Self-Organising Neural Networks with Word Embedding

Patrick Tjahjadi

16 October 2020

Table of Contents

Abstract

Word embedding is widely used in the field of natural language processing that aims to determine the relationships between words. It provides a robust and compressed representation of words using distributed values. Fusion Adaptive Resonance Theory (Fusion ART), an application of machine learning that encode and retrieve data using numerical values, can have their values represented through word embedding. As an extension of Tjahjadi's previous paper, this paper introduces word embedding in Fusion ART (Tjahjadi, 2020). In addition, several experiments using code to simulate Fusion ART models are conducted to illustrate the process involved and the viability of the models. The process of encoding and retrieving data can be done by calculating similarities between words, using an intrinsic vector representation of each word or generating a probabilistic language model. Future improvements can then be suggested after experimenting through the code.

Acknowledgements

I would like to express my appreciation to Prof. Tan Ah Hwee of the Singapore Management University, who introduced me to Adaptive Resonance Theory and its generalisation to fusion models through various research articles, and who encouraged me to write this paper.

In addition, I would like to thank Dr. Budhitama Subagdja of Nanyang Technological University for creating the API to develop the Fusion ART models used in this paper. I also would like to thank him for answering my questions regarding his API and Adaptive Resonance Theory in general.

This paper would not have been possible without their guidance.

1. Introduction

Word embedding is a prevalent technique in natural language processing which seeks to identify words used in similar contexts and meaning. Word embedding aims to capture relationships between words and represent words with distributed numerical values in the form of vectors. Using cosine similarity as a measure of calculating similarity, words can be semantically analysed to determine other words that may be used in a similar context (Ganguly et al., 2015).

Many machine learning algorithms experience a problem where they forget past information while learning new information. Adaptive Resonance Theory (ART) arises from the need to balance acquiring new information and retaining previously acquired ones. This problem is called the stability-plasticity dilemma, in which ART aims to alleviate (Grossberg, 2013). Particularly, ART concerns a type of neural network that uses unsupervised machine learning methods.

There are many variations of ART models, including ART 1 that accepts binary inputs, ART 2 that accepts continuous inputs and Fuzzy ART that implements fuzzy logic (Carpenter & Grossberg, 2002). Nguyen et al. (2008) suggests that Fusion ART is a generalisation of ART architecture that uses multiple pattern channels instead of one. Unlike ART, Fusion ART falls within several machine learning paradigms, such as supervised learning, unsupervised learning, reinforcement learning, etc. that learns in an online and incremental manner (Tan et al., 2019).

The search cycle of ART concerns an input layer of nodes *F1* that accepts an input from a pattern. The *F1* layer then transfers the input to the recognition layer of nodes *F2* for comparison. This comparison is in the form of a hypothesis test. In this report, the aforementioned hypothesis test is called the vigilance test: for each node *J* in F2, a level of matching against the input is generated and if the match is close enough to the input pattern, the node *J* will learn from the mentioned input pattern (Carpenter & Grossberg, 2002). The vigilance test compares the level of matching against a defined vigilance parameter *rho*. The test succeeds if the level of matching is higher than *rho*. As such, as *rho* increases, the nodes tend to produce more detailed information. If the vigilance test fails, the process then iterates through all nodes in *F2* in sequential order until a successful test is found. If there are no successful tests, the uncommitted node in *F2* becomes committed, and a new uncommitted node is added which grows the network (Tan, Subagdja, Wang & Meng, 2019). The parameters of the Fusion ART model will be discussed in more detail in Chapter 3.

This report aims to present the findings from experiments regarding Fusion ART software, created by Dr. Budhitama Subagdja. The goal of this research is to implement Fusion ART in a word embedding context. Whilst binary encoding provides a binary vector representation of words in values of either 0 or 1, word embedding provides a distributed representation of vectors, comprising real values ranging from 0 to 1 (Tjahjadi, 2020). This can be used to encode and retrieve data, and by implementing noisy inputs, can be used to assess their performance against its binary counterparts.

2. Software and Dataset Materials

The software concerns a schema-based Fusion ART model, written in Python. This software develops a Fusion ART model that defines and specify fields with respect to a Python dictionary with attributes and values. The schema-based model allows each node in the *F1* layer to have an attribute name.

This paper concerns 3 experiments, separated into 3 Python notebooks: "professor-fusionART-word2vec-similarity.ipynb", which uses similarity as a measure of word embedding and "professor-fusionART-word2vec-representation.ipynb", which uses the word embedding model's word value as a measure of word embedding. The final experiment concerns the use of dynamic word embedding or language model, which build probabilistic models of word sequences that assigns each word to multiple vectors instead of one (Mnih & Hinton, 2009). The Python notebook for the language model is "professor-fusionART-BERT.ipynb".

The functions in which the schema-based Fusion ART models are developed can be found within the API "fusionART.py". These include the processes to create the model, such as code competition, resonance search and top-down *F1* search. The code "fusionART.py" also takes the functions in "ARTfunc.py". Whereas "fusionART.py" contains functions to develop the Fusion ART models, "ARTfunc.py" provides the computational methods necessary for the functions in "fusionART.py" to work.

The Fusion ART model is then run through a dataset. For this experiment, the dataset contains the names of 83 professors, along with their group, universities, and research interests. The dataset is named "SCSE ProfProfile". The aim is to represent the professors' data using distributed vectors before being encoded to the Fusion ART model.

A word embedding model is developed that represents words into values using the *gensim* library. The word embedding model learns from several text corpora before converting words into vectors of real values. The corpora include "it_text.txt", a text file containing a collection of IT terminologies and their definitions gathered from several news sites, research papers and tutorials (see Appendix A). Moreover, the word embedding model is trained using Wikipedia articles of all of the professors' research keywords using the *wikipedia* library (see Appendix B). The way the model is trained is by capturing the distances between words in the text corpora. Words that are close to each other in the corpora signifies that those words are similar in meaning, which would have a very high similarity score.

In this experiment, we expect to implement word embedding to each professor's research interests to determine the relationships between research keywords. Since names, groups and universities are distinct from each other, they are binary encoded using the *sklearn* library, while research interests are encoded using the *gensim* and *Transformers* library.

3. Experimental Analysis

3.1 Structure and Parameters

With respect to the code, the Fusion ART model concerns several underlying structures:

- Input layer *F1*: The input vectors that is equivalent to the number of attributes, which is represented as *attrib,* that are fed to the model schema. Each element of the input vector represents a real value $I \in [0, 1]$. During resonance search, the input vectors are represented by *val*.

- Competitive layer *F2*: Let $J = 0, \ldots, D{-}1$ denote the activity vector of size D. Initially, *F2* only contains 1 uncommitted node for learning. In itself, the activation value F2 of a particular node denotes its level of matching from bottom-up activation, which is part of the resonance search process. The uncommitted node of an *F2* layer has an F2 value of 0.

- Weight vector: The weight vector, shown in the code as *weights*, corresponds to the weight in which the J^{th} node in *F2* learns the input patterns in *F1* for all *I*. Hence, the weight vector has the same dimensions as the *F1* layer. Initially, all elements of the weight vector are set to 1, indicating that node is still uncommitted, and yet to undergo resonance search.

- Complement coding: Complement coding augments the input vector *I*, with a complement vector I^c such that $I^c = 1 - I$ (Nguyen et al., 2008). As a result, they are both presented together in both the input and weight vectors of the model. This is defined as a Boolean-value *compl* in the code. Upon setting *compl* to True, a vector *vcompl* is generated that is the complement of the vector *val*. The values of the *vcompl* vector are concatenated with the *val* vector in the *weights* vector.

