determined to be around 10.

**Figure 2**

*Elbow plots for each word embedding technique*



The clusters were then visualised to see which models were able to differentiate words better. The further interdistance of cluster centroids and the closer the intradistance of cluster points, the better the word differentiation is. The words in the clusters were also examined to see if they were able to be clustered into thematic groups.

### 3.7b Activity words for Comparison

To evaluate the performance of each model, the cosine similarity between ‘activity words’ in each model was utilised. Doing that, it is possible to measure somewhat the effectiveness of each model in classifying ‘activity words’.

With reference to the “Base Activities” excel sheet provided by Sunway Construction Group, 10 activity words were chosen: ‘casting’, ‘drawing’, ‘plastering’, ‘proofing’, ‘tiling’, ‘widening’, ‘installation’, ‘laying’, ‘cabling’, ‘coating’ . To compare each model, the top 5 closest words for each activity word were taken for each model and were manually classified into “activity word” or “non-activity word”. The model was then measured using two metrics in recognising activity words, ‘average correct activity cosine similarity (ACACS)’ and ‘activity classification score’, given as:

Average correct activity cosine similarity =

Activity classification score =

*Where ɑ = activity word*

ACACS measures how closely are the activity words in the corpus recognised by the trained word embedding technique. Activity classification score measures the model overall performance in identifying similar activity words given the list of activity words.

# 

# 4.0 Results and Discussion

In this section, the effectiveness of each word embedding technique is compared and discussed.

## 4.1 Visualisation

A variation of base words in three different categories of words – “Activity”, “Location”, and “Malay” – were plotted along with 5 of their most similar words for each word embedding technique to briefly see how they differ from each other. The distance between two points indicate the similarity for the word. The further the distance, the less similar it is. The base words used are listed as follows: Activity (‘casting’, ‘hacking’), Location (‘f’, ‘level', ‘b2’, ‘f3’, ‘l1’, ‘swimming'), Items(‘drawing’) Malay (‘jalan’, ‘kawasan’). The results are shown in [Appendix A](#pomvrn5n21zo) for reference.

### 4.1a Word2Vec

Embedding the corpus with Word2vec in Corpus A is of mixed results. Activity words have similar vector representations as most of the similar words associated with ‘casting’ and ‘hacking’ are also activities themselves. However, for Location words, ‘swimming’ did not have good vector representation as it has dissimilar words close to it such as ‘receival’ and ‘vertical’. Similar words for Items could be considered related, but it would depend mostly on the context of the document. Malay words were mostly grouped together without any closely related words in English. This could mean that Word2Vec can pick out Malay words from the corpus.

For Corpus B, the word list is changed to account for the deletion of all numbers. For example, ‘b2’, ‘f3’, and ‘l1’ were changed to ‘b’, ‘f’, and ‘l’ respectively. Embedding the corpus with Word2Vec in Corpus B gives a different result. Activity words had similar words relating to their context. For example, ‘hacking’ would have similar words like ‘balance’ and ‘buttress’ which are not activity words but would be relevant in the context of ‘hacking’. Location words mostly have similar words in location as well, except for ‘swimming’ which has dissimilar words close to it as well. To note, the location word ‘b’ has grouped many single letter words. Items word ‘drawing’ has more or less similar words as in Corpus A. The Malay words are also grouped together and have almost the same similar words.

### 4.1b Word2Vec with Gensim Phraser

When the corpus was slightly transformed by the Gensim Phraser, the Word2vec algorithm was able to vectorise common phrases. For examples, the words ‘swimming’ and ‘pool’ were merged to become the bigram ‘swimming\_pool’, and ‘issuance’, ‘of’, ‘construction’ were merged to become the trigram ‘issuance\_of\_construction’. From this, we could find similar words to the phrases.

