Y-X-Y Encoding for Identifying Types of Sentence Similarity

Miss Thanaporn Jinnovart

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science and
Information Technology
Department of Mathematics and Computer Science
FACULTY OF SCIENCE
Chulalongkorn University
Academic Year 2022

การเข้ารหัสวายเอ็กซ์วายสำหรับการระบุชนิดความคล้ายของประโยค

น.ส.ธนภรณ์ จิณโณวาท

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Thesis Title	Y-X-Y Encoding for Identifying Types of Sentence Similarity
By	Miss Thanaporn Jinnovart
Field of Study	Computer Science and Information Technology
Thesis Advisor	Professor CHIDCHANOK LURSINSAP, Ph.D.

Accepted by the FACULTY OF SCIENCE, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the FACULTY OF SCIENCE
(Professor POLKIT SANGVANICH, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Associate Professor SUPHAKANT PHIMOLTARES, Ph.D.)

..... Thesis Advisor
(Professor CHIDCHANOK LURSINSAP, Ph.D.)

..... External Examiner
(Dr. Prem Junsawang, Ph.D.)

ธนภรณ์ จิณโณวาท : การเข้ารหัสวายเอ็กซ์วายสำหรับการระบุชนิดความคล้ายของประโยค. (Y-X-Y Encoding for Identifying Types of Sentence Similarity) อ.ที่ปรึกษาหลัก : ศ. ดร.ชิดชนก เหลือสินทรัพย์

การหาความคล้ายคลึงระหว่างสองประโยคใดๆประกอบด้วยสองขั้นตอนหลักคือ การเข้ารหัสให้กับประโยคทั้งสองประโยคเพื่อสร้างเวกเตอร์ของคุณลักษณะที่มีความยาวเท่ากัน และการวัดความคล้ายคลึงระหว่างสองประโยคตามลำดับ คุณภาพของวิธีการเข้ารหัสสามารถกำหนดระดับของความสำเร็จของโมเดลในการวัดความคล้ายคลึงระหว่างสองประโยคได้ ทั้งนี้ก็เพราะการสร้างตัวแทนที่ดีขึ้นอยู่กับความละเอียดในการนิยามการแยกความคล้ายคลึง ยิ่งการนิยามความคล้ายคลึงชัดเจนมากเท่าใด การสร้างตัวแทนก็จะยิ่งดีขึ้นเท่านั้น ซึ่งจะช่วยในการแยกประเภทของความคล้ายคลึง โดยทั่วไปแล้ว ทุกวิธีที่มีอยู่สำหรับการวัดความคล้ายคลึงถูกออกแบบให้ข้อมูลในรูปแบบเวกเตอร์อยู่ในพื้นที่คุณลักษณะที่มีมิติตายตัว เพราะฉะนั้นการแปลงชุดประโยคที่มีความยาวต่างกันให้กลายเป็นชุดเวกเตอร์ของคุณลักษณะที่อยู่ในมิติเดียวกันเป็นเรื่องที่สำคัญมาก ชุดข้อมูลที่ใช้ในวิทยานิพนธ์นี้จัดเตรียมให้ทั้งค่าความเกี่ยวข้องเป็นตัวเลขและประเภทของความเกี่ยวข้อง ประเภทของความเกี่ยวข้องมีสามประเภท กล่าวคือ เป็นกลาง เกี่ยวข้อง และ ขัดแย้ง การแยกประเภทความเกี่ยวข้องบ่งบอกชนิดของความคล้ายคลึง นอกจากนี้โมเดลเข้ารหัสที่มีประสิทธิภาพสูงมักเรียนรู้ก่อนหน้าโดยใช้พารามิเตอร์จำนวนเป็นล้าน หรือแม้กระทั่งพันล้าน นี่คืออุปสรรคหนึ่งในการเทรนโมเดลเข้ารหัสด้วยตัวเองเนื่องจากจำเป็นต้องใช้ทรัพยากรที่มีความสามารถทางด้านการคำนวณมหาศาล

ในวิทยานิพนธ์นี้ เรานำเสนอวิธีการแปลงรหัสคำด้วยตนเองเพื่อจำแนกประเภทความเกี่ยวข้องออกเป็นสามประเภท ความเกี่ยวข้องของแต่ละคำในประโยคถูกจับได้อย่างพร้อมกันโดยโครงสร้างการเข้ารหัสด้วยตนเองนี้ นอกจากนี้ยังต่างจากโมเดลการเข้ารหัสอื่นๆที่ขึ้นอยู่กับการเรียนรู้แบบตามลำดับ เพราะโมเดลที่นำเสนอนี้ไม่ถูกรบกวนจากการสูญเสียความทรงจำที่เกิดจากความยาวของประโยค โครงสร้างของงานมีการคัดกรองคู่ประโยคที่ขัดแย้งออกจากชุดข้อมูลในขั้นตอนเบื้องต้นและใช้โมเดลเข้ารหัสจำนวนหนึ่งในขั้นตอนหลัง ซึ่งตัวเข้ารหัสแต่ละตัวจะอยู่ในรูปแบบ y-x-y โดยที่ y คือความยาวที่ได้จากการต่อสองประโยคเข้าด้วยกัน และ x คือความยาวที่เหมาะสมที่สุดสำหรับข้อมูลที่มีความยาว y นอกจากนี้โมเดลคัดแยกประเภทจำนวนหนึ่งยังถูกนำมาใช้แยกข้อมูลระหว่าง กลุ่มเป็นกลาง กับ กลุ่มเกี่ยวข้อง โดยให้ค่าออกมาเป็นความน่าจะเป็น ด้วยความแม่นยำกว่า 90% สำหรับการคัดแยกแต่ละประเภททั้งสามประเภท วิธีที่นำเสนอได้พิสูจน์แล้วว่าเราสามารถทำการแยกประเภทความเกี่ยวข้องได้อย่างมีประสิทธิภาพโดยไม่ต้องใช้ชุดข้อมูลที่มีขนาดใหญ่และทรัพยากรที่มีความสามารถทางด้านการคำนวณอย่างมหาศาล

สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อนิสิต .....
ปีการศึกษา	2565	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6278010123 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORD: Sentence Similarity

Thanaporn Jinnovart : Y-X-Y Encoding for Identifying Types of Sentence Similarity. Advisor: Prof. CHIDCHANOK LURSINSAP, Ph.D.

The task of finding semantic similarity of any two arbitrary sentences consists of two main steps, which are encoding sentences to produce feature vectors of equal length and measuring the similarity, respectively. The quality of an encoding technique can determine the degree of success a model can achieve in measuring the similarity. This is because a good representation is subjected to how finely established the spectrum of similarities is. The clearer the definition of similarity is, the better the representations can be constructed. This, in turn, helps distinguish between types of sentences. Generally, all existing methods for measuring similarity were designed for vectorized data in a feature space of fixed dimensions. Thus, transforming a set of various-length sentences into a set of feature vectors in the same dimension is very essential. The dataset used in this thesis provides both relatedness score and textual entailment. Textual entailment distinguishes sentence pair relations among three classes: namely, neutral, entailment and contradiction. The task indicates the types of entailments, which is interpreted as relatedness in this thesis. Additionally, powerful pretrained encoding models are usually of millions of parameters, or even billions. This is one obstacle in training one's own embedding model due to the need of resources with heavy computing capabilities.

In this thesis, we propose a self-encoding scheme to classify among the three classes of textual entailment. The relevancy of all words in a sentence is simultaneously captured by this self-encoding structure. Unlike the other encoding methods based on sequential learning, no interference of memory loss due to the length of sentence occurs in this approach. The framework involves filtering contradiction pairs at an early stage and employing a set of *y-x-y* encoders, where *y* is the length after two sentences are concatenated and *x* is the optimal encoding size for samples of length *y*, and classifiers to output neutral and entailment probabilities. With over 90% accuracy for all classes, our method has proven that this task is possible to be carried out effectively without the need of large-scale datasets and heavy computational resources.

Field of Study:	Computer Science and Information Technology	Student's Signature .....
Academic Year:	2022	Advisor's Signature .....

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Professor Chidchanok Lursinsap for his continuous support throughout the journey of this thesis. The completion of this project is made possible owing to his patience and devotion on cultivating the best out of his student and raising her standard to the next level. His expert guidance extends over his academic experiences to his philosophy in other aspects of life. It has been an honour to work under his supervision.

Besides my advisor, I would like to extend my appreciation to my thesis examination committees, Professor Suphakant Phimoltares and Professor Prem Junsawang, for investing their valuable time in considering this prolonged work with great care and kind comments.

Finally, I would like to thank my family for their never-ending encouragement. They had been through all ups and downs with me as if they were walking the path themselves. This accomplishment is undoubtedly also theirs. I am sincerely blessed by their genuine positive energy. I would also like to add my friends to my gratitude list, for their moral support.

Thanaporn Jinnovart

TABLE OF CONTENTS

Page

Chapter 1

INTRODUCTION

Communication between humans and machines has been in the trend for decades. That is because we have yet to find convenient ways to send and receive requests such that average users without knowledge in fields of Computer Science and Artificial Intelligence can interact, control, and operate a machine in order to complete a task. While there is a plenty of room for other tasks in artificial intelligence and machine learning to be explored and developed, it is undeniably important for machines to understand what they are needed to accomplish given human commands. Hence, natural language processing (NLP) tasks can be seen as a frontline, or a gateway, to accessing numerous possibilities a machine is capable of doing.