- Parameters: The Fusion ART code contains several parameters that determine the behaviour of the model, such as how strict should the model accept patterns from inputs. For all *I*, there exists choice parameters *alpha* > 0, learning rate parameters *beta* ∈ [0, 1], contribution parameters *gamma* ∈ [0, 1] and vigilance parameters *rho* ∈ [0, 1] (Tan, Carpenter & Grossberg, 2007).

The ART architecture also introduces a reset mechanism, that checks the similarity between the weight vector and the input vector. The reset mechanism takes effect during the vigilance test and decides whether a node in *F2* learns the input pattern. A reset occurs when the level of matching of a node in *F2* is smaller than the defined vigilance parameter. The reset mechanism would then call the next sequential node in the *F2* layer for the vigilance test (Ignizio Burke, 1991).

3.2. Experimentation and Discussion

We would like to assess the effectiveness of the Fusion ART model in storing and retrieving data. For this experiment, the dataset "SCSE ProfProfile" is used that contains 83 professors, with their group, universities, and research interests. Before developing the Fusion ART model, several data pre-processing methods are implemented:

1. Each attribute of the dataset (name, group, university and research interest) is converted to separate Python lists.
2. The dataset contains a semicolon ";" that separates values of universities and research interests. Since each professor can have multiple universities and research interests, the universities and research interests of each professor are split into lists.
3. Since the model accepts numerical values, the current data could not be input to the model. The data is transformed from categorical data to numerical data using one-hot encoding and word embedding.
4. For each professor, their groups and universities are transformed into one-hot arrays that can contain multiple binary values through vector summation. Each element in the binary arrays represent a specific keyword. For example, if a professor belonged to three groups out of ten possible groups, their group vector may look like [0, 1, 0, 0, 1, 0, 0, 0, 1, 0].
5. For each professor, their research interests are transformed into distributed arrays through word embedding. In this experiment, there are 3 methods to represent research keyword values. These values can be realised using the Python library *gensim* by measuring similarities between keywords and by *gensim*'s word representation values. For dynamic word embedding, Google's Bidirectional Encoder Representations from Transformers (BERT) is used from the *Transformers* library that generates word vector values.
6. We call a list of stop words from the *NLTK* library and remove all stop words in the text corpora. These are a list of common words that do not contribute to the performance of the word embedding models.
7. The word embedding model is trained using various word corpuses. This includes corpuses that contain IT terminologies and Wikipedia articles. Next, each word is then cleaned by converting every letter into lower case and removing all punctuations. Moreover, since research keywords may comprise multiple words, or phrases, the word embedding model group common phrases from a sequence of words. Each professor then has their multiple research keywords vectors summed.
8. After generating the summed distributed vectors of each professor, their values range between -1 to 1. Since the Fusion ART model only accepts values between 0 to 1, the distributed vectors are normalised through min-max feature scaling as shown in Equation 1.

```
{'were', 'my', "wasn't", 'of', 'the', 'when', 'some', "shouldn't", "
couldn't", 'but', 'if', 'ma', 'his', 'hers', 'before', 'as', 'betwee
n', 'should', 'been', 'who', 'off', 'or', 'any', 'are', 'those', 'wi
th', 'then', 'too', 'is', 'down', 'while', 'in', 'and', 'how', 'by
',, 'o', 'into', 're', 'theirs', 'herself', 'this', 'until', "should
've", "haven't', 'after', 'that', 'yours', 'do', 'other', 'out', 'ag
ainst', 'own', "don't", 'during', 'just', "you've", 'ourselves', 'ha
ving', 'what', 'm', "weren't", 'ain', 'at', 'more', 'once', 'himself
', 'so', 'where', "mustn't", 'nor', 'me', "shan't", 'does', 'we', 't
o', 'its', 'being', 'both', "that'll", 'haven', 'has', "wouldn't"}
```

Figure 1: Examples of stop words found in the NLTK library.

$$x_i' = \frac{x_i - min(x)}{max(x) - min(x)}$$

Equation 1: Formula for [0, 1] min-max feature scaling of vector x. x_i represents the original value and x_i' represents the new normalised value.

Furthermore, n-grams are developed for all research keywords. N-grams are sets of string slices that form part of a longer string (Cavnar and Trenkle, 1994). They are concatenated and cleaned through the following processes:

1. Every letter is transformed into lower cases.
2. Every punctuation is removed.
3. For each keyword, their words are concatenated through an underline between words (“_”).

Thus, the research keyword “Agent Oriented Software Engineering” would be composed of a 4-gram and cleaned into “agent_oriented_software_engineering”. As a result, Python would classify this 4-gram as one word. The n-grams are placed between the words that form the n-grams, so that the word embedding model would consider the words similar to each other (See Appendix A and B).

Next, the schema for the Fusion ART model is defined as the list of unique names, groups, universities and research interests sorted alphabetically. This will ensure that the one-hot vectors associated with each attribute correspond to the values in the schema since the one-hot vectors are also sorted alphabetically.

name	*compl*	*attrib*
Name	False	[“AS Madhukumar”, “Alexei Sourin”, ...]
Group	False	[“Biomedical Informatics”, “Computational Intelligence”, ...]
University	False	[“Anna University”, “Birla Institute of Technology”, ...]
Research Interests	True	[“ad_hoc_and_mobile_networks”, “agent_oriented_software_engineering”, ...]

Table 1: Input schema of the professor model

The research interests attribute is complement-coded because we would like to use research interests as a query to retrieve professor data. More information regarding querying is explained below.

Suppose we initialise the Fusion ART model with the following parameters:

alpha	beta	gamma	rho
[0.1, 0.1, 0.1, 0.1]	[1, 1, 1, 1]	[0.25, 0.25, 0.25, 0.25]	[1, 1, 1, 1]

Table 2: Parameters of the professor model

Next, the model is updated by inserting each professor's full data, including their names, groups, universities, and research interests using a for loop. This is done by calling the function *updateF1bySchema* and inserting their respective distributed vectors. By performing resonance search for each node, the network can grow to 83 committed nodes, with each node containing each professor's full data along with one uncommitted node (Nguyen et al., 2008). This brings the network to a total size of 84 nodes.

As a background, the code aims to select a node in the *F2* layer that has a high level of matching against the input. Each node in *F2* is assigned a matching value and the node's index that has the highest matching value will be selected for learning. This process is called resonance search and is defined as the function *resSearch()* in the Python code (Tan et al., 2019).

Once the model has successfully stored the data, we would like to test the model whether it can retrieve data using an input query. Given an input of a list of research interests, we would like to convert the keywords into the form of an input vector to be read by the model. The aim is to retrieve the professor that best matches the list of research interests. This is called query by research keywords.

To query by research keywords, a batch file is created that contains the list of professor names and their research keywords, separated by a tab. The batch file is named "research_query.bat". For instance, the first five lines of the batch file are:

```
Alexei Sourin	Computer Graphics;Shape Modelling;Virtual
Reality;Web Visualization;Visualization on the
Grid;Cybermedicine;Scientific Visualization

Anupam Chattopadhyay		High level Synthesis;Application
specific Processors;Heterogeneous MPSoC;Synthesis for Emerging
Technology

Anwitaman Datta		Distributed Systems;Security and
Privacy;Data Analytics;Algorithms;Self organization

Arijit Khan		Graphs Querying and Mining;Databases;Data
Mining;Algorithms;Machine Learning

Arvind Easwaran		Real time Systems;Cyber physical
Systems;Formal Methods
```

Figure 2: Sample lines of the query by research keywords batch file

For each professor, their research keywords are used as input for the query. For this query, the model's gamma parameters are set to [0, 0, 0, 1] to solely account for research interests. The rho parameters are set to [0, 0, 0, 0] since we do not require vigilance during the query retrieval process. The research interests are converted into an input vector. The *compChoice* function retrieves the node with the highest F2 value.