For both corpus, the words ‘swimming’ and ‘kawasan’ were changed to ‘swimming\_pool’ and ‘kawasan\_station’ respectively due to their original words being transformed by Gensim Phraser. They also gave similar results to Word2ec in both corpus. However, for the phrases, they have different similar words compared to their original counterparts in Word2vec. ‘kawasan\_station’ has the biggest improvement. It has similar Malay phrases around it consisting of names such as ‘sri\_andalas’, ‘jalan\_meru’, and ‘klang\_station’. However, the Malay word ‘jalan’ was found not to be grouped with any other Malay words, but instead more construction terms like ‘road\_kerb’, ‘premix’, and ‘widening’. This suggests Word2vec with Gensim Phraser is better at identifying Malay location names.

### 4.1c FastText

For the FastText plot, the two base words were intentionally changed to include a typo, ‘casting’, and ‘drawing’ became ‘castig’, and ‘dawing’ respectively.

In both corpus, the similar words for each base word are much more spread out than the previous two. This shows that the word similarities in FastText could be lower. In the plot, it seems words with typos are considered very similar. For example, ‘castig’ and ‘dawing’ both have their original words ‘casting’ and ‘drawing’ similar to them. Surprisingly, for corpus B, although there are no longer locations with several numbers, the algorithm plots ‘l1’, ‘f3’, ‘f’, and ‘b2’, close together, showing they have similar vector representations. Unfortunately, their similar words were mostly dissimilar.

## 4.2 Clustering using K-means clustering

By using K-means to cluster the word vectors, we visualised the vector representation of words in the corpus while trying to see the differences in word clusters of different word embedding techniques.

By visualising the 10 cluster plots as shown in [Figure B1](#nu1qqz3sy8un). Overall, the different word embedding techniques did not seem to have much difference in clustering the words.

However, clustering with different numbers of clusters gave different results. When following the optimal number clusters, 10, shown in [Figure 2](#5h8pt2g58rnv) it did not give good results as there are too many words in one cluster. It was difficult to find a common theme in the word cluster (see [Figure B2](#spjh1tde76ju)). When clustering with more clusters such as 30 clusters or 50 clusters they gave slightly better results in grouping words with similar semantics as shown in [Figure B3](#dzrx5x9qsfrn) . Some clusters are easier to see a common theme such as location, but others are much more difficult when they have activities in them.

## 4.3 Comparison with Activity words

The results on recognising activity words (see Figure [B4](#s49yp138l5ux)-[B9](#xdgun18q9qfq)) are given below in [Table 1](#d3jrk37eqpuc).

**Table 1**

*Scores of different word embedding techniques on recognising activity words given the closest 5 similar words of 10 activity words*

|  | Word2Vec  Corpus A | Word2Vec  Corpus B | Word2Vec with Gensim Phraser Corpus A | Word2Vec with Gensim Phraser Corpus B | FastText  Corpus A | FastText  Corpus B |
| --- | --- | --- | --- | --- | --- | --- |
| No. of similar words | 11 | 7 | 12 | 12 | 21 | 25 |
| ACACS score | 0.70503 | 0.77254 | 0.85823 | 0.84045 | 0.80163 | 0.80625 |
| Activity classification score | 0.15510 | 0.10816 | 0.20598 | 0.20171 | 0.33669 | 0.40312 |

From Table 1, we can see that if trying to build an activity classification model, FastText Corpus B would be the most suitable. Given the closest 5 similar words to 10 activity words, it is able to return 25 activity words. Its activity classification score is also the highest at 0.40312, meaning that it is able to perform the best overall in identifying similar activity words given the list of activity words. It is also shown that removing all numbers in the text may improve the performance in classifying activity words.

On the other hand, Word2Vec with Gensim Phraser seems to have the highest ACACS score. This shows that the activity words in this technique have their vector representation closest to each other out of all the word embedding techniques.

## 

## 4.4 General Comparison of Word Embedding Techniques

The different word embedding techniques are compared based on their complexity, word vector representations and features.