Although the history of NLP has been introduced on the advent of machine translation in the seventeenth century, there remains several challenges that are still unresolved. These challenges mostly owe their existence to the properties of language. Generally speaking, a language inherits ambiguity owing to demographic and cultural influences, and specific events. The use of language evolves over time. Well-constructed phrases and sentences are usually loosely considered in conversations; however, the context is not lost because a common background knowledge is shared among the people. This reflects how ill-structured groups of words can still make perfect sense among groups of people. Even well-structured phrases and sentences have imprecise rules. Therefore, it is important for an NLP system to understand some world knowledge. To elaborate, "***this***" may implicitly be established by a group of people in the same conversation as referring to the same object, while a newcomer without any knowledge on the subject is likely to have very few, to none, knowledge on which object is being referred. Suppose this newcomer is an NLP system, "***this***" may never have been learnt to refer to the true object in the same conversation at all. This is an example of unmodeled variables where one is assumed to have known the context of an unmodeled representation. Ambiguity may happen at different levels. At word senses level, "***bank***" may refer to a financial institution or a land alongside a river. At part-of-speech level, "***look***" may be a noun or a verb. At syntactic structure level, "***A boy can see a man with***

a telescope" may refer to "***a boy with a telescope***" or "***a man with a telescope***". It is not only words and groups of words that accumulates multiple level challenges, but also acronyms that a specific group of people has established and can only understand. For example, Figure 1.1 shows a supposed sentence which contains intentional misspelled words and acronyms. In order to interpret the sentence, a system is required to contain some knowledge on the established acronyms, specific spellings, and world knowledge. The last is particularly for interpreting "Facebook" since this is a proper noun.

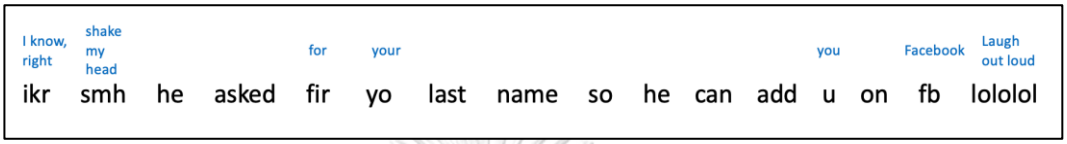


Figure 1.1 *Example of misspelled words and acronyms*

Even if a system is capable of supporting one language fully, it is not as easy to transfer its knowledge and capability to another language and hope that things will work exactly as well. A language exhibits multiple variations and expressivity. That is because each and every individual has one's own opinion and character. Such hindering factors occur in both popular languages, such as English and some other European languages, and thousands of moderate-to-poor-resourced languages which scales over 6,000 languages. According to Zipf's law, Figure 1.2a and 1.2b, the frequency of different words in a large text corpus illustrates an inverse proportion when drawn against rank. That is there are numerous infrequent, and probably important, words regardless of how large a corpus is. The challenge lies in a mean to estimate probabilities for unseen, and rarely seen, information. We may sum up the challenges into seven parts: namely, ambiguity, variation, expressivity, scale, sparsity, unmodeled variables and unmodeled representations.

any word

Frequency	Token
1,698,599	the
849,256	of
793,731	to
640,257	and
508,560	in
407,638	that
400,467	is
394,778	a
263,040	I

Figure 1.2a *Zipf's law*

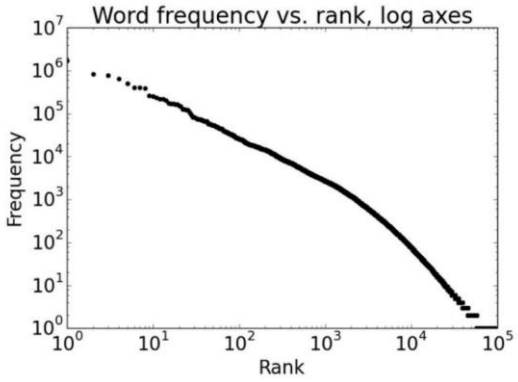


Figure 1.2b *Zipf's law*

Several core technologies, such as language modeling, part-of-speech tagging, syntactic parsing, named-entity recognition, word sense disambiguation and semantic role labeling, have been proposed and continually studied with one ultimate goal. That is to create a seamless natural human-to-computer communication for a machine to perform further advanced operations. Such applications may include machine translation, information retrieval, question answering, dialogue systems, information extraction, text summarization and sentiment analysis.

A conversational agent is made up of multiple capabilities; namely, speech recognition, language analysis, dialogue processing, information retrieval and text-to-speech. A typical conversational flow begins with an automated speech recognition (ASR) which detects and converts an utterance into text. The information is then passed along a natural language understanding engine where an intent is extracted, classified, and interpreted in order to be solved or fulfilled by a logical unit. This could be a business logic such as deducting or adding a certain amount of money out of or into an account. At this stage, the goal is to summon useful semantic information as much as possible. This is known as slot values. It is believed that more information will lead to more accurate responses. NLP researchers and developers are generally interested in developing language technologies that reside inside the engine. At the end of the journey, a response is generated and passed through a text-to-speech unit.

Information retrieval, text summarization, question-answering systems, and paraphrase detection are NLP tasks that require measuring the degree of similarity between texts, in particular, short sentences. For instance, in a question-answering system, a machine aims to find the most similar existing query in a database to the input query under an assumption that every existing query in the database contains at least one answer to it. Once a query is interpreted and a match with the database is found, a system can then respond with an appropriate answer. Pairing questions to answers, and vice versa, is also another part in which may require matching phrases, and sentences, based on similarity.

Determining sentence similarity usually consists of two essential steps. These are sentence encoding and similarity measure. A simple yet popular classical encoding scheme is

to reserve each bit for each word at each location as shown in Figure 1.3. The resulting matrix consists of bits set to 1 for words appearing in the sentence in one dimension and the location, or position, of the words in another dimension. In this example, the dictionary used for encoding the sentence only contains six words. In practice, there could be billions of possible words which would potentially result in a tremendously sparse and huge encoding matrix. This is considered inefficient as it would waste memory usage without giving much information and could easily cause memory leak due to its inability to scale over different lengths of sentences. In other words, the encoding matrix wholly depends on the size of the dictionary. This could lead to overfitting problem. Therefore, statistical-based direct encoding and pretrained encoding methods are two typical approaches in performing sentence encoding. Statistical-based encoding generally involves measuring the frequency of word occurrence in forms of n-grams [1], while pretrained encoding applies long short-term memory (LSTM) [2] and BERT [3], for instance.

The advent of the Transformers model [4] has raised the bar for many downstream NLP tasks, despite being originally designed as a Neural Machine Translation (NMT). This is also one of the fundamental elements used in BERT models which is designed to predict the next word. Its variants of pretrained models can be found prevalently in public for the community. However, many, if not all, are of hundreds and thousands of embedding dimensions. Perhaps, using powerful computational resources is the key to producing rich text representations.

It is an ant.

	it	they	is	ant	are	an
it	1	0	0	0	0	0
is	0	0	1	0	0	0
an	0	0	0	0	0	1
ant	0	0	0	1	0	0
sentence	1	0	1	1	0	1
location	0	1	2	3	4	5

Figure 1.3 *A classical encoding scheme*

1.1 Aims and Objectives

In this thesis, we are interested in finding the similarity between sentences. Our aim is to diverge from using huge computational resources to train complex models. A machine with ordinary computing capability should be able to replicate all stages associated with preparing and performing sentence similarity task presented in this work. Therefore, the objectives of this thesis can be summarised as follows.

1. To measure the degree of similarity between two short sentences into three classes, namely, contradiction, entailment, and neutral
2. To compare between two short sentences of different length with no maximum length limit using a proposed embedding technique
3. To evaluate the proposed model on existing benchmarks and widely known dataset SICK-2014 [5]

The reason for choosing to evaluate on entailment relation is that we believe the performance should better reflect a system's understanding of computational semantics at a more general level. Instead of semantic relatedness, a system would be able to classify whether high relatedness means contradiction or entailment while low relatedness would naturally fall into neutral.

1.2 Scope of Work

1. Sentences are taken from SICK-2014 [5]. These sentences are generally short and complete.
2. Sentences are embedded into vector representations. Regardless of sentence length, vector representation must result in fixed size.
3. All sentences being accounted for are in English due to data availability.

1.3 Contributions

In Chapter 3, we demonstrated our methodology in extracting features from input sentences and constructing vector representations. Here, we have solved the issue on how to handle variable-sized sentence pairs by manipulating them at input. This means that sentences do not have to be truncated or padded. They are accounted for their real lengths. In Chapter 4, we evaluated our proposed method against some baselines.

The approach proposed in this work avoids heavy pre-training of millions of parameters and of which to be trained on large scale datasets. The constraints imposed by unequal lengths of input sentence pairs are compensated such that the system does not require truncating sentences that are too long and padding those that are too short.

Chapter 2

LITERATURE REVIEW

According to [6], we may examine methods of measuring word similarity and sentence similarity separately. Word similarity methods include corpus-based, knowledge-based and string-based. Sentence similarity methods include word-based, structure-based and vector-based. The literature claims that finding similarity between words lies in the core of finding similarity between sentences. Therefore we consider methods at both levels in this chapter.

2.1 Word Similarity Methods

Corpus-based approach relies on big corpus analysis. It believes that semantically similar words would appear in similar manner; there is not much different in structure, or pattern, which similar words appear in. Word co-occurrence is also another key aspect which is used to indicate word similarity [7]. Two techniques used in analysing this approach are normal statistical analysis and deep learning techniques. The aim is to extract word semantic representation. For normal statistical analysis, this can be seen in Latent Semantic Analysis, or LSA. LSA's main objective is to calculate term frequency inverse document frequency and uses this as weights to the corresponding words. This is under an assumption that words that co-occur in the same context usually have similar meaning. In LSA, a word matrix is constructed where vectors of words and paragraphs are the rows and the columns respectively. Singular Value Decomposition is used to reduce the dimensionality. Words can be extracted and used in calculating their cosine similarity.