In this experiment, noise is introduced with varying probabilities from $N = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. With a probability of N%, each value of the input research keyword vector is flipped to its complement, called bit flipping. For instance, a value of 0.2 has a probability of N% to change to 0.8, and so on. Python generates a random real value M ranging from 0 to 1 for each index. If M is lower than N, the value for that index is flipped.

$$X_{noisy} \begin{cases} 1 - X_{original}, & if\ M < N \\ X_{original}, & otherwise \end{cases}$$

Equation 2: Formula for bit flipping. The bit $X_{original}$ is flipped to its complement, X_{noisy} with a probability of N%.

The accuracy of the model is determined by the number of queries where the output of the query matches the name stated in the batch file. Therefore, it is expected that 0% noise would yield 100% retrieval accuracy. Since the probability of bit toggling is random, the model's accuracy differs for each attempt. Hence, the experiment concerns 5 attempts. For each attempt and level of noise, a text file is output that includes the research keywords and the retrieved professor name(s). The sample mean ($\bar{X}$) and standard deviation (s) is calculated for each level of noise.

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Equation 3: Formula for sample mean. x_i denotes the retrieval accuracy for each attempt and n signifies the number of attempts.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n - 1}}$$

Equation 4: Formula for sample standard deviation. x_i denotes the retrieval accuracy for each attempt and n signifies the number of attempts.

There are three methods to represent the research keywords into distributed vectors. These methods produce different query results.

3.2.1. Distributed Representation Using Similarity

The first experiment concerns using a measure of similarity between research keywords to generate the keywords' distributed vectors. The *gensim* library provides a similarity function that calculates the similarity score between one research keyword to another, using cosine similarity as a measure. The similarity score ranges between -1 to 1, where 1 means the research keywords are essentially the same in meaning, 0 denoting no similarity and -1 denoting opposite in similarity. The similarity scores comprise real values.

For each 261 unique research keywords processed from the dataset, the similarity score to each other research keywords are calculated. This is done using a for loop and put into a Python dictionary, called weight dictionary, that provides distributed vector values for each keyword. The vectors contain the similarity scores of one research keyword to another, with each element corresponding to a research keyword sorted alphabetically. For instance, the first element of a particular keyword vector corresponds to the similarity score between that keyword and "ad_hoc_and_mobile_networks", and so on. This implies that the number of dimensions *D* of the research keyword vectors is equal to the number of unique research keywords.

Next, the research keywords of each professor from the dataset are converted to distributed vectors with respect to the weight dictionary. Since each professor may follow multiple research keywords, the vectors are added using element-wise summation. Then, the values of the summed research keywords are normalised using the min-max scaling formula in Equation 1. The normalised values range between 0 and 1.

```
[0.21093685870034565, 0.9049768184605252, 0.4324959485003487, 0.6278747
285712776, 0.4236468240359887, 0.2839650668366084, …]
```

Figure 3: Normalised research keyword vector for Professor Alexei Sourin using similarity. The value 0.21 represents how similar the professor's research interests match "ad_hoc_and network_analysis". The value 0.90 represents "agent_oriented_software_engineering", and so on.

The experiment is separated into two trials. The first trial concerns querying by research keywords without complement coding. This means that the compl value for Research Interests when building the Fusion ART model is set to False. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	100%, 100%, 100%, 100%, 100%	100%	0
20%	96.39%, 96.39%, 93.98%, 93.98%, 93.98%	94.94%	1.32
30%	49.4%, 45.78%, 55.42%, 45.78%, 45.78%	48.43%	4.21
40%	12.05%, 15.66%, 12.05%, 15.66%, 21.69%	15.42%	3.94
50%	9.64%, 7.23%, 7.23%, 13.25%, 10.84%	9.64%	2.55

Table 3: Query by research keywords results for the similarity model without complement coding.

There is a notable drop of accuracy when the noise level is set beyond 20%. There also seems to be a high amount of variability in the accuracy for 30% and 40% noise, as shown with their high value of sample standard deviation.

The second trial uses complement coding to query by research interests. We can implement complement coding by setting the *compl* value to True, which generates a *vcompl* vector. By inserting both *val* and *vcompl* vectors to *updateF1bySchema*, the model can query using complement coding. For this trial, to prevent the bias of retrieving queries with very high accuracy, noise is implemented after the complement-coded vector is initialised. This means that if the *val* vector became subject to noise, its *vcompl* vector equivalent will as well.

By default, *updateF1bySchema* automatically sets *vcompl* to be the complement of *val*. By setting *refresh* = False in *updateF1bySchema*, the *vcompl* vector can be manually set, since the vector has undergone bit toggling. This is referred to as the “don’t-care” condition. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	100%, 100%, 100%, 100%, 100%	100%	0
20%	100%, 100%, 100%, 100%, 100%	100%	0
30%	100%, 100%, 100%, 100%, 100%	100%	0
40%	100%, 100%, 100%, 100%, 100%	100%	0
50%	85.54%, 81.93%, 83.13%, 81.93%, 77.11%	81.93%	3.07

Table 4: Query by research keywords results for the similarity model with complement coding.

Interestingly, the accuracy increases significantly compared to its non-complement coding counterpart. The model was able to retrieve the correct professor 100% of the time despite noise, with a bit toggling probability of up to 40%. Only when 50% noise is introduced the model begins to drop in retrieval accuracy. In order to further assess the performance of this model, higher levels of noise of up to 100% are introduced. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
60%	19.28%, 20.48%, 15.66%, 19.28%, 20.48%	19.04%	1.98
70%	2.41%, 3.61%, 4.82%, 2.41%, 4.82%	3.61%	1.21
80%	1.2%, 0%, 0%, 0%, 0%	0.24%	0.54
90%	0%, 0%, 0%, 0%, 0%	0%	0
100%	0%, 0%, 0%, 0%, 0%	0%	0

Table 5: Query by research keywords results for the similarity model with complement coding, up to 100% noise.

It would seem that the accuracy dropped tremendously beyond 50% noise, with a mean accuracy of 19.04% for 60% noise. Noise of 80% and above would disrupt the query such that it is almost impossible to determine the correct professor. This is because most of the values in the input query vector are inverted and differs greatly from the expected research keyword vector.

3.2.2. Distributed Representation Using Word Value

The second experiment concerns using an intrinsic distributed vector representation of each word that is generated by the word embedding model. The *gensim* model provides a function that outputs the value of a keyword in distributed vector form, with values comprised of real numbers. Like the similarity score discussed previously, the values range between -1 and 1.

The *gensim* model outputs the vectors of all 261 research keywords. The dimensions of the research keyword vector D can be adjusted when building the *gensim* model. In order to match the number of dimensions used during the similarity query, D equals the number of unique research keywords, which is 261. After the professors' research keywords are converted into distributed vectors, the professors' research keyword vectors are added using element-wise summation. Then, the values of the summed research keywords are normalised using min-max scaling as shown in Equation 1.

```
[0.5139246405702792, 0.7242976226114285, 0.4955219755086069, 0.73643287
50487741, 0.0, 0.5927072880885027,…]
```

Figure 4: Normalised research keyword vector for Alexei Sourin using word value. The values are intrinsic and arbitrary, ranging between 0 and 1.

The first trial concerns querying without complement coding. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	96.39%, 93.98%, 96.39%, 93.98%, 96.39%	95.43%	1.32
20%	91.57%, 87.95%, 90.36%, 91.57%, 86.75%	89.64%	2.19
30%	80.72%, 73.49%, 79.52%, 79.52%, 77.11%	78.07%	2.88
40%	67.47%, 69.88%, 68.67%, 72.29%, 69.88%	69.64%	1.79
50%	51.81%, 51.81%, 51.81%, 46.99%, 50.6%	50.6%	2.09

Table 6: Query by research keywords results for the word value model without complement coding.