Out of all the word embedding techniques used in this paper, Word2Vec is the most simple and easiest to deploy. Its vector representation of words can identify words that may be similar to each other to varying degrees of success.

Word2vec with Gensim Phraser is slightly more complex as it identifies phrases that occur in the text. The vector representation of terms here seems to have captured more contextual information with phrases and have more similar words based on context rather than type of word compared to Word2vec. For example, many of the Malay words are actually phrases of names, e.g. ‘taman\_besar’, ‘taman\_selatan’, ‘jalan\_meru’, ‘klang\_station’. The basic Word2Vec represents these phrases as individual words which may not be desired in certain downstream tasks. Furthermore, the Malay word ‘jalan’ has similar words such as ‘widening’, ‘road\_kerb’, and ‘premix’ which hints that it finds similar words based on context. Other successful instances of English phrasing are ‘swimming\_pool’, ‘southern\_elevation’ and ‘air\_cond’. Unfortunately, it also considers some uninformative words as a part of a phrase such as ‘do\_app’, ‘do\_sub’, ‘make\_good’ and ‘tray\_tray’. This could have occurred as they are considered uncommon in this corpus which in turn, affects the Gensim Phraser algorithm to detect it as part of a phrase.

The FastText word embedding technique is more complex as it learns on subwords such as 2/3/4/5-gram of characters. Its vector representation of words seems to be worse compared to the previous two. Most of the words ending in -ing are found to be similar, giving some bias. However, in construction, terms like ‘hacking’, and ‘footing’ are not similar at all and thus, this might be detrimental for word representation for construction terms. A benefit of FastText is that it can represent OOV words which is beneficial to generalise to foreign words in this small corpus. Taking the existence of Malay and English words in this corpus, some construction words in the Malay language which has similar morphology to its synonym in the English language could have similar vector representations. As an example, ‘kasting’ would be very similar to ‘casting’. In addition to that, typos or slight misspellings would have similar vectors to their base word. For example, ‘castig’ was found to be similar to ‘casting’ and ‘cast’ while ‘dawig’ was found to be similar to ‘drawing’.

## 

## 4.5 Limitations of the system

The corpus used for this system consists of the aggregated short documents from 8 construction projects. From these few projects, it is difficult to say that the system is able to generate word representations that can generalise to other construction projects.

Furthermore, there is the issue of disproportionate sampling (Indurkhya & Damerau, 2010). The different construction projects have varying amounts of text which can skew the word representations towards the projects with more data in them.

Another problem is the small corpus size. With only 50347 lines of text with 16 words maximum, the model may fail in representing vocabulary with very few occurrences. The rare words are problematic because the word embedding algorithms used rely on the basis of contextual words and suffer from the rare word sparsity problem (Li et al., 2016). According to a survey done by (Zeroual & Lakhouja, 2018), it may take a few years to produce a corpus that is deemed large enough for language modeling purposes.

The system is also limited in representing documents. It is only able to represent words up to bigrams and trigrams due to Gensim Phraser. For future document representation using vectors, consider using Doc2Vec (Le & Mikolove, 2014).

# 

# 5.0 Conclusion

For some time, NLP techniques have been applied to extract insights from documentation in specialised industries such as the medical industry (Weng et al., 2017). However, other industries like the construction industry do not have resources such as a construction lexicon to apply the same NLP techniques to extract insights from construction documents. In this paper, a few word embedding techniques with the Word2Vec architecture as the basis, were used to model the construction language of a corpus consisting of 8 construction projects with 50347 short documents in total. The different word embedding techniques have benefits and downsides to consider. Word2Vec by itself is quick to train and easy to deploy while giving decent word representations, but is generally worse than the other two. Word2Vec combined with Gensim Phraser, was able to represent words more contextually with bigrams and trigrams, but suffers from phrasing unrelated words in a small corpus. FastText is good to account for OOV words and foreign words but suffers a bias for words that are spelt similarly. When comparing the different word embedding techniques to classify ‘activity words’, FastText is deemed to work the best in this small corpus. This study is an important step towards building downstream NLP tasks with word embeddings in construction related texts.