Deep learning techniques aim at producing word embeddings in a semantic space. Again, the word representation is based on co-occurrence of words in a corpus. In other words, a deep learning model is either trained to guess a word given its surrounding words, or vice versa. This is known as a Continuous Bag of Word model (CBOW) and Skip-gram model, respectively. Word vector representations generated from these two types of models are generally of size 200 to 400 based on the parameters set at training. These deep learning

techniques have shown promising results. The semantics of words can be captured in the vectors as their relations can be found via arithmetic summation and subtraction. The famous vector("King") – vector("Man") + vector("Woman") is closest to vector("Queen") example has successfully shown the semantic relations between the generated vector representations in Word2Vec model. Furthermore, this vector representation can be used in calculating similarity between words using cosine similarity as their semantic space coincides with our mathematical intuition. This had led to several attempts in employing high-quality word representation at sentence-level directly. However, a mere level transfer may not be an appropriate choice as sentences cannot be seen as just a group of words but their relations and structure should be accounted for as to interpret the true meanings. This may lead to a wider spectrum of NLP challenges, such as, syntactic structure, dependency parsing and semantic analysis.

Knowledge-based approach depends heavily on handcrafted knowledge semantic network. The network, ideally, contains both semantics of words and relations between words. The famous external resource WordNet is usually employed. This is a semantic network which consists of grouped synsets where each group shares a single meaning. This is also known as a group of synonyms. Hyponyms and hypernyms add a hierarchical structure to the network. With a well-documented API, word senses and relations can be conveniently extracted. One idea of measuring similarity of this network is to count the number of hops from one word to another. However, hops between words of different part-of-speeches may not be recognised, and therefore not possible, as synsets and other relations are usually of same part-of-speeches.

s1: acdeb s2: abcde

q = 2	$P_q(s1)$	$P_q(s2)$
ab	1	0
bc	1	0
cd	1	1
de	1	1
ac	0	1
eb	0	1

$$dist_{q\text{-}gram}(s1,s2) = \Sigma|P_q(s1)[i] - P_q(s2)[i]|$$
$$dist_{q\text{-}gram}(s1,s2) = 1 + 1 + 0 + 0 + 1 + 1 = 4$$

Figure 2.1 *q-gram distance*

String-based approach, or terminological approach, considers a word as a sequence of characters. Some methods include Levenshtein distance, q-gram and Jaccard distance. For Levenshtein distance, the aim is to transform one string to another with minimal operations. The operations are insert, delete and replace. The Levenshtein distance is the number of edit operations. In [8], the similarity measure is the ratio between the edit distance and the length of the longer word. Levenshtein distance algorithm can be seen very similar to sequence alignment algorithms such as Needleman Wunch algorithm and Smith Waterman algorithm. The aims for the edit distance algorithm and sequence alignment algorithm distinguish between the two. While the former minimises the number of transform operations, the latter maximises the similarity by assigning different weights to different types of operations. For q-gram distance in Figure 2.1, q signifies the length of substring to be compared. This is typically much smaller than the strings of the words. Each string is truncated into all possible combinations of length q. The occurrence count of each substring of length q is recorded. The q-gram distance is the sum of absolute difference between the occurrence counts of the two strings. While Levenshtein distance has $O(n^2)$ time complexity, q-gram only has O(n) time complexity.

Some combined methods have shown improved results. [9] employs both corpus-based and knowledge-based methods. The final similarity measure is an average of the two methods. In [10], Word Sense Disambiguation (WSD) first attaches every word in a corpus to a word sense. The word senses follow the assignments in WordNet and therefore shortest path distance is conveniently determined. The semantic relations in the corpus can even be found when exploiting the depth of words as WordNet holds information on the level of specificity. This is another cooperation between corpus analysis and a knowledge network.

2.2 Sentence Similarity Methods

Word-based similarity employs word-to-word similarity methods on words in sentences to calculate sentence similarity. This method treats a sentence as a set of words. One idea is that the similarity values between each word in sentence 1 and every word in sentence 2 are measured one at a time. For each word in sentence 1, it is compared against all words in sentence 2 and the average similarity is taken. This average similarity value denotes the word in sentence 1. Once all words in sentence 1 are measured, the sentence similarity is

taken by selecting the maximum average similarity value. This is known as max similarity. If all words in sentence 2 are also measured in the same manner, a similarity matrix can be constructed. With a sentence binary vector, sentence similarity can also be calculated as mentioned in [11]. Moreover, a semantic matching function is proposed to construct semantic matching vector for each word which can be used with the similarity matrix instead of a sentence binary vector. An example of a semantic matching function is the max similarity where the maximum similarity is the measure between each word in sentence 1 and every word in sentence 2. A modification of this semantic matching function is to consider only a window of words in sentence 2 when compared with each word in sentence 1. Each word is represented as a weighted average between the word and the window. Another matching function performs decomposition on generated match vectors to obtain similar and dissimilar parts for each vector to construct a similar matrix and a dissimilar matrix. From these two matrices, a sentence similarity can be computed, [12]. Using Word Sense Disambiguation (WSD) is another approach in measuring sentence similarity. [13] argues that, in small-sized text fragments such as sentences, word cooccurrence may not be the case for detecting two semantically similar sentences. This is because same keywords may not appear in both sentences; therefore, this approach employs word senses to identify synonyms by expanding WordNet synsets. A vector representation is formed for each sentence using word-to-word semantic similarity. This is done by either finding shortest path distance in WordNet or using Jiang and Conrath method in which the set of expanded words is considered in the formula. The sentence similarity is simply the cosine similarity between the two vectors. Some methods, [14], employ both WordNet and word embedding which produce better results than those using either stream. Despite the ease of implementation, word-based sentence similarity methods typically fail to capture structural information of given sentences. Some believe this could have been the key to interpreting the core semantic information. On the other hand, such approaches that see a sentence as a bag of words are appropriate for social media texts as they are usually poorly structured.

Structure-based similarity exploits sentence syntactic information. Many approaches believe that similar sentences are those that share similar structures. This is particularly for short texts. Examples of such methods utilise grammar, part-of-speech and order of words. [15] extracts grammar links between words in sentences to construct a grammar matrix in which the rows represent the links of the shorter sentence and the columns are the links of the longer sentence. Semantic similarity values between words with links of type are measured

based on WordNet ontology. The weighting strategy in [16] is done by assigning weights to words based on their part-of-speech (POS) and the relations between POS. This method gives unequal importance to the words in a sentence as it believes that certain POS and their relations are more important. Word order is usually used along with other syntactic information or other methods. The benefit of adding word order information is to prevent similar structures being falsely classified as having similar meanings at the first place. In other words, having similar structures may not convey similar meanings. For example, "***a woman kills a boy***" and "***a boy kills a woman***" share the same set of words but have different meanings. This is detected using word order. Nevertheless, it is reported in the survey that people usually use similar structures for sentences of similar semantics.

Vector-based similarity calculates similarity between sentence vectors. The key for this kind of approach is to come up with efficient and powerful sentence representations in which the features are concatenated into one vector. Obtaining such vector representations may be through statistical-based or learning-based methods. In a distributional method, a matrix of feature counts is constructed. Each row represents a sentence in a corpus and each column represents a feature in the sentences. Features could be n-grams or dependency pairs. A weighting schema is proposed to give weights to more important features. This is done by using the probability of existence of the feature in paraphrased and non-paraphrased sentences. After that, a sentence vector is extracted from matrix factorisation, and sentence similarity is measured, [17]. Another vector-based approach is to use an average of pre-trained word vectors in a sentence to obtain the vector representation of the sentence. Popular pre-trained word vectors are GloVe, [1], and Word2Vec [18]. Cosine similarity is, again, usually the choice for sentence similarity measure. This method does not account for word relations or structure of the sentence, despite manipulating vectors in a properly defined semantic space for word embeddings. [19] proposed a learning-based method in which regarding to the word embedding techniques used, a training model, which is a modified Long Short-Term Memory (LSTM) model, adjusts itself to learn to find sentence similarity. The model is known as Manhattan LSTM (MaLSTM). The last hidden layer is used as the representation for the input sentence. This representation is then used to learn the semantical similarity measure.

Skip-Thought vectors are obtained from training an encoder-decoder model. This is essentially an unsupervised learning. The recurrent neural network (RNN) encoder aims at mapping words to sentence vectors and the RNN decoder aims at generating surrounding sentences. This is similar to training for Word2Vec, in particular, the skip-gram model. A linear mapping solves the difficulty in setting up unseen words in the encoder's vocabulary space. This is because any word in Word2Vec can be mapped to the encoder's vocabulary space. The training phase requires a large collection of text from books. One reason for doing so is to reassure that the model is unbiased towards any domain. This vector representation is said to accurately capture both semantics and syntax. A similarity learning task is then carried out.

Although deep learning methods have shown good results, there is still room for improvement. Hybrid methods for measuring sentence similarity may show more promising results in many cases as each method treats the problems from different perspectives.

2.3 Word Embeddings

In [20], the survey explores methods of word embeddings which are of fixed-length, dense and distributed representations for words. This work acknowledges the importance of word embeddings as it mentions representations for words and documents prevail in, if not all, most NLP tasks. These include chunking, question answering, parsing and sentiment analysis.

Vector representations are useful as they enable intuitive interpretation and useful arithmetic operations, such as, addition, subtraction and distance measures which are suited in manipulating machine learning strategies. An embedding layer was first introduced as a key feature in developing a neural network language model to mitigate the curse of dimensionality on language models and help generalisation. It was realised that this layer implicitly contains syntactic and semantic word relationships, [21]. This is shown through the work of Word2Vec model [22]. Another influential embedding technique emerged in [1] which leverages word-context matrices. This is the GloVe model. Nevertheless, both models are based on the assumption that words in similar contexts have the same meaning. This is the distributional hypothesis, which dates back to literature works including [23]. Hence word

embeddings can be said to be built upon word co-occurrence statistics based on the distributional hypothesis. Embedding models can be divided into prediction-based models and count-based models. Prediction-based models are those derived from neural network language models as they predict the next word given its context. Count-based models are matrix-based models which account for co-occurrence counts in global context. In order to study word embeddings, this survey suggests navigating through two utmost important topics, namely the vector space model and statistical language modelling. The former clearly aids in complex interpreting mathematical theories such as linear algebra and statistics and incorporating them into a range of machine learning methods regarding NLP tasks. The latter interestingly originates word embeddings as by-products upon attempts in producing more efficient and more accurate language modelling. It arguably has not been long since the topics in word embeddings are decoupled from the task of language models [24].