Compared to querying using similarity as shown in Table 3, we can notice a drop in accuracy even when small noise is implemented. However, the reduction in accuracy is not as significant when the query is subject to higher levels of noise.

For the second trial, we implement complement coding by setting *compl* to True. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	100%, 100%, 100%, 100%, 100%	100%	0
20%	100%, 100%, 100%, 100%, 100%	100%	0
30%	100%, 100%, 100%, 100%, 100%	100%	0
40%	100%, 100%, 100%, 100%, 100%	100%	0
50%	100%, 100%, 100%, 100%, 100%	100%	0

Table 7: Query by research keywords results for the word value model with complement coding.

The results are peculiar. The query was able to retrieve the correct professor name 100% of the time despite implementing noise levels of up to 50%. We would like to determine when does the query drop in accuracy by implementing higher levels of noise. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
60%	92.77%, 90.36%, 90.36%, 77.1%, 84.34%	86.99%	6.34
70%	16.87%, 16.87%, 20.48%, 19.28%, 19.28%	18.56%	1.62
80%	1.2%, 0%, 1.2%, 0%, 0%	0.48%	0.66
90%	0%, 0%, 0%, 0%, 0%	0%	0
100%	0%, 0%, 0%, 0%, 0%	0%	0

Table 8: Query by research keywords results for the word value model with complement coding, up to 100% noise.

We can see that the accuracy starts to drop at 60% noise. At 60% noise, their accuracies also vary greatly. Interestingly, there is a significant drop in accuracy when noises are implemented at 70%. The query becomes ineffective beyond 70% noise. As opposed to using similarity as shown in Tables 4 and 5, querying with word value representation seems to provide higher retrieval accuracy overall.

There are several plausible reasons why complement coding increases the retrieval accuracies for this method significantly. By modifying some parameters of the model, we can investigate the cause of the increase in accuracy.

Since the dimensions of the research keyword vectors D can be adjusted, one proposed solution is to reduce the value of D. Previously, D equals the number of unique research keywords. If D is set to 50. The results of the query retrieval become as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	100%, 100%, 100%, 100%, 100%	100%	0
20%	100%, 100%, 100%, 100%, 100%	100%	0
30%	100%, 100%, 100%, 100%, 100%	100%	0
40%	95.18%, 98.8%, 95.18%, 92.77%, 95.18%	95.42%	2.16
50%	69.88%, 71.08%, 73.5%, 77.11%, 65.06%	71.33%	4.46
60%	40.96%, 42.17%, 34.94%, 27.71%, 39.76%	37.11%	5.93
70%	2.41%, 9.64%, 8.43%, 4.82%, 9.64%	6.99%	3.23
80%	6.02%, 0%, 0%, 0%, 2.41%	1.69%	2.64
90%	0%, 0%, 0%, 0%, 1.2%	0.24%	0.54
100%	0%, 0%, 0%, 0%, 0%	0%	0

Table 9: Query by research keywords results for the word value model with complement coding, for $D = 50$.

It seems that there is already a drop in accuracy starting from 40% noise. The overall retrieval accuracy seems to have also decreased, compared to $D = 261$.

From here, it can be hypothesised that the value of the dimensions of the research keyword vector D is proportional to the overall retrieval accuracy. This is because the more dimensions a research keyword vector contain, the more values a professor's node has that can distinguish itself from other nodes. This would mean that as D increases, it becomes easier to differentiate one professor from another, based on their different vector values. In addition, complement coding doubles the number of dimensions of each professor's research keyword vector, significantly increasing the retrieval accuracy.

However, increasing the value of D has a particular drawback. There is a complexity trade-off that needs to be considered upon determining the value of D. Increasing the number of dimensions would increase computational complexity, increasing time and space required to perform both encoding and retrieval. This would become problematic for larger datasets and when handling big data, where the number of unique keywords is significantly higher than this dataset. Moreover, implementing high values for D would skew the query retrieval because distant between nodes would become sparse and dissimilar (Aggarwal, 2005). As such, more data and nodes would be required for high values of D to balance the query.

3.2.3. Distributed Representation Using Language Models

The third experiment uses a BERT language model to develop numerical vectors of words. Some additional Python libraries were used, including *Keras* and *TensorFlow* in order to generate the numerical vectors through their natural language processing feature.

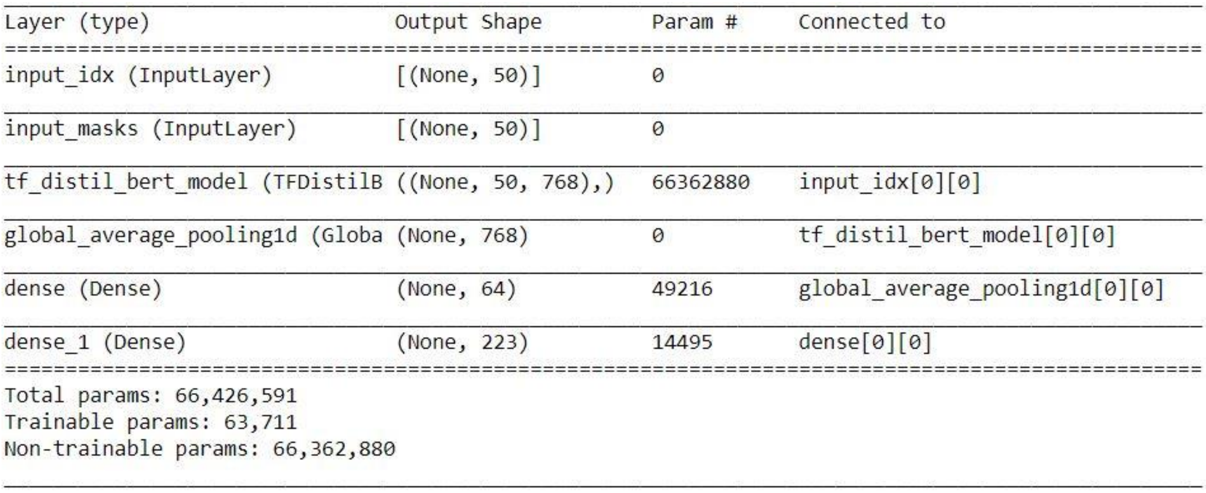
For each professor, their research keywords are concatenated and fed to the language model. The language model then returns a numerical vector. The numerical vectors are then normalised to the values between -1 and 1 before being fed to the Fusion ART model.

```
[0.43464731362481773, 0.4876900644347147, 0.45914761814290733, 0.489454
89220428423, 0.4639325217457486, 0.47321801598561913,…]
```

Figure 5: Normalised research keyword vector for Alexei Sourin using the BERT language model. The values range between 0 and 1.

In this experiment, a lighter version of BERT, called Distil-BERT is used, because the current dataset size does not require as many parameters. Distil-BERT takes less parameters for its word vectors than BERT. This results in a faster processing time during the development of the language model.