# References

Devlin, J., Chang M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Hanks, P., & Pustejovsky, J. (2005). A pattern dictionary for Natural Language Processing. Revue Française De Linguistique Appliquée, X(2), 63–82. https://doi.org/10.3917/rfla.102.82

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. Science, 349(6245), 261–266. https://doi.org/10.1126/science.aaa8685

Indurkhya, N., & Damerau, F. (2010). Handbook of Natural Language Processing. ProQuest Ebook Centra. CRC Press. Retrieved from http://ebookcentral.proquest.com/lib/sunway/detail.action?docID=565922.

Li, W., Wu, Y., & Lv, X. (2016). Improving word vector with prior knowledge in semantic dictionary. Natural Language Understanding and Intelligent Applications, 461–469. https://doi.org/10.1007/978-3-319-50496-4\_38

Li, Z., Gurgel, H., Dessay, N., Hu, L., Xu, L., & Gong, P. (2020). Semi-supervised text classification framework: An overview of dengue landscape factors and satellite Earth Observation. International Journal of Environmental Research and Public Health, 17(12), 4509. https://doi.org/10.3390/ijerph17124509

Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. https://arxiv.org/abs/1405.4053

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. https://arxiv.org/abs/1310.4546

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). https://doi.org/10.3115/v1/d14-1162

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). https://doi.org/10.18653/v1/n18-1202

Rashid, S. M., McCusker, J. P., Pinheiro, P., Bax, M. P., Santos, H. O., Stingone, J. A., Das, A. K., & McGuinness, D. L. (2020). The semantic data dictionary – an approach for describing and annotating data. Data Intelligence, 2(4), 443–486. https://doi.org/10.1162/dint\_a\_00058

Thomas, A. M., & Resmipriya, M. G. (2016). An efficient text classification scheme using clustering. Procedia Technology, 24, 1220–1225. https://doi.org/10.1016/j.protcy.2016.05.095

Wang, S., Zhou, W., & Jiang, C. (2019). A survey of word embeddings based on Deep Learning. Computing, 102(3), 717–740. https://doi.org/10.1007/s00607-019-00768-7

Weng, W.-H., Wagholikar, K. B., McCray, A. T., Szolovits, P., & Chueh, H. C. (2017). Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach. BMC Medical Informatics and Decision Making, 17(1). https://doi.org/10.1186/s12911-017-0556-8