The Vector Space Model (VSM) can be seen back in the field of Information Retrieval in [25]. The encoding procedure represents each document in a collection as a t-dimensional vector. Each element in a vector can be binary or real number. It is a distinct term in the document which may be normalised by a weighting scheme such as TF-IDF. The aim is to accentuate the difference in information of each term. Calculation of similarity between document vectors can be done at this point. More information of leveraging the VSM and its suitable applications can be read in the survey of [26].

Statistical Language Modelling is a development of probabilistic models of distribution of words in the language. In the early days, it aimed at recognising words and phrases in noisy and fault channels. A language model is intended to predict the next word given the words consecutively preceding it, which is also known as context. Hence a full probabilistic model contains the likelihood of every word in the vocabulary.

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{t-1}), \qquad (1)$$

Equation 1 shows a mathematical formulation of an n-gram model with window size T, where w_t is the t-th word, w_t^T is a sequence from w_1 to w_T. Hence $P(w_t|w_1^{t-1})$ is the

probability of w_t appearing after a sequence from w_1 to w_{t-1}. The next word prediction is done via maximum likelihood estimation over all words in the vocabulary.

Prediction-based models began with projecting a raw word vector into the first layer of a neural language model. This by-product is simply called the embedding layer. [24] is claimed to be the first to design a model with an intent of learning embeddings only. The training strategy is similar to that of a language model. The objective is to predict a center word given both preceding and following words around the predicted word. They also trained the model with false and negative examples, in which the center word is replaced by a random word in the vocabulary, for the model to distinguish positive answers from false ones. In 2013, [18] introduced two models for learning word embeddings, namely, CBOW and skip-gram (SG) models. These models are log-linear models and use the two-step procedure for training. The two-step procedure was introduced in [27]; the first step is to train using a single word as a preceding context word, and the second step is to train using larger context word and embeddings found in the first step as initial embeddings. CBOW and SG models differ by loss functions. As mentioned earlier, CBOW model aims to predict a word given context words, whereas SG model aims to predict the surrounding context words given a center word. Two variants of CBOW and SG models are [18] where it uses hierarchical softmax layers and [28] where it uses negative sampling. A more recent prediction-based model is an improvement over the skip-gram model [28], also known as FastText, [29, 30]. Instead of word embeddings, FastText learns to model n-gram embeddings which can be used to form words. This is introduced under the belief that word parts may contain information in which can help generalise unseen words for languages that heavily rely on compositional word-building especially highly inflectional languages.

Count-based models leverage global co-occurrence of word-context in a corpus and are usually represented as word-context matrices [26]. In LSA [31], which is one of the earliest examples, the word-context matrix is a term-document matrix. This matrix is decomposed by singular value decomposition. Although in information retrieval, one would be more interested in document vectors, the rows of the factorised matrix can give word vectors. [32] proposed Hyperspace Analogue to Language (HAL). The idea is to calculate all word co-occurrences between target words and context words where each context word has a distance inversely proportional to the target word. The context window size is optimal at 8.

As there is no normalisation to the word co-occurrence counts, a disproportionate amount of very common words, such as, ***the***, tends to bring about erroneous results. Therefore, [33] suggests using conditional co-occurrence. This rather induces a question of how much more likely word 1 is to occur with word 2 than another word, instead of only how much likely word 1 is to occur with word 2. The results show positive improvements. Additionally, another interesting method [34] proposes a Hellinger PCA transformation on the word-context matrix. The well-known GloVe by [1] prompts the use of ratios of co-occurrences instead of raw word counts. They suggest this method is able to capture the true semantic information between pairs of words when trained to maximise the similarity of every pair on a log-linear model.

The word embeddings that have been mentioned so far are either traditional or static word embeddings. This means that regardless of the semantic meanings or the context of a text surrounding the target word, the word's embedding will always remain the same. They essentially aim at representing global word embeddings. However, we know that a word can have different meanings in different scenarios. Hence contextualised word embeddings have been introduced. This type of word embedding aims to grasp context-dependent representations of words. It is crucial that the models learn from a large-scale dataset so that they have as much information as possible in inferring different vector representations of different words. We may formulate contextualised embeddings as associated to a function of an entire input sequence instead of a direct one-to-one mapping. In other words, a one-to-one mapping may still occur to obtain a non-contextualised embedding in prior to applying an aggregation function to obtain a contextualised embedding. [3, 35] have shown state-of-the-art contextual embeddings pretrained on large-scale unlabelled corpora. Their performances are evident in many NLP tasks, such as, text classification, text summarization, and question-answering. Contextualised embeddings have shown more promising results as their methods are able to better capture sequence-level semantics than non-contextualised embeddings. The key feature in creating contextualised embeddings is choosing suitable aggregation functions. In general, pretraining methods for contextual embeddings can be done by either unsupervised learning via language modeling or supervised learning in machine translation or natural language inference.

Like on the previous survey [20], this survey [36] also stated that a typical method in learning distributed token embeddings is to learn via language modeling. It models a probability distribution where it learns to factorise the probability of sequence given a specific number of tokens. A traditional language model learns to predict a token given a sequence of contexts on the left of the target word. This is usually trained on large-scale unlabelled corpora using neural networks.

ELMO [35] is a bidirectional language model consisting of L-layer LSTM both forward and backward. This is to encode both left and right contexts, respectively. There are *N* hidden LSTM states at each layer j which, in turn, denote *N* representations given contexts from both directions ($\mathbf{h}_{1,j}$, $\mathbf{h}_{2,j}$, …, $\mathbf{h}_{N,j}$). When applied to downstream tasks, in prior to advancing to higher layers, a common practice is to concatenate global word representations found in the lowest layers of the supervised models to ELMo's context-dependent representations $\text{ELMo}_k^{\text{task}}$,

$$\text{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}, \qquad (2)$$

where γ^{task} is a task-specific constant and s_j^{task} is a layer-wise weight normalized by softmax at layer *j*. One observation from this combination of bidirectional LSTMs is that the overall model does not consider the interactions between the left and right contexts.

The GPT family [37] learns universal representations through unsupervised pre-training using a language model and then supervised fine-tuning. The language model is built on a Transformer architecture [4]. GPT is trained on over 7,000 books from various genres while GPT2 [38] creates a new dataset of millions of web pages. The Transformer architecture has shown its superiority over the precursor recurrent networks on capturing global dependencies on a range of sequence learning tasks such as machine translation and document generation. It is an encoder-decoder model with multiple self-attention heads. The decoder reads from left to right and therefore can only attend to the left context.

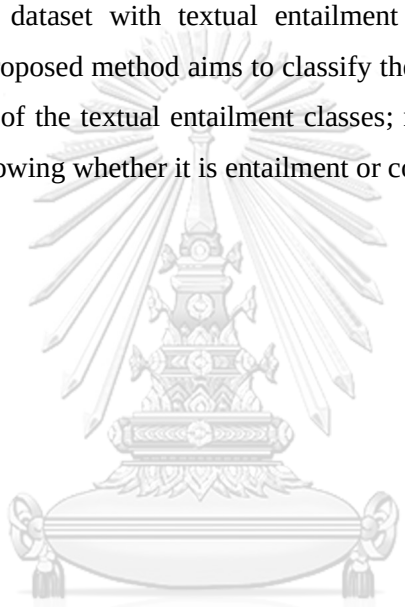
BERT [3] is trained on a masked language model. This means that some, typically 15 per cent, of the tokens in the input sentence are masked with random tokens. The objective is to predict the correct tokens given these masked tokens. The bidirectional training strategy also uses a Next Sentence Prediction (NSP) objective. This means that when a language model is given two input sentences, it is trained to predict whether the second sentence is the true following sentence of the first sentence. The purpose is to improve tasks such as question answering and natural language inference. BERT is trained on 3,300M words and it is critical that a document-level corpus is used in order to extract long contiguous sequences.

Both GPT and BERT use special tokens to obtain a single contiguous sequence for an input sentence. In BERT, the first token of an input sequence is always a special classification token, [CLS] token. This token holds information on the input sentence. It is a vector of size n where n is also the vector size of each of the remaining tokens in the input sequence. This [CLS] token can be extracted at the last layer of a BERT model via mean-pooling. This is one of the common practices in obtaining an embedding for an input sentence. Cosine similarity can be applied on vectors from [CLS] tokens between two sentences in order to find the sentence similarity. However, like GPT2, the pretrained language models do not provide any information on how they are able to improve on downstream tasks. Therefore, despite having performance improvement on several downstream tasks as per training on a huge amount of data, developers might still seek alternative methods that are interpretable in order to make their model sustainable in terms of maintenance and development.

In this work, we aim at seeking methods that can bypass the need of training on large-scale dataset and still able to capture information on given input sentence in our proposed word embeddings. The objective is to measure the similarity between sentences that are encoded using our proposed method. We acknowledge how powerful and useful the aforementioned models have contributed to the NLP community upon declaring improvements in several tasks, and we do not intend to question their supremacy. Instead, we are determined in pursuing an improvement in one simple task. That is to find an appropriate embedding for sentence similarity task. We believe that for one simple task, it is unnecessary to employ such huge amount of data in constructing a model. We also seek to mitigate the constraint such that two input sentences must be of equal length. In other words, we do not

wish to truncate sentences that are too long and pad sentences that are too short in order to fit them into the model.

SICK-2014 dataset was introduced during SemEval 2014 workshop [5]. It contains sentence pairs with scores on semantic relatedness and textual entailment. Semantic relatedness is a score rating between 0 and 5, where 0 indicates no relationship between two sentences and 5 indicates strong relatedness between two sentences. Textual entailment consists of three classes, namely, entailment, neutral and contradiction. In this work, we chose to work on SICK-2014 dataset with textual entailment labels as they can be used in classification task. Our proposed method aims to classify the type of relatedness between two given sentences into one of the textual entailment classes; rather than calculating the degree of relatedness without knowing whether it is entailment or contradiction.