The BERT model outputs a feature matrix that contains 3 features: id, mask and segment. Using *TensorFlow*, we can generate a summary of the BERT model that highlights the number of trainable and non-trainable parameters upon feeding our corpora to the BERT model.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_idx (InputLayer)          [(None, 50)]         0
__________________________________________________________________________________________________
input_masks (InputLayer)        [(None, 50)]         0
__________________________________________________________________________________________________
tf_distil_bert_model (TFDistilB ((None, 50, 768),)   66362880    input_idx[0][0]
__________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 768)          0           tf_distil_bert_model[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 64)           49216       global_average_pooling1d[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 223)          14495       dense[0][0]
==================================================================================================
Total params: 66,426,591
Trainable params: 63,711
Non-trainable params: 66,362,880
__________________________________________________________________________________________________
```

Figure 6: BERT model summary for the professor dataset experiment.

Similar to the previous experiments, by feeding the numerical vectors of the Fusion ART model, the retrieval accuracy of varying levels of noise can be assessed.

The first trial concerns querying without complement coding. The results are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	32.53%, 28.92%, 33.73%, 33.73%, 32.53%	32.29%	1.98
20%	22.89%, 22.89%, 22.69%, 25.3%, 27.71%	24.3%	2.19
30%	16.87%, 16.87%, 19.28%, 16.87%, 19.28%	17.83%	1.32
40%	14.46%, 13.25%, 12.05%, 15.66%, 12.05%	13.49%	1.57
50%	12.05%, 12.05%, 12.05%, 12.05%, 12.05%	12.05%	0

Table 10: Query by research keywords result using the BERT language model without complement coding.

The accuracies seem rather low without complement coding, even when 10% noise is implemented. One plausible reason is because how close the values of the research keyword vectors are for each professor. As shown in Figure 5, the values of the normalised research keyword vector were all very close to 0.5. As other research keyword vectors share similar values, small changes in the values due to noise may return the wrong professor more frequently than other word embedding methods. Hence, complement coding may alleviate this problem, since the "don't-care" condition is able to generate a complemented vector that distinguishes each professor more against other professors.

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	100%, 100%, 100%, 100%, 100%	100%	0
20%	100%, 100%, 100%, 100%, 100%	100%	0
30%	100%, 100%, 100%, 100%, 100%	100%	0