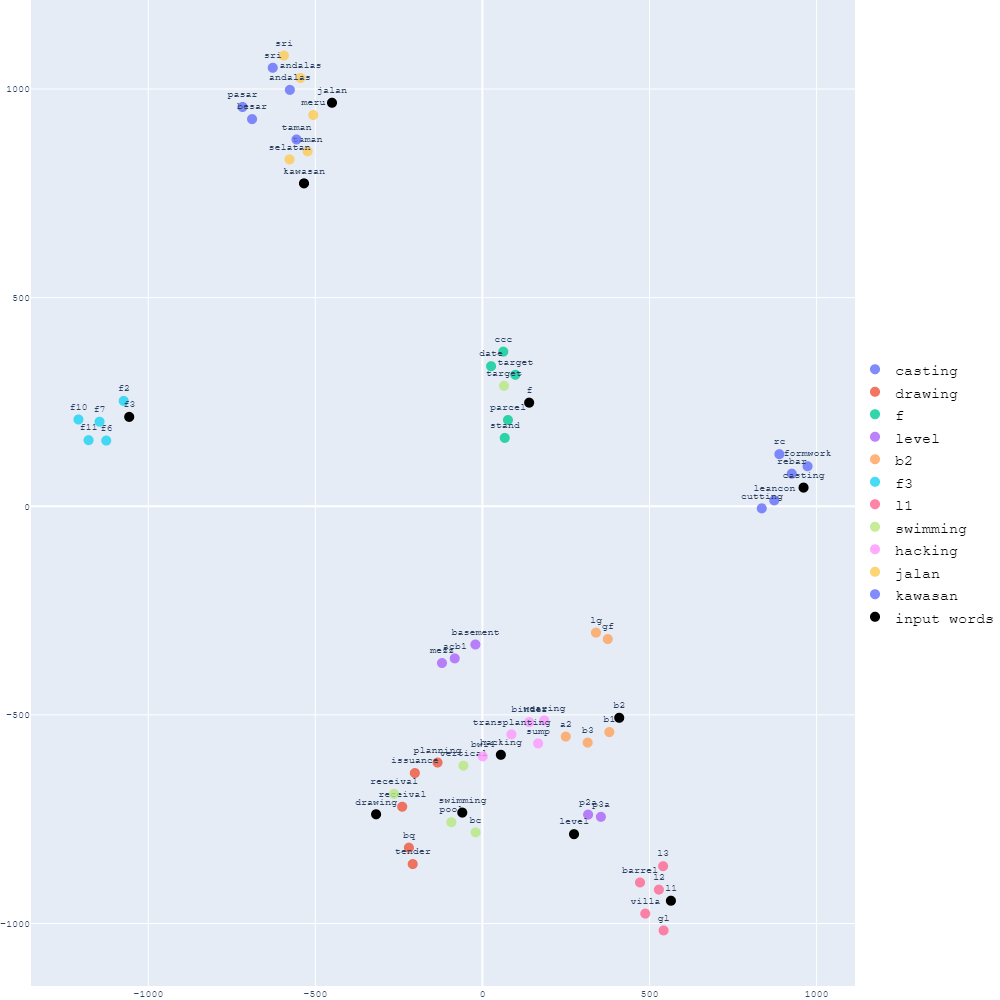
Zeroual, I., & Lakhouaja, A. (2018). Data science in light of Natural Language Processing: An overview. Procedia Computer Science, 127, 82–91. https://doi.org/10.1016/j.procs.2018.01.101

Zhang, J., & El-Gohary, N. M. (2016). Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. Journal of Computing in Civil Engineering, 30(2), 04015014. https://doi.org/10.1061/(asce)cp.1943-5487.0000346