Chapter 3

METHODOLOGY

The proposed method builds a system that can determine whether two input sentences are relevant or not by measuring their semantic similarity. We classify the types of similarity into ENTAILMENT, CONTRADICTION and NEUTRAL. There are two main consecutive procedures, which are (1) word and sentence encoding, and (2) classification, to determine the similarity. In this work, the classification problem is addressed in two steps. First, the set of words and their meanings are directly extracted without any complex computation, such as, neural learning. This is for separating contradiction types and non-contradiction types. To elaborate, if there exists a phrase containing a negative cue in one sentence but not in the other, given a pair of sentences, then this pair is identified as a contradiction type. Negative cues, or negative markers, imply potential opposition between two sentence inputs. This is further explained in section 3.3. Second, appropriate feature vectors are extracted and passed into a classifier. No contradiction cases should appear in an ideal scenario at this stage. This is where unequal length sentence pairs must be addressed. Although several powerful encoding schemes have been proposed, such as Word2Vec [18], bag-of-word [23], and mixture of BERT [3] and LSTM [2], their space and time complexities are rather high. This work proposes a shallow network encoding scheme in order to achieve lower space and time complexities.

Figure 3.1 illustrates the framework of the proposed method. The framework consists of five main stages. First, synonymous verbs and phrasal verbs are detected in both input sentences. This is to prepare for further detection and word translation in later stages. Second, any opposite words and negative markers are detected. If opposite words appear in both sentences and they refer to same entity, the sentence pair will be considered as a contradiction type and the process terminates. Otherwise, the sentences could be either neutral or entailment types. This proceeds to the next stage. The third stage translates all useful words into their corresponding numeric word codes. Useful words are those that remain from pre-processing. This stage is described in section 3.2. Once translated, the word vectors from both sentences are concatenated sequentially and become an input to the fourth stage. Here, a set of encoders is trained to obtain the vectors from the intermediate layers. These vectors are the encoded

vector representations with predefined lengths. This leads to the last stage which classifies these fixed length vectors into either neutral or entailment classes. More details are in the following sections.

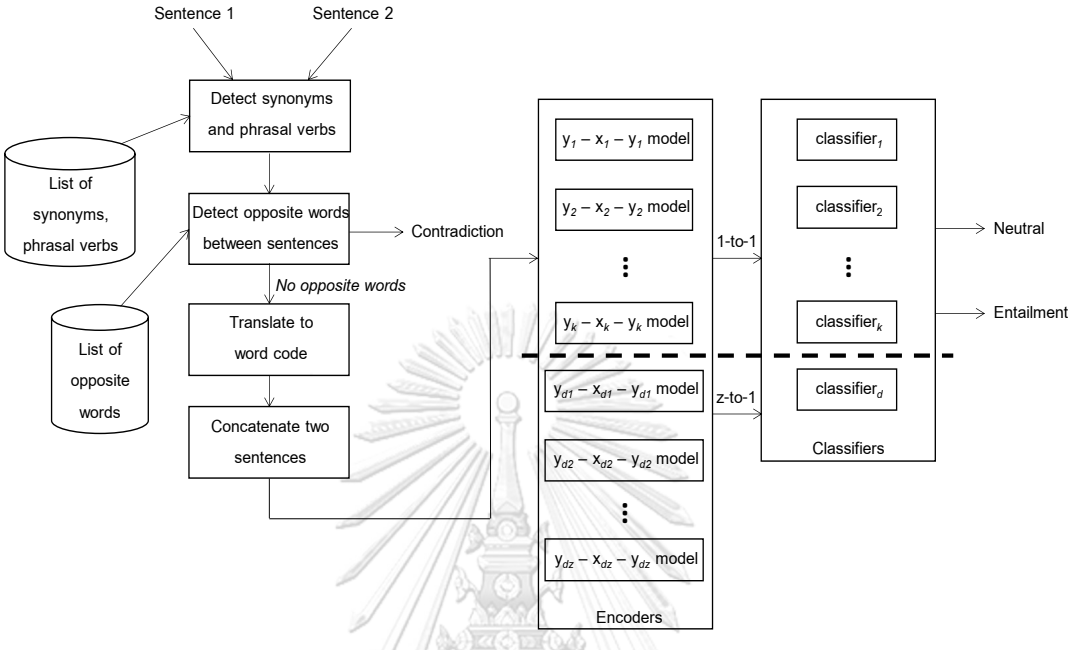


Figure 3.1 *Proposed framework. This consists of databases of synonyms, phrasal verbs and opposite words, and a set of encoders and classifiers. A set of k encoders are mapped onto a set of corresponding classifiers. A set of z encoders are mapped to a dedicated classifier. The input comprises two sentences. Contradiction samples are filtered out at early detection. Neutral and entailment samples are encoded and classified using simple network architecture.*

3.1 Input Sentences

Ideally, the input to the system is a complete sentence. A complete sentence consists of a subject, a predicate which is a verb, and an optional object or subject complement. In general, SICK-2014 dataset contains simple sentences. However, there are occasional compound, complex and compound-complex sentences too. Moreover, a complete sentence may be an existential sentence which begins with an expletive *there* or *it*. Some examples of simple, compound, complex, compound-complex, and existential sentences are the followings. The sentences are taken from https://style.mla.org/types-of-sentences/.

Simple sentence: Only a single independent clause appears.

The girl bought an ice cream cone.

The girl went to the park.

Compound sentence: Two or more independent clauses are joined by a coordinating conjunction (and, but, yet, for, or, nor, so), a conjunctive adverb (e.g. however, furthermore, likewise, rather, therefore), or a semicolon.

The girl bought an ice cream cone, but she dropped it in the park.

Complex sentence: One or more dependent clauses are connected to an independent clause by a subordinating conjunction (e.g. because, after, when) or relative pronouns (who, which, that).

After she bought an ice cream cone, the girl went to the park.

The girl, who had a freckled face and wore a striped shirt, was knocked over by a large dog, which ate her ice cream cone.

Compound-complex sentence: One or more dependent clauses are attached to one or more independent clauses.

After she bought an ice cream cone, the girl, who had a freckled face and wore a striped shirt, went to the park, but she was knocked over by a large dog, which ate her treat, so she ran home to her mother, who made her an ice cream sundae.

Existential sentence:

There is a girl buying an ice cream cone.

It is important to note that SICK-2014 dataset generally contains simple sentences. This means that any opposite words or synonymous verbs detected between any sentence pairs usually refer to the same entity. This fact enables our system to avoid containing a complex logic unit. In such scenario, the need of a sophisticated mechanism would be inevitable in pairing adjectival modifiers and verbal predicates to their nominal entities to ensure the counterparts are comparable. This would also apply to opposite nouns where the whole sentence would be required to be parsed in order to ensure the nouns are comparable.

3.2 Pre-processing

Before proceeding to the encoding procedure, a sentence is lemmatized by Stanford CoreNLP tool [39] and stop words are removed. Non-alphabetical characters and numerical characters are included in the set of stop words. Punctuations are also removed from the

sentence. The stop words play no roles in semantic encoding since they do not contain truly relevant information. Furthermore, all contractions such as ***isn't***, ***weren't*** are transformed into their original full forms to enable encoding. Phrasal verbs are also generalised to single words. A corresponding word code will be assigned to one of the words in the compound while the other will be discarded after the translation. Transforming multi-words into single ones enables simple word encoding to take place which also increases the chance of finding synonymous words between sentence pairs. This is because synonymous words will be mapped to same word codes during word code translation in later stage. Once pre-processed, the remaining words are generally content words; those that give meanings and appear as single individuals.

3.3 Filtering Contradiction Class

According to the nature of SICK-2014 dataset, we observed that sentence pairs of contradiction type generally contain either antonym pairs or negative markers. In the first scenario, this requires each of the two sentences to contain a word, or a phrase, that is opposite to another. In example 2 of Antonym pairs below, the opposite words, are ***near*** and ***far from***. The system recognises that both entries refer to the distance of the ***brown horse*** from ***red barrel***. This demonstrates that the proposed method considers the surrounding context in prior to deducing. Once an antonym pair is detected and confirmed to be referring to the same entity, the sample is then categorised as contradiction. If, for instance, an antonym pair exists but are seen to refer to different objects, the sample would not be deduced as contradiction. This is because the opposite words are likely to be in different context and hence the sentences would be irrelevant, which means a neutral type. As mentioned in section 3.1, SICK-2014 dataset mostly contains simple sentences. This fact allows the system to stop investigating for further complex relations that could change the decision of the system.

The second scenario is when there exists a negative marker in one sentence but not in the other. Negative markers include ***no***, ***none***, ***no one***, ***nobody***, ***there is no***, ***there are not***, and ***there have no***. Again, the system assumes that if only one out of the two sentences contains a negative marker and that it refers to an entity that also appears in the other sentence, the sample is contradiction. For example, in example 1 of Negative markers, the negative marker ***no*** in the first sentence refers to ***biker***. The second sentence also contains ***biker*** but does not have a negative marker attached to it. Therefore, the two sentences must be in contradiction.

Negative markers.

Example 1

- *There is no* biker jumping in the air.
- A lone biker is jumping in the air.

Example 2

- A deer is jumping over a fence.
- A deer *isn't* jumping over a fence.

Example 3

- Several people are in front of a colourful building.
- *Nobody* is in front of the colourful building.

Example 4

- Two people are kickboxing and spectators are *not* watching.
- Two people are kickboxing and spectators are watching.

Antonym pairs.

Example 1

- A man is jumping into an *empty* pool.
- A man is jumping into a *full* pool.

Example 2

- The brown horse is *near* a red barrel at the rodeo.
- The brown horse is *far* from a red barrel at the rodeo.

For each input sample, the system terminates if the sample is considered contradiction. After filtering the contradiction cases, the rest of the samples must be either entailment or neutral types.