40%	100%, 100%, 100%, 100%, 100%	100%	0
50%	92.77%, 93.98%, 96.93%, 91.57%, 97.59%	94.57%	2.61
60%	59.03%, 56.63%, 61.45%, 57.83%, 61.45%	59.28%	2.16
70%	27.71%, 26.51%, 24.1%, 25.3%, 27.71%	26.27%	1.57
80%	8.43%, 8.43%, 7.23%, 10.84%, 10.84%	9.15%	1.62
90%	0%, 0%, 0%, 0%, 0%	0%	0
100%	0%, 0%, 0%, 0%, 0%	0%	0

Table 11: Query by research keywords results using the BERT language model with complement coding.

The BERT model was able to perform accurate queries with complement coding. The decrease in accuracy was also not as significant as using word value, as the mean accuracy for 70% and 80% levels of noise are higher. As such, BERT may be the best model for querying with high levels of noise

3.2.4. Comparison and Discussion

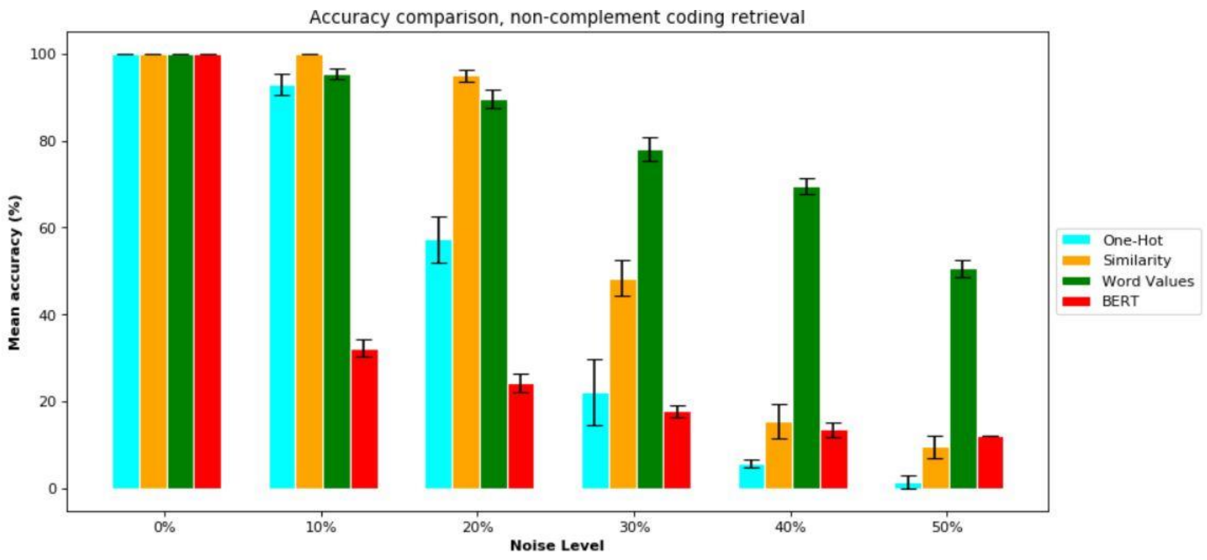
All three methods of using word embedding to encode and retrieve data has been proven more effective than binary encoding in terms of retrieval accuracy. By considering the distance and relationships between keywords, we can make more accurate predictions in finding the most similar professors with respect to our queries. As a comparison, the results of the binary encoding queries using one-hot from Tjahjadi's paper "Preliminary Experiments Regarding Fusion Adaptive Resonance Theory Through Binary Encoding" are as follows:

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	90.36%, 90.36%, 93.98%, 95.18%, 95.18%	93.01%	2.47
20%	63.86%, 51.81%, 59.04%, 51.81%, 60.24%	57.35%	5.36
30%	15.66%, 22.89%, 13.25%, 30.12%, 28.92%	22.17%	7.6
40%	4.82%, 6.02%, 4.82%, 7.23%, 6.02%	5.78%	1
50%	1.2%, 0%, 2.41%, 0%, 3.61%	1.44%	1.57

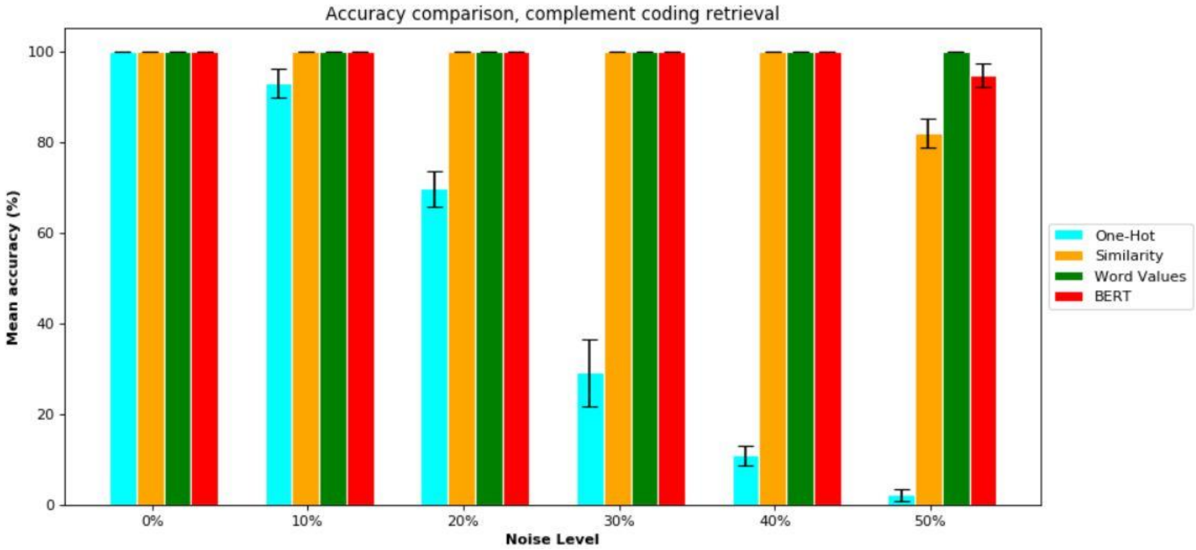
Table 12: Query by research keywords results using binary encoding without complement coding (Tjahjadi, 2020).

Noise	Retrieval Accuracy	$\bar{X}$	s
0%	100%	100%	0
10%	95.18%, 92.77%, 87.95%, 92.77%, 96.39%	93.01%	3.23
20%	74.7%,65.06%, 71.08%, 71.08%, 66.27%	69.64%	3.94
30%	19.27%, 27.71%, 31.33%, 39.76%, 27.71%	29.16%	7.4
40%	12.05%, 12.05%, 12.05%, 10.84%, 7.23%	10.84%	2.09
50%	3.61%, 1.2%, 3.61%, 1.2%, 1.2%	2.16%	1.32

Table 13: Query by research keywords results using binary encoding with complement coding (Tjahjadi, 2020).



Figures 7: Comparison of mean retrieval accuracy for using various encoding methods for non-complement coding.



Figures 8: Comparison of mean retrieval accuracy for using various encoding methods with complement coding.

Between similarity representation and word value representation, however, the latter has a higher retrieval accuracy overall. One plausible reason is how close the values of the research keyword vectors were before noise to 0.5. Since the bit flipping mechanism returns the complement of the value with probability N% as shown in Equation 2, keywords whose value are close to 0.5 suffers less noise. However, the number of dimensions D for word value representation contributes to its high retrieval accuracy as well.