**Appendix A**

**Figure A1**

*Word2vec similar words graph for Corpus A*



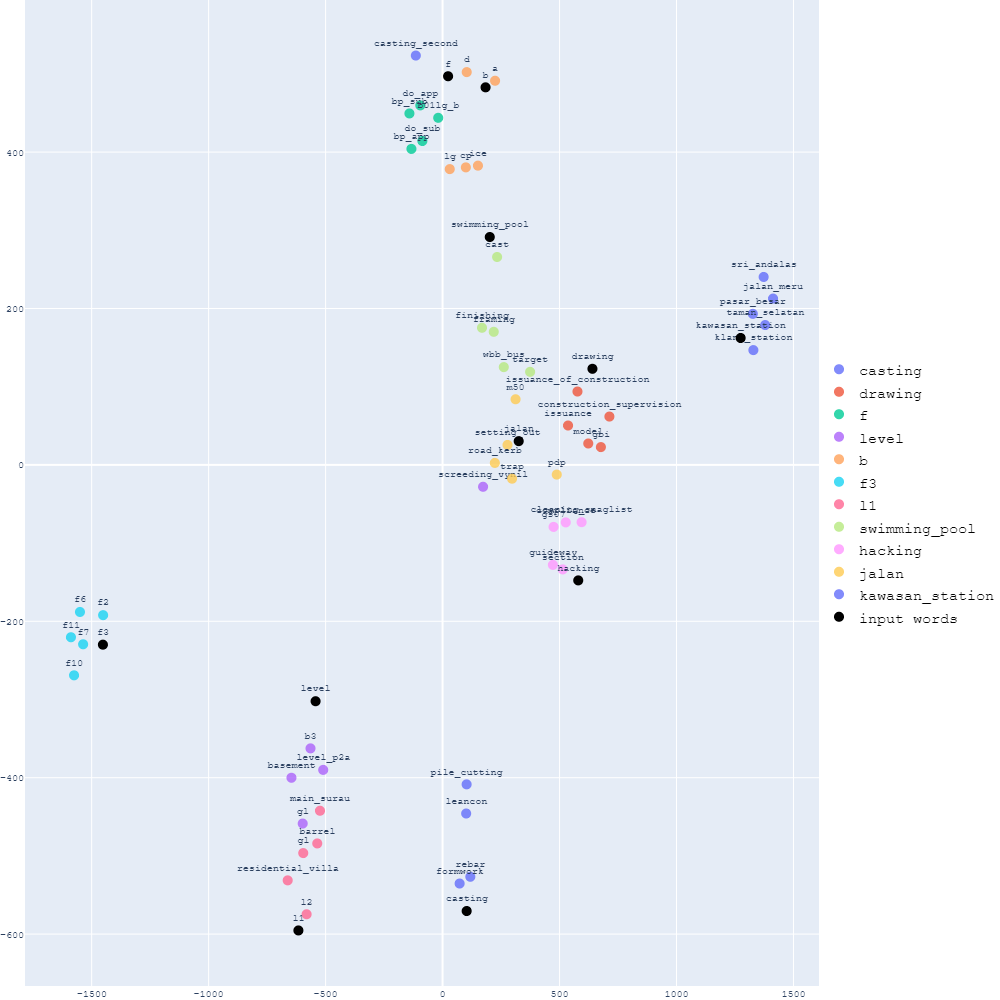
**Figure A2**

*Word2vec similar words graph for Corpus B*



**Figure A3**

*Word2vec with Gensim Phraser similar words graph for Corpus A*