3.4 Encoding Procedure

All words must be converted into their corresponding numerical values in order to proceed to the encoding procedure. Words with same meanings, or synonymous words, are assigned to the same values.

The main issue being addressed in the encoding procedure is handling two sentences of unequal lengths. A set of encoders is employed. All encoder models have the same structure which consists of an input layer, an intermediate layer and an output layer. The input layer and the output layer have the same size. Hence, an encoder model exhibits a network such as *y*-*x*-*y* where *y* is the size of the input sentences combined and *x* is the size of the intermediate layer. The goal is to construct a set of encoders whereby the intermediate layers can be used for binary classification; the classes are neutral and entailment. The purpose of having multiple encoders is to handle different lengths of input samples once they are concatenated. The whole encoding and classification stages, in turn, could be seen as creating a multiplexer-like model. The length of a concatenated input sample selects which encoder model the data should flow into in order to achieve the output signal indicating either a neutral or an entailment class. The one difference this proposed workflow has with a multiplexer is the two output signals instead of one. This is because the classifiers return a set of two values each corresponding to the probability of being neutral and entailment. The one with higher probability wins and sets the sample to that type.

Let sentence i, $\mathbf{S}^{(i)} = \left\{w_1^{(i)}, \cdots, w_m^{(i)}\right\}$, consist of a set of words $w_a^{(i)}$. Each word is translated into its corresponding numerical word code $c_a^{(i)}$. Hence the sentence is converted into $\mathbf{S}^{(i)} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}\right\}$. Each encoder accepts a certain size of input. The input to the encoder model is obtained by concatenating the vector representations of both sentences. Hence an input is represented as $\mathbf{I} = \left\{\mathbf{S}^{(i)}, \mathbf{S}^{(j)}\right\} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}, c_1^{(j)}, \cdots, c_n^{(j)}\right\}$ where m denotes the length of the sentence i and n denotes the length of the sentence j. At the end of an encoder, the original input vector is expected. This is the goal of all the encoders used in this work; to make the output identical to the input, although not perfectly so in practice. The intermediate layer is then extracted and used for classification.

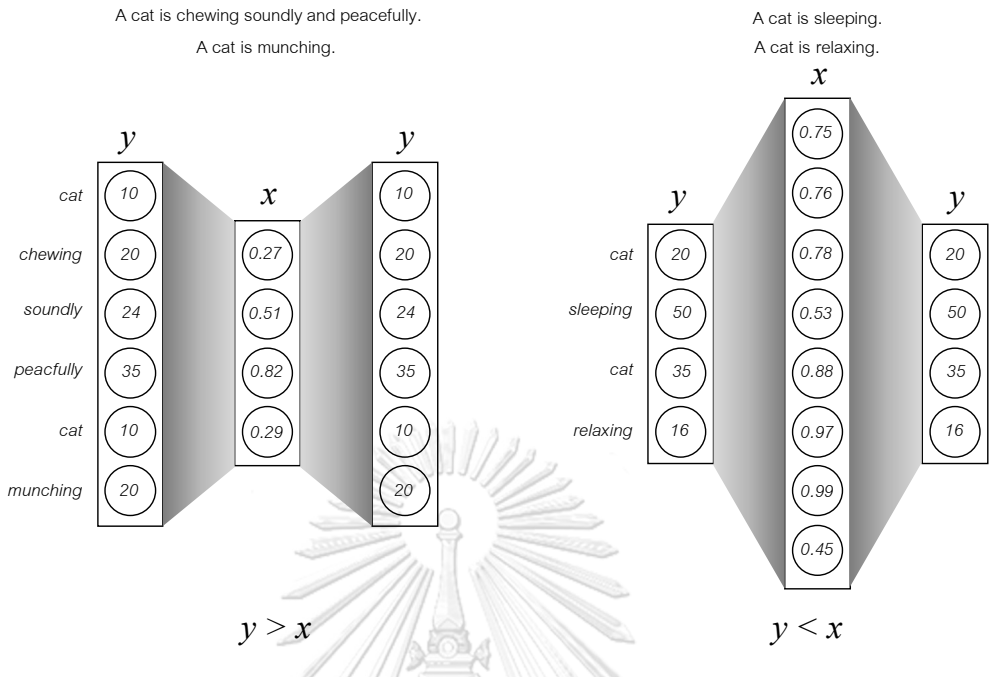


Figure 3.2 *y-x-y encoder model.The input size y, and hence output size, depends on the length of the two sentences after preprocessing. The intermediate layer size x can be greater than or less than the input size y.*

The *x* in the *y-x-y* configuration of the encoders is the size of the intermediate layer, or an encoding size. As shown in Figure 3.1, each encoder is adjusted to find the optimal encoding size for its input length *y*. The optimal encoding size is the size that gives the highest accuracy from mapping inputs to themselves. As shown in Figure 3.2, *x* may be greater than *y*, and vice versa. In an ideal situation, the input lengths, *y*s, should cover a wide range of possibilities. However, this is limited to the unique concatenated lengths of the preprocessed sentences in the dataset. The overall encoding and classification processes are summarised in Algorithm 1. The system is further constrained by the fact that the total unique lengths available in the dataset do not contain equal number of data entry. In other words, some *y*s may contain more samples than others. For those *y*s that contain too few samples, training a separate classifier for them is not feasible as the number of data entries is too low. Instead, the system has a dedicated classifier to accept samples whose concatenations result in the same length *x*. These samples are separately encoded to same length and accumulated. To simplify, x_{d1} to x_{dz} all have the same length. The common optimal encoding length is found to be 750. This is derived from majority voting among all corresponding *y*s. In Figure 3.1, this is

the y_d-x_d-y_d model. This model, in fact, is a group of encoders, with different y_ds, whose intermediate layers, x_d, are of the same size.

ALGORITHM 1: TRAINING ENCODING AND CLASSIFYING ALGORITHM

Input: a set of preprocessed sentence pairs $\mathbf{S}^{(i)} = \left\{w_1^{(i)}, \cdots, w_m^{(i)}\right\}, \mathbf{S}^{(j)} = \left\{w_1^{(j)}, \cdots, w_n^{(j)}\right\}$.

Output: a set of encoders and classifiers

1 Assign word code to each $w_a^{(i)}$ and $w_b^{(j)}$ to obtain $\mathbf{S}^{(i)} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}\right\}$ and $\mathbf{S}^{(j)} = \left\{c_1^{(j)}, \cdots, c_n^{(j)}\right\}$.

2 Concatenate $\mathbf{S}^{(i)}$ and $\mathbf{S}^{(j)}$ to obtain $\mathbf{I} = \left\{\mathbf{S}^{(i)}, \mathbf{S}^{(j)}\right\} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}, c_1^{(j)}, \cdots, c_n^{(j)}\right\}$.

3 For each y from **I**s with sufficient samples, vary x on training $y - x - y$ model to obtain separate optimal encoding sizes.

4 For each y from **I**s with few samples, vary x on training $y - x - y$ model to obtain one common optimal encoding size.

5 For each separate optimal encoding size, train separate classifiers.

6 For common optimal encoding size, train one classifier.

3.5 Classification

All classifiers used in the work are identical. Each classifier consists of a 100d and 300d fully connected layers, accordingly, and a Softmax layer. Both first and second layers

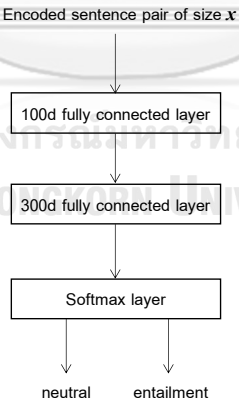


Figure 3.3 *All classifiers are identical.*

use sigmoid activation functions. All classifiers are optimised using stochastic gradient descent with learning rate of 0.001. The input size of a classifier is the size of the intermediate layer of its corresponding encoder. The final output gives two values of which sum up to 1. These are the probabilities of being neutral and entailment. The one with higher probability wins and the sample input is predicted as that class. The chance of having equal probability, which is 0.5 each, has never happened. Although presumably a very rare case, such

occurrence would result in a random assignment between neutral and entailment. After all, this is essentially simplified as a binary classification with an early removal of contradiction class. The classification stage is summarised in Algorithm 2.

As a continuation of the encoding procedure, another ideal situation is that for all *y*s, the optimal encoding size is the same. This would enable the classification stage to be simplified to be using only one classifier. This could increase the accuracy of the classification problem as the system would be able to focus on training one classifier with more useful data. In practice, this is not the case, as mentioned in section 5, which led the system to have one classifier per encoder as many as possible.

The purpose of using different encoders is believed to unveil characteristics of different lengths of concatenated sentence pairs. For example, a sentence pair concatenated to length 12 could give an optimal encoding size at 25 whereas length 11 could give size 250. This could be interpreted that the combined length 11 generally contains overlapping words that the encoding dimension is 10 times higher since a more complex set of weights is needed to distinguish between the classes.

The reason for having different classifiers instead of one after encoding is that different encoder inputs have different optimal encoding lengths. The proposed method uses these optimal encoding lengths to decide on the input lengths to the classifiers.

ALGORITHM 2: SENTENCE PAIR CLASSIFICATION ALGORITHM

Input: preprocessed sentences $\mathbf{S}^{(i)} = \left\{w_1^{(i)}, \cdots, w_m^{(i)}\right\}$, $\mathbf{S}^{(j)} = \left\{w_1^{(j)}, \cdots, w_n^{(j)}\right\}$.

Output: sentence pair type (*neutral, contradiction, or entailment*)

1 If found opposite words and/or one-sided negative marker, return *contradiction*.

2 Assign word code to each $w_a^{(i)}$ and $w_b^{(j)}$ to obtain $\mathbf{S}^{(i)} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}\right\}$ and $\mathbf{S}^{(j)} = \left\{c_1^{(j)}, \cdots, c_n^{(j)}\right\}$.