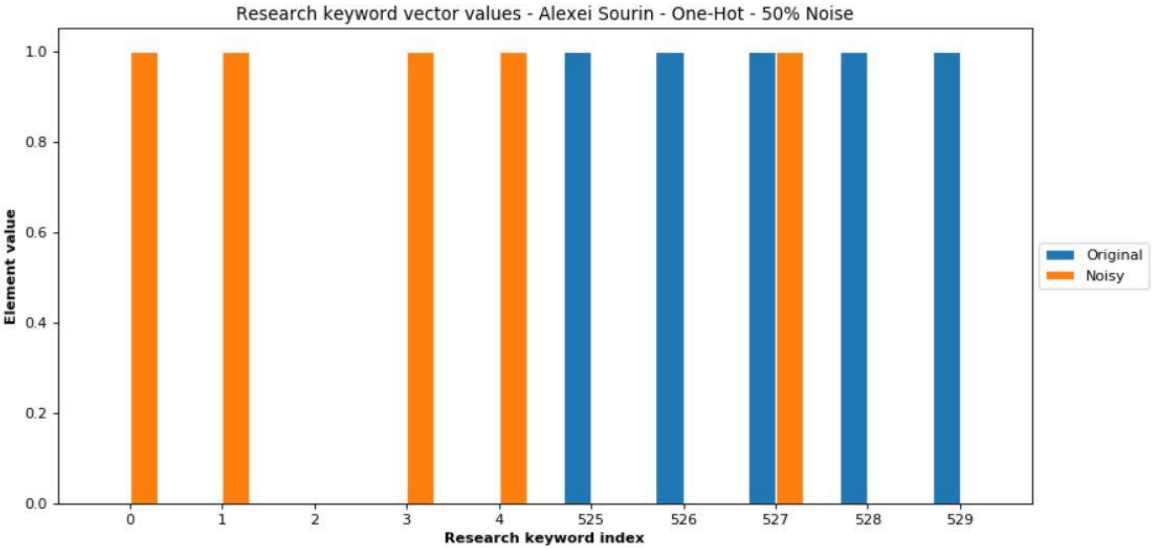


Figure 9: The differences in Alexei Sourin's research keyword values before and after noise for binary encoding with 50% noise and complement coding.

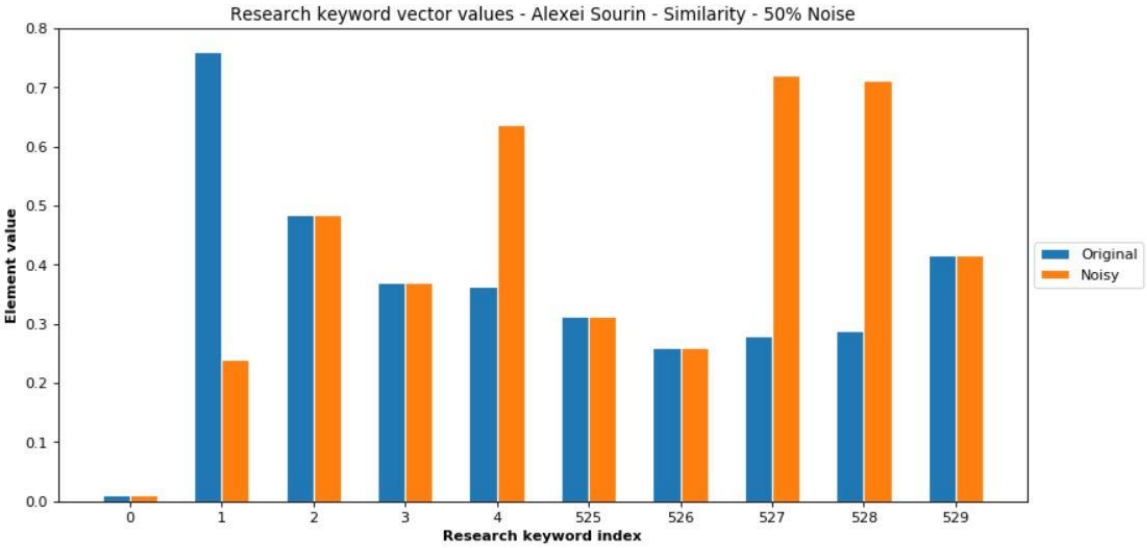


Figure 10: The differences in Alexei Sourin's research keyword values before and after noise for similarity representation with 50% noise and complement coding.

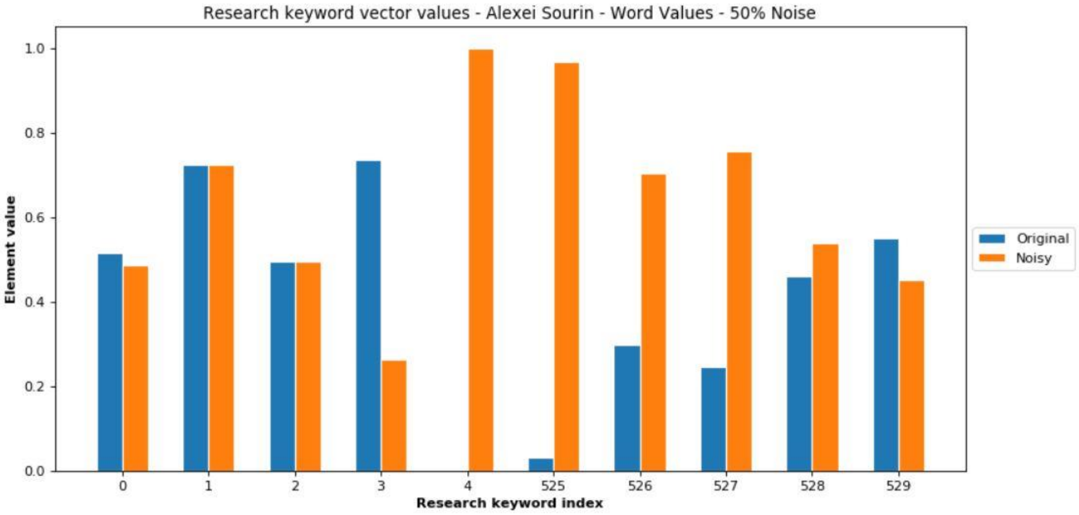


Figure 11: The differences in Alexei Sourin's research keyword values before and after noise for word value representation with 50% noise and complement coding.

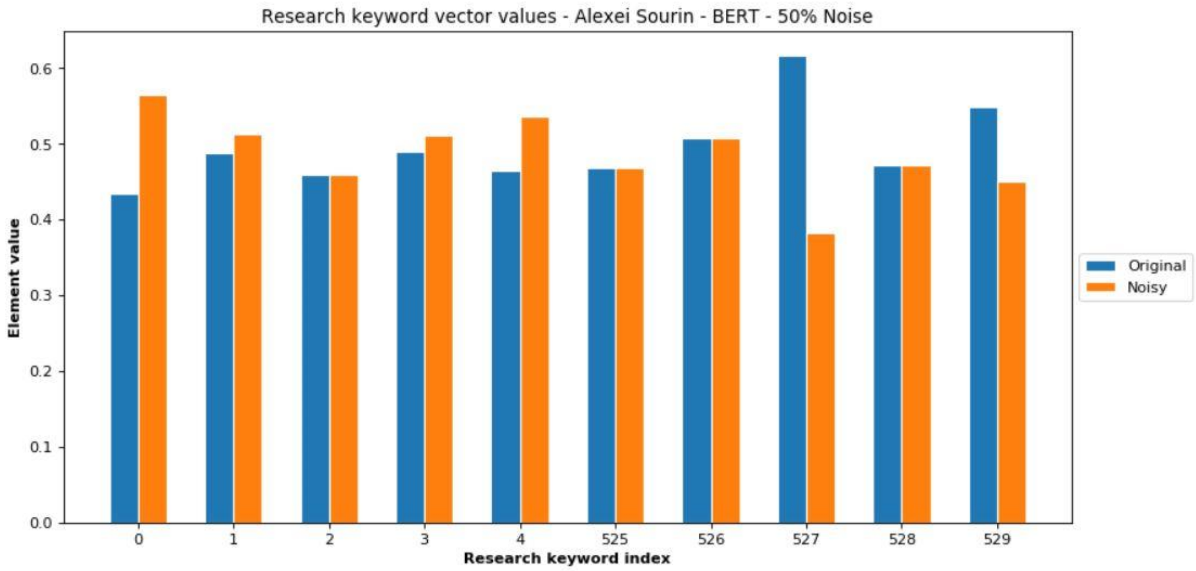


Figure 12: The differences in Alexei Sourin's research keyword values before and after noise for the BERT language model with 50% noise and complement coding.

Figures 9, 10, 11 and 12 indicate the differences between values of a research keyword vector before and after noise was implemented. The values are categorised as Original, indicating the values before implementing noise, and Noisy, indicating the values after noise was implemented. The first and last five research keyword vector indexes from complement coding are highlighted. Indexes that have the same original and noisy values indicate that bit flipping did not occur for those indexes.

Figures 9, 10, 11 and 12 may suggest why distributed representation using word embedding works more effectively than binary encoding. The differences in values between the original and noisy values differ greatly in binary encoding because the differences in bit flipping is always 1. For similarity and word value representation, the values do not differ greatly when bit flipping occur.

Moreover, the graphs indicate that the closer the values are to 0.5, the smaller the difference of those values when those values suffer bit flipping. This would lead to a smaller effect of noise, which may prompt the Fusion ART model to retrieve the correct professor name more often.

One drawback of having all research keyword values of each professor close to 0.5 is that one professor becomes indistinguishable from another. This can be seen from the huge decrease in accuracy for the BERT model without complement coding, as shown in Table 10 and Figure 5.

4. Reflections

4.1. Initial Expectations and Limitations

After several considerations for the experiments in Chapter 3, several issues were expected that did not match initial expectations.

4.1.1. Similarity Representation Experiment

One expectation during measuring similarity between words and research keywords is that, when using a function call to return the most similar words of a particular keyword, the word embedding model would output words that are expected from human knowledge. Although some research keywords produce expected outputs, some do not.