**Figure A4**   
*Word2vec with Gensim Phraser similar words graph for Corpus B*



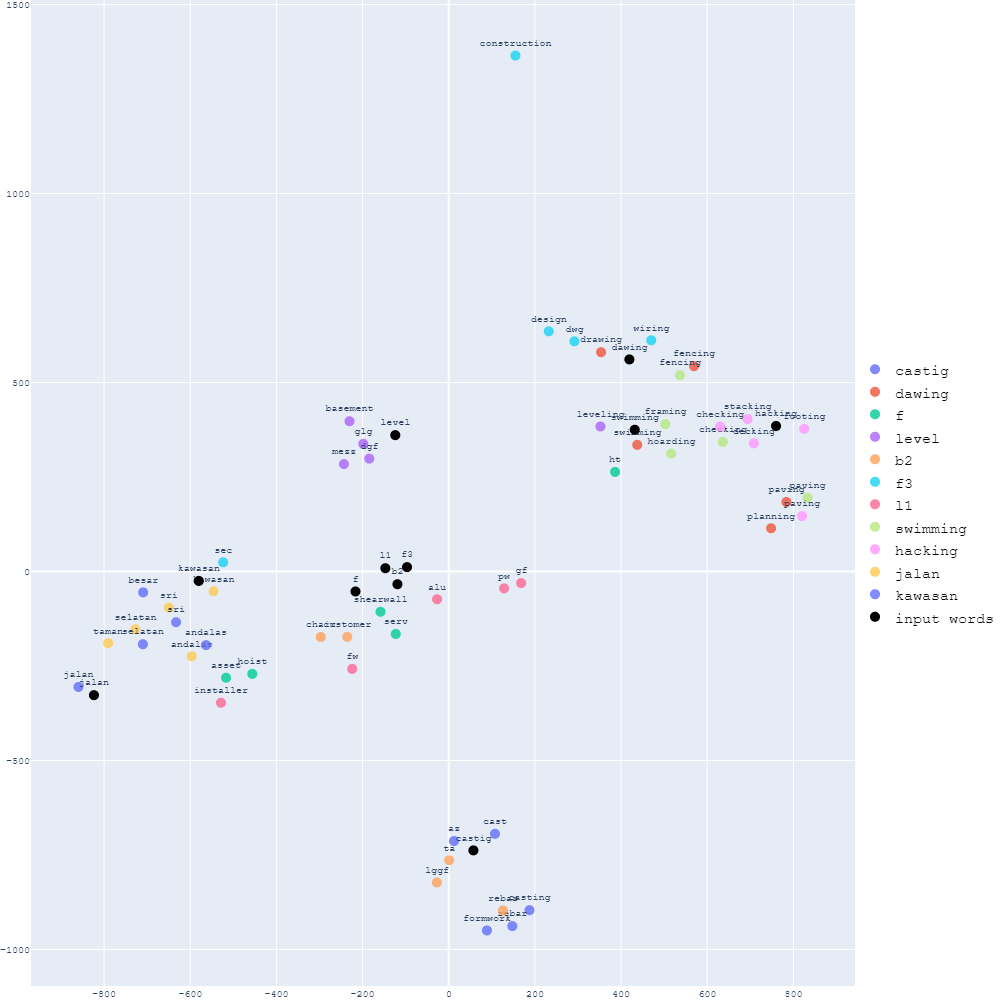
**Figure A5**

*FastText similar words graph for Corpus A*



**Figure A6**

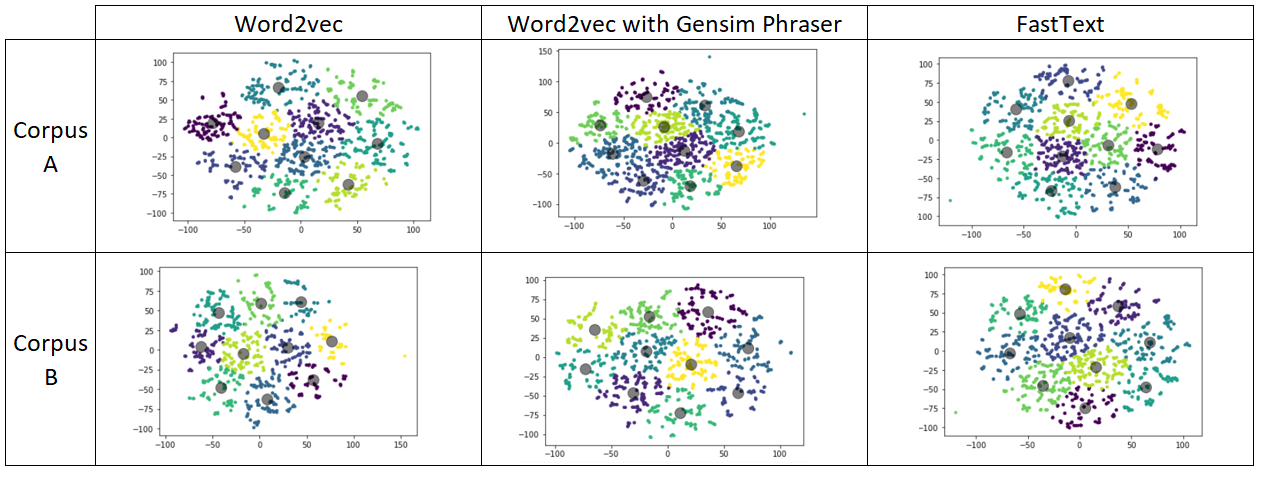
*FastText similar words graph for Corpus B*