3 Concatenate $\mathbf{S}^{(i)}$ and $\mathbf{S}^{(j)}$ to obtain $\mathbf{I} = \left\{\mathbf{S}^{(i)}, \mathbf{S}^{(j)}\right\} = \left\{c_1^{(i)}, \cdots, c_m^{(i)}, c_1^{(j)}, \cdots, c_n^{(j)}\right\}$.

4 Encode **I** using optimal encoding size.

5 Calculate probability of being *neutral* and *contradiction* using corresponding classifier.

6 Select class whose probability is greater than the other; randomly select one upon equal probability, return *neutral* or *entailment.*

Chapter 4

RESULTS AND DISCUSSION

4.1 Results

In this chapter, we observe and report the results of our experiments. We evaluate the performance of our proposed methods using accuracy as complied with the existing models' evaluation in SemEval 2014 Task 1 [5]. We executed our model on three different sets as divided in the manner of SICK-2014 dataset, namely, training set, trial set and test set. The reason for doing so is solely to make our results comparable to other methods as proposed in SemEval 2014 Task 1.

We have successfully classified sentence pairs into three classes with 95.2% accuracy in two steps. The first step is separating contradiction sentence pairs from those of neutral and entailment. Our proposed model was able to recognize contradiction samples correctly by 92% when run on training set. In addition, there are 666 samples of contradiction out of 4501 samples in the training data. Further experiment was done on the test data which contains 720 contradiction samples. The results validate the initial accuracy of the training data by gaining 0.8% increase in the test data. The model was able to do so by first attempting to detect negative markers and antonym pairs between sentences. Then, to ensure that both cues were referring to the same subject, or object, in both sentences, the model looks at adjacency words that follow the cues within one to two tokens. If same entity is found to be described by either the negative markers or antonym pairs, a contradictory pair is confirmed.

The second step in our experiment setup is to classify between neutral and entailment sentence pairs. A set of encoders and classifiers were trained to perform such task. The accuracy for the trial set is 97.14% for neutral and 91.09% for entailment. All encoders and classifiers are identical in terms of network layers. An encoder model comprises three layers where the input and output layers are of same size y and the intermediate layer is dependent on the desired encoding size x; each encoder has its own set of y and x. A classifier comprises

100d and 300d fully connected layers accordingly. The final layer is a Softmax layer which outputs two probabilities. For samples of certain length *y*s whose amount did not exceed 100, they are considered too few to be trained on a classifier on their own; otherwise, the models would be fed with insufficient examples. There are 15 lengths of which contain under 100 samples. We only considered the first 10, the middle column of Table 4.1. The other five lengths, the right column, are extremely few that encoders cannot be trained on. An important point here is that we reluctantly allowed each encoder to be dedicated to each of these lengths because we need encoded representations of the samples of size *x*. The purpose for accumulating these representations is to serve them as training samples to one dedicated classifier, classifier_d in Figure 3.1.

Table 4.1 *Total number of samples based on length after two sentences are concatenated.*

Length of samples, *y*	Total	Length of samples, *y*	Total	Length of samples, *y*	Total
8	765	7	91	27	2
10	558	19	78	30	2
9	447	6	78	28	1
12	425	20	74	29	1
11	363	22	52	32	1
14	355	21	39		
13	312	23	23		
16	238	25	17		
15	235	24	17		
17	174	26	4		
18	148				

Note that an optimal encoding size means the encoding size of which gives the highest accuracy among the others for a certain length *y*. An example of finding an optimal

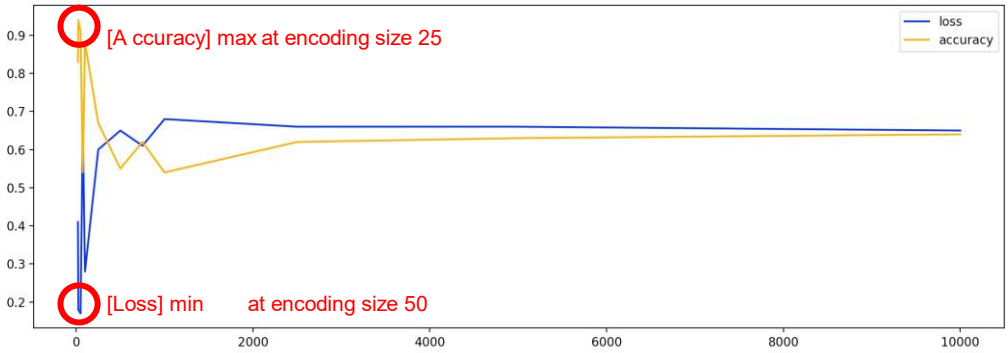


Figure 4.1 *Determining accuracy and loss for y-x-y model where y = 8*

encoding size for sentence pair of length 8 is shown in Figure 4.1. When accuracy is at its peak, the corresponding loss is not necessarily at its minimum.

As shown in Table 4.2, there is no obvious correlation between sentence pair length *y* and optimal encoding size *x*. The optimal encoding size for sentence pair of length 12 is 1000, whereas that of length 11 and 13 is 750 and 250, respectively. The randomness goes as low as 25 at length 8 and as high as 1000 at length 12 and 18. Despite encoding size of 1000, none of the encoders, and classifiers, contain up to 1M parameters. The training time for all models is of seconds and therefore proves that our system does not require heavy computational resources. With the accuracy achieved, our system has also proven not to be dependent on large-scale dataset.

Table 4.2 *Sentence pair length y vs. Optimal encoding size x*

y	x
8	25
9	250
10	75
11	750
12	1000
13	250
14	750
15	250
16	250
17	50
18	1000

In experiment setup, we first set aside all contradiction sentence pairs and fed the remaining samples through the encoders and eventually the classifiers. The models were trained and tested using TensorFlow v2.8.0. As compared to other state-of-the-art methods submitted in SemEval 2014 Task 1 track, our method exceeded the top performing models when run on both trial and test sets. We may also claim that our model has other advantages worth taking attention. That is our method does not require pretraining a heavy model. We instead employ a set of simple encoders and classifiers. The overall model is more lightweight and can easily be used across multiple platforms and environments. In fact, there is no need to train any model on a large-scale dataset and the number of parameters is not as high as 1M in

any of the encoders and classifiers. This meets one of our aims which is to be independent of high resources in training.

4.3 Benchmarks and Other Models

Benchmarks were chosen according to SemEval 2014 Task 1. Chance, Majority, Probability and Overlap baselines [5] are included in Task 1 and hence should be included in our evaluation too. Illinois-LH [40] ranked top of the category in Task 1 and is also chosen for comparison. A state-of-the-art method, to the best of our knowledge, incorporates a BERT variant which is popular, as of the time this study is conducted, and therefore should undeniably be included for comparison.

Regarding to dimensionality, we may compare our model to that of [41]. [41] attempts to classify among the three types of sentence pairs, namely, contradiction, entailment and neutral. It uses pretrained word embeddings, which is from a GloVe model, of 300d. Each sentence is represented as the sum of all word embeddings. This is a usual practice that has been prevalent across several methods. Our method only assigns one unique value, also known as word code, for each word in a sentence. Upon word representation construction, we only assign an array, or 1d tensor, of word codes, as opposed to 300d. In [41], both premise and hypothesis are fed through 100d layers in parallel before concatenating the outputs to a tanh layer. In our opinion, this 100d layer can be viewed as an embedding layer of the sentence model. Although our method may vary dramatically between 25 to 1000 encoding sizes, we may argue that our encoders consume much lower resources at training time. The total number of our training data was approximately 5000 for training the encoders which are divided unevenly for each of the encoder depending on the length of the concatenated sentence pairs. This also applies to the classifiers.

Chance, Majority and Probability baselines, [5], are derived from randomness that they do not account for sentence semantics. The Chance baseline is presumably obtained by randomly assigning one of the three entailment labels to each sentence pair under a uniform distribution. The Probability baseline follows the same assignment procedure but its distribution acts accordingly to the relative frequency of the training set. The Majority baseline sets all labels to the most occurring entailment type which is neutral. Our system, on

the other hand, seeks clarity on semantics relating to given sentence pairs. This is specially demonstrated by using the encoders to demystify the relations within the concatenation between sentence pairs. Therefore, it is no surprise that our system performs better than the three baselines. Moreover, the Overlap baseline finds common word occurrences in sentences but it also includes stop words. We viewed stop words as non-content words, which are words that do not give useful meanings on its own, that must not be accounted for; otherwise, this could lead to incorrect results especially for sentences that contain several entities which require preceding articles. Hypothetically, frequent appearances of non-content words, such as ***a***, ***and***, and ***the***, could infer completely different patterns that could be misleading for the encoders to unravel the relations between the actual content words that matter.

Table 4.3 *Performance on the SICK trial set*

	Accuracy		
	Neutral	Entailment	Contradiction
Overlap baseline	77.3	44.8	0.0
Illinois-LH [40]	86.5	83.3	77.0
- negation	85.4	0.0	86.4
Our results	97.1	91.1	94.6

Table 4.4 *Performance on the SICK test set*

	Accuracy
Chance baseline	33.3
Majority baseline	56.7
Probability baseline	41.8
Overlap baseline	56.2
Illinois-LH [40]	84.5
NeuralLog [42]	90.3
- without neural-based	71.4
- without logic-based	74.7
Our results	95.2

We observe that the number of contradictory sentence pairs are relatively low as compared to the other two types in training set, trial set and test set, despite the success rate of 90% accuracy in identifying contradictions. Contradictory sentence pairs also mostly contain negative markers in one of the two sentences, [40]. The results in Table 4.3 support the observation. Illinois-LH model performs up to 86.4% accuracy when only use its negation

feature. When all features are combined, the model only reaches 77% accuracy. In addition, only a few samples contradict by matching antonym pairs. This makes detection relatively easy but at the same time might be misleading. This is because not all negations imply contradiction. The dataset might just happen to contain negations in most contradictory pairs. Therefore, along with potential mislabeled sentence pair entries, these issues could be further investigated and confirmed by running the program on more samples that have more diverse styles of contradiction.