```
Top 5 similar words from augmented_reality:
[('augmented', 0.9627571105957031), ('virtual_reality', 0.8896215558052
063), ('reality', 0.8569343090057373), ('ar', 0.844650149345398), ('saf
ety', 0.8435107469558716)]

Top 5 similar words from enterprise_mobility_platform:
[('powersaving', 0.24785466492176056), ('startlingly', 0.24289645254611
97), ('subsequence', 0.23388811945915222), ('cursory', 0.23112191259860
992), ('dedication', 0.22258298099040985)]

Top 5 similar words from mathematical_analysis:
[('sentiment_analysis', 0.8253815174102783), ('parsing', 0.817862033843
9941), ('rigorous', 0.8139424324035645), ('geometry', 0.812340676784515
4), ('pattern_recognition', 0.8056570887565613)]

Top 5 similar words from visualisation:
[('popplewell', 0.8083217740058899), ('methodssocial', 0.80395841598510
74), ('drip', 0.799386203289032), ('spelling', 0.7948865294456482), ('c
ritique', 0.787341833114624)]

Top 5 similar words from visualization:
[('scientific_visualization', 0.8620719909667969), ('bioinformatics',
0.8353193402290344), ('engineeringontology', 0.8228553533554077), ('sci
entific', 0.8195556402206421), ('modelling', 0.8107743263244629)]
```

Figure 9: Top 5 similar words for "augmented_reality", "enterprise_mobility_platform", "mathematical_analysis", "visualisation" and "visualization".

Whilst the output for augmented reality, mathematical analysis and visualization were expected, the word embedding did not provide expected similar words for enterprise mobility platform and visualisation.

There are two separate reasons why this dissimilarity in measurement occurs. For enterprise mobility platform, there are not many text corpora for training the word embedding model that contains the words "enterprise", "mobility" and "platform", in order. As a result, the 3-gram "enterprise_mobility_platform" could not learn and its relationship between other words could not be measured.

For visualisation, the word embedding model was not able to capture the linguistic differences between regional spellings. As such "visualisation" and "visualization" are considered distinct words. This shows that the text corpora used to train the word embedding model seldom contain the word "visualisation". As a result, the word embedding model could not establish an accurate similarity measure for "visualisation".

Establishing relationships between words with similarity as a measure is an effective method to correspond each index of the research keyword vector to a research keyword. However, this signifies that the dimensions of the research keyword vector are equal to the number of unique research keywords. This can become computationally expensive for larger datasets with more keywords.

4.1.2. Word Value Representation Experiment

It is expected that using word value as a measure of word embedding would produce similar results to similarity representation. However, the model performs better in retrieving queries than its similarity representation counterparts. There are many plausible reasons for this to occur, such as how close the values are to 0.5 and how distinct the values of the research keywords are for each professor. When the values are close to 0.5, as discussed, the effects of noise become less significant, and the model would still be able to accurately predict the correct professor. When the values of the research keywords vector are distinct for each node, the nodes become less similar to each other. As a result, the F2 activation value would still be the highest for the correct node, and more noise would be required so that the model would mistakenly predict another node.

Another limitation of using word value representation is determining the number of dimensions of each research keyword vector D. Since there is a trade-off between accuracy and computational complexity with D, the values of D would depend on many factors:

1. How tolerant is the experiment with regards to handling model inaccuracy?
2. What hardware limitations are there for the computer that runs this experiment?
3. How much time is allocated to run this experiment?
4. How big is the dataset overall?

4.1.3. Language Model Experiment

The Distil-BERT language model was not able to query with high accuracy without complement coding, as shown in Table 10. The research keyword values may affect this decrease in accuracy, as each professor possess similar values. However, complement coding alleviates this problem and was able to query with higher accuracy in high levels of noise compared to other word embedding methods.

Using a language model that possesses more parameters may increase the overall retrieval accuracy. Nevertheless, a larger, more reliable dataset would be required to verify that more sophisticated language models produce more accurate results.

4.2. Suggested Improvements

Several improvements can be made in this experiment. For the word embedding model, they can predict and establish relationships between words more accurately by giving the model more training corpora. Particularly, providing text corpora that pertain to the information technology sector would benefit the word embedding model in understanding meaning and similarities for research keywords in the professor dataset. More training texts that possess such research keywords would yield more accurate relationships between words and query results. However, there should be a limit on how much corpora the word embedding model should train with. Too much data used for training would cause the model to overfit and yield high accuracy solely on training data. In contrast, too little training data would cause the model to underfit, such that the model did not learn enough to predict words accurately (Ghasemian et al., 2019).

Moreover, for the word value representation, optimising the number of dimensions of the research keyword vector *D* would provide a good balance between computational complexity and retrieval accuracy. More experiments can be done to test this value, and other model parameters, including the model's choice parameters *alpha*, learning rate parameters *beta*, contribution parameters *gamma* and vigilance parameters *rho*.

Experiments using the language model can be further improved using a language model that possesses more parameters to be considered for the *TensorFlow* library in generating research keyword vectors. As a result, the retrieval accuracy may increase for all levels of noise.

Finally, this experiment has only been limited to one dataset so far. Implementing this experiment and process through several other, larger datasets would prove beneficial in determining the viability of the model in storing and representing large volumes of data. The model is overall expected to behave similarly to this dataset in terms of word similarities and retrieval accuracy.

5. Conclusion

This paper has presented an introduction to how word embedding can be implemented to Fusion Adaptive Resonance Theory models, a type of neural network that learns from multiple pattern channels and machine learning paradigms in an online and incremental manner. Able to be used for a variety of applications, Fusion ART is a practical model for natural language processing and cognitive information processing. Several works of literature have reaffirmed the efficiency of Adaptive Resonance Theory for these applications (Grossberg, 2013) (Nguyen et al. 2008) (Tan et al., 2019).

The Fusion ART model was proven to be successful in storing and retrieving data. Using the professor dataset as an example, the model was able to store the details of each professor and retrieve those data using a query. Querying using a distributed representation of vectors can be performed through similarity representation, word value representation or language models. Regardless of the three methods, word embedding is an effective approach to determine the relationships between keywords, which leads to a higher retrieval accuracy as opposed to binary encoding overall. This is because determining relationships between words allows the Fusion ART model to predict professors which follow similar research interests. The Fusion ART API contains functions that allow such processes to work.

Several factors may affect the performance of the model, and more experiments can be done to determine the best-performing parameters for encoding and retrieval. This model can be used for a wider variety of datasets and applications.

References

Aggarwal, C. C. (2005). On k-anonymity and the curse of dimensionality. In VLDB (Vol. 5, pp. 901-909).

Carpenter, G., & Grossberg, S. (2002). Adaptive Resonance Theory. The Handbook Of Brain Theory And Neural Networks, 2.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. In Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval (Vol. 161175).

Ganguly, D., Roy, D., Mitra, M., & Jones, G. (2015). A Word Embedding based Generalized Language Model for Information Retrieval.

Ghasemian, A., Hosseinmardi, H., & Clauset, A. (2019). Evaluating Overfit and Underfit in Models of Network Community Structure. IEEE Transactions On Knowledge And Data Engineering. doi: 10.1109/tkde.2019.2911585

Grossberg, S. (2013). Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. Neural Networks, 37, 1-47. doi: 10.1016/j.neunet.2012.09.017

Ignizio Burke, L. (1991). Clustering characterization of adaptive resonance. Neural Networks, 4(4), 485-491. doi: 10.1016/0893-6080(91)90044-6

Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Advances in neural information processing systems (pp. 1081-1088).

Nguyen, L., Woon, K., & Tan, A. (2008). A Self-Organizing Neural Model for Multimedia Information Fusion.

Tan, A., Carpenter, G., & Grossberg, S. (2007). Intelligence Through Interaction: Towards a Unified Theory for Learning.

Tan, A., Subagdja, B., Wang, D., & Meng, L. (2019). Self-organizing neural networks for universal learning and multimodal memory encoding. Neural Networks, 120, 58-73. doi: 10.1016/j.neunet.2019.08.020

Tjahjadi, P. (2020). Preliminary Experiments Regarding Fusion Adaptive Resonance Theory Through Binary Encoding.

APPENDIX

```
['computer', 'computer_graphics', 'graphics', 'means', 'drawing', 'pict
ures', 'on', 'a', 'computer', 'screen', 'whats', 'so', 'good', 'about',
 'that', 'sketch', 'something', 'on', 'paper—a', 'man', 'or', 'a', 'hou
se—and', 'what', 'you', 'have', 'is', 'a', 'piece', 'of', 'analog', 'in
formation', 'the', 'thing', 'you', 'draw', 'is', 'a', 'likeness', 'or',
 'analogy', 'of', 'something', 'in', 'the', 'real', 'world', 'depending
', 'on', 'the', 'materials', 'you', 'use', 'changing', 'what', 'you', '
draw', 'can', 'be', 'easy', 'or', 'hard', 'you', 'can', 'erase', 'penci
l', 'or', 'charcoal', 'marks', 'easily', 'enough', 'and', 'you', 'can',
 'scrape', 'off', 'oil', 'paints', 'and', 'redo', 'them', 'with', 'no',
 'trouble', 'but', 'altering', 'watercolors', 'or', 'permanent', 'marke
rs', 'is', 'an', 'awful', 'lot', 'more', 'tricky', 'thats', 'the', 'won
der', 'of', 'art', 'of', 'course—it', 'captures', 'the', 'fresh', 'dash
', 'of', 'creativity—and', 'thats', 'exactly', 'what', 'we', 'love', 'a
bout', 'it', …]
```

Appendix A: List of words retrieved from a custom text corpus. The text has been cleaned, including converting all cases to lower case and removing all punctuations. N-grams have also been developed and placed between words that form the n-grams.

```
 ['computer', 'computer_graphics', 'graphics', 'is', 'the', 'branch', 'o
f', 'computer', 'science', 'that', 'deals', 'with', 'generating', 'imag
es', 'with', 'the', 'aid', 'of', 'computers', 'today', 'computer', 'com
puter_graphics', 'graphics', 'is', 'a', 'core', 'technology', 'in', 'di
gital', 'photography', 'film', 'video', 'games', 'cell', 'phone', 'and
', 'computer', 'displays', 'and', 'many', 'specialized', 'applications
', 'a', 'great', 'deal', 'of', 'specialized', 'hardware', 'and', 'softw
are', 'has', 'been', 'developed', 'with', 'the', 'displays', 'of', 'mos
t', 'devices', 'being', 'driven', 'by', 'computer', 'computer_graphics
', 'graphics', 'hardware', 'it', 'is', 'a', 'vast', 'and', 'recently',
'developed', 'area', 'of', 'computer', 'science', 'the', 'phrase', 'was
', 'coined', 'in', '0', 'by', 'computer', 'computer_graphics', 'graphic
s', 'researchers', 'verne', 'hudson', 'and', 'william', 'fetter', 'of',
 'boeing', 'it', 'is', 'often', 'abbreviated', 'as', 'cg', 'or', 'typic
ally', 'in', 'the', 'context', 'of', 'film', 'as', 'computer', 'generat
ed', 'imagery', 'cgi', 'some', 'topics', 'in', 'computer', 'computer_gr
aphics', 'graphics', 'include', 'user', 'interface', 'design', 'sprite
', 'graphics', 'rendering', 'ray', 'tracing', 'geometry', 'processing',
 'computer', 'computer_animation', 'animation', 'vector', 'graphics', '
d', 'modeling', 'shaders', 'gpu', 'design', 'implicit',…]
```

Appendix B: List of words retrieved from Wikipedia articles, starting from the first research keyword: computer graphics. The text has been cleaned, including converting all cases to lower case and removing all punctuations. N-grams have also been developed and placed between words that from the n-grams.