**Appendix B**

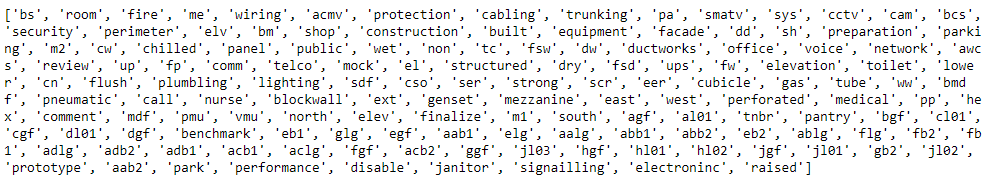
**Figure B1**

*Cluster plots of different word embedding techniques*



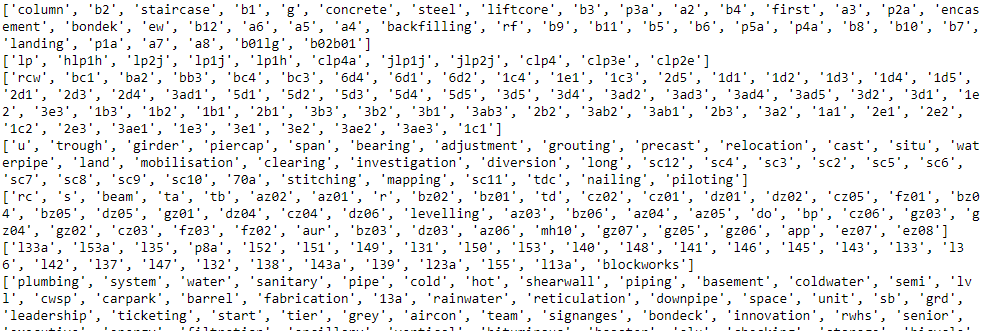
**Figure B2**

*Sample cluster when clustering with 10 clusters*

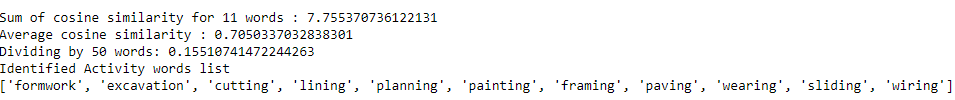
****

**Figure B3**

*Sample clusters with location-themed words from Word2Vec (50 clusters)*

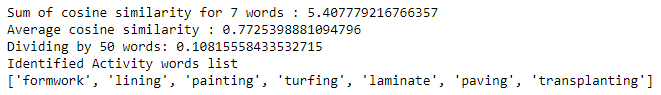


**Figure B4**

*Activity words Comparison for Word2Vec Corpus A*

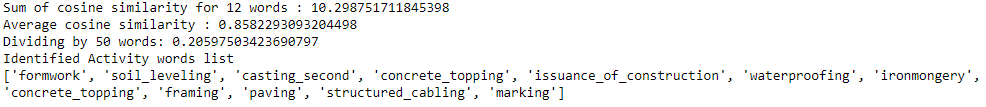
**Figure B5**

*Activity words Comparison for Word2Vec Corpus B*



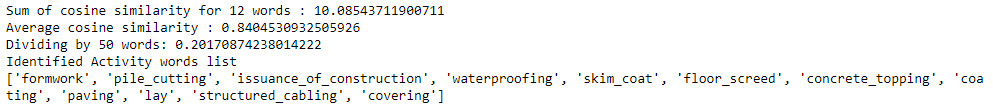
**Figure B6**

*Activity words Comparison for Word2Vec with Gensim Phrases Corpus A*



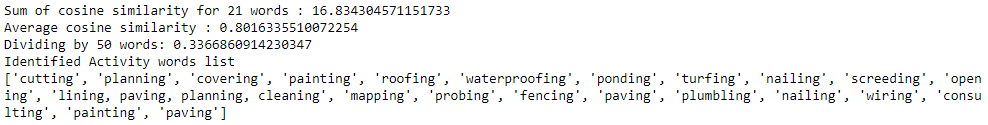
**Figure B7**

*Activity words Comparison for Word2Vec with Gensim Phrases Corpus B*



**Figure B8**

*Activity words Comparison for FastText Corpus A*



**Figure B9**

*Activity words Comparison for FastText Corpus B*