In NeuralLog [42], a joint logic-based and neural-based method is used to perform natural language inference on SICK-2014 dataset. When the method is performed individually, the performance drops to 71.4% for logic-based method and 74.7% for neural-based method, Table 4.4, in which both are lower than that of our program. The paper also emphasizes the importance of handcrafting knowledge relations. This is seen in our proposed method where our program contains a list of opposite words and that altering the list would affect the decision on classifying relation types.

4.4 Our System

We handle multiword expressions by detecting phrasal verb expressions and converting them into single verbs. We gathered possible phrasal verbs and their corresponding single verbs into one file. Each pair is collected into one line. When a phrasal verb is recognised, the system will change the occurrence in the sentence to its single verb correspondence. This increases the chance of matching synonyms between sentence pairs.

When matching an entity to a descriptive word or a modifier, especially adjectives, the model is designed to approach by looking at the nearest noun that follows the adjective. This imposes a problem in one scenario. An example of this is shown between "***a blond child …***" and "***a child with dark hair …***". As a human reading this, there is no doubt we can infer that both refer to a ***child*** having particular hair colours. According to the rule of our model, the first, "***a blond child***", can correctly infer that "***blond***" belongs to a "***child***", but the second, "***a child with dark hair***", is seen that "***dark***" belongs to "***hair***" and is not related to the preceding "***child***" unless we specially mark an entity on the right of "***with***" to always belong to an entity to the left of "***with***"; in other words, "***hair***" on the right of "***with***" will belong to

"***child***" on the left of "***with***". As for our current program, this is not yet implemented. If implemented, the program would be able to view that the two sentences are referring to a child with different hair colours and this could lead to contradiction prediction if the rest of the two sentences are relevant and convey similar messages. Another common example is "***red rose***" from [42]. Our program identifies a modifier of an entity only when the modifier is to the left of the entity. If a modifier is to the right of an entity, such as, "***a rose which is red***", our system would ignore the modifier. This kind of phrase is prevalent in the dataset. However, it might not be a primary concern because although the synonymity might be overlooked during detection, the encoders are likely to counteract by inferring the relations between the words themselves. The resulting encoding models may consider entities and modifiers according to proximity. Additionally, extracting relations between entities and modifiers could be a choice of future work.

For our interest, we set up an experiment to reverse the order of concatenation between sentences; all leading sentences became ending sentences, and ending sentences became leading sentences. The purpose of this experiment is to prove that our method still works the same way as its initial setup (i.e. before the switch). The results showed that the encoding vectors might differ in value but they exhibited consistent patterns. For example, if the first four elements and the following two elements in the encoding vector of a concatenation are of values a and b respectively, then the first four elements and the following two elements in the encoding vector of the reversed concatenation are also of values c and d respectively. Nevertheless, the results on the test and trial sets showed that pairs that are correctly classified in the initial setup are also correctly classified in the reverse setup.

4.5 Dataset Controversy

There are places in the dataset where we disagree with the labels. Mostly, these are neutral-labelled sentence pair entries. For example, "***a group of children is playing in the house and there is no man standing in the background***" and "***a group of kids is playing in a yard and an old man is standing in the background***" are labelled as neutral, but we argue that this should have been labelled as contradiction because one shows a presence of a ***man standing in the background***, whereas another explicitly does not. Hence the meaning implies opposition rather than irrelevance. Perhaps, this could explain why neutral-labelled entries

outnumber the other two classes. This is a very important aspect when evaluating the results because this could be the main cause for having unsatisfactory accuracy. If our conjecture on this is true, we might never figure out the main cause of poor results, whether it is from a proposed model, or from poorly labelled dataset. We might even need to make an assumption on the former and never improve the performance no matter how well a model might perform. Additionally, we also agree with [42] in opposite cases. Samples of such include "***the turtle is following the fish***" and "***the fish is following the turtle***". These sentences are originally labelled as contradiction and we could see why swapping subjects and objects could lead annotators to such decision. However, [42] suggests this as a neutral pair. To be a contradiction pair, it should either be that the second sentence keeps the ***turtle*** as the subject and the ***fish*** as the object and explicitly convey that the "***turtle***" is not following the "***fish***", or the first sentence makes the "***fish***" the subject and the "***turtle***" the object and explicitly shows that the "***fish***" is not following the "***turtle***".

Table 4.5 and 4.6 show some examples that we view should have been labeled differently from what were given in the dataset.

Table 4.5 *Entailment samples*

ID	Sentence A	Sentence B	Proposed Label	Remarks
ENTAILMENT-labelled samples				
2272	The woman is adding sugar to the meat.	A woman is adding spices to some meat.	Contradiction	Antonymy exists between "sugar" and "spices".
2715	The man kicking a boxing trainer.	The man is kick boxing with a trainer.	Neutral	"Kicking" someone and "kick boxing" with someone are different
2868	Two people are stopping on a motorcycle.	Two people are riding a bike.	Contradiction	"Stopping" and "riding" have opposite meanings.
3580	A man is playing a flute.	The man is not playing the guitar.	Neutral	Both refers to different kinds of musical instruments which causes irrelevancy.
4152	A woman is cutting meat.	There is no woman cutting an onion.	Neutral	Although could be contradiction, "meat" and "onion" are not relevant.
6819	There is no man on a bicycle riding on the beach.	A person is riding a bicycle in the sand beside the ocean.	Contradiction	One implies presence of human entity while another implies absence.

Table 4.6 *Neutral samples*

ID	Sentence A	Sentence B	Proposed Label	Remarks
NEUTRAL-labelled samples				
1649	The girl is recklessly jumping onto a vehicle.	One girl is jumping on the car.	Entailment	In both sentences, a "girl" is jumping on a vehicle which can be a car.
4002	A man is climbing a rope.	A man is coming down a rope.	Contradiction	Opposite meanings exist between "climbing" and "coming down".
4007	A man is climbing a rope.	The man is not climbing up a rope.	Contradiction	One comprises a "man" climbing but another explicitly indicates a "man" not climbing.
5391	A man is removing some food from a box.	A man is putting some food in a box.	Contradiction	"Removing" and "putting" some food have opposite meanings.

Note that many cases are taken from entailment and neutral samples as this type of sentence pairs are more challenging to determine, both manually and automatically, with high level of confidence. There are rarely doubtful contradiction-labelled samples. This could be the reason for having such a small portion of contradiction samples while a load of uncertain neutral samples.

4.51 Synonymity

To elaborate on the degree of similarity, synonym pairs between sentences could be and were used as part of the equation for determining the degree of similarity in the form of numerical values. For example, "***the man is playing the piano with his nose***" and "***a man is playing the keyboard with his nose***" are labelled as entailment. "***Piano***" and "***keyboard***" could further be considered by measuring the pair's similarity, or simply be given the same word code. The two words could be interpreted in two ways. One is that "***keyboard***" is a musical instrument which consists of black and white keys but could differ from "***piano***" by its appearance and tone. Here, both are musically related. Alternatively, "***keyboard***" could be a device which consists of a panel of keys used in typing to operate a computer. When a high level of confidence in entailment means total relevancy, the second difference would deviate the sample pair from obtaining a high score, because of low similarity or different word code. This is because the different possible meanings of "***keyboard***" would leave some room for uncertainty. Another example where ambiguity between two words causes level of

uncertainty in relevancy is between "***an animated airplane is landing***" and "***a plane is landing***". In general, "***plane***" can mean "***airplane***". This depends on the context which requires looking back and forth in the sentence. This example is more obvious that "***plane***" means "***airplane***" than "***piano***" to "***keyboard***" because only one meaning of "***plane***" can be landing, out of all the possible meanings of "***plane***". In the proposed program, it is not certain whether same word codes are always assigned to potential synonym pairs as there is a room for uncertainty.

For "***a man is squatting in brush and taking a photo***" and "***a man is crouching and holding a camera***", "***photo***" and "***camera***" are associated to each other. The current program has yet to include a feature for such association. It simply finds that both words are synonymous and therefore shares the same word code which supports the level of confidence for having high similarity. One possible upgrade is to use a knowledge graph, such as, WordNet's connections. If there exists a connection between two entities, or words, there is likely some level of relevance, either similarity or dissimilarity. The possibility of being classified as neutral would be lowered.

A challenging synonymy occurs between "***a rabbit is playing with a toy rabbit***" and "***a rabbit is playing with a stuffed bunny***". First, the program must distinguish between two kinds of "***rabbit***", namely, an animal and a toy. Second, the program must recognize that a "***toy rabbit***" is closely related to a "***stuffed bunny***", and that "***bunny***" is a toy. In our proposed method, we handcrafted the synonym list such that a "***toy rabbit***" is equal to a "***stuffed bunny***" and therefore entailment is deduced. This is an example where handcrafting synonyms could be the best option as these words are specific and do not appear often.

4.52 Hypernyms and Hyponyms

Hypernyms and hyponyms are interesting features worth taking into consideration when measuring the degree of similarity between two words. We speculate that this would have the most effect on entailment pairs among the three types. This is because in our proposed method, many synonyms can be further specified as hypernyms and hyponyms. For example, "***vegetable***" is a hypernym of "***tomato***". In the proposed method, "***tomato***" is seen as synonymous with "***vegetable***". The reason was mainly to prove that the two words are in

the same domain and therefore provide relevancy which leads to entailment. However, we can see that, in general, synonymy is bidirectional meaning x1 is x2 and x2 is x1. This is an assumption taken in the proposed method that is not always true because although a "***tomato***" is a "***vegetable***", a "***vegetable***" is not always a "***tomato***". In order to be more precise in determining similar sentence pairs, directional relation could navigate the level of specificity. For instance, "***someone is slicing a tomato***" and "***the person is slicing a vegetable***" entailment samples could be more precisely scored numerically when the program knows that there exists hypernymy/hyponymy instead of synonymy. Another simple example is "***a dog is looking around***" and "***an animal is looking around***".

4.53 Indefinite Pronouns

Generalising entities to indefinite pronouns, such as, someone, somebody, somewhere and something, could promote the level of confidence in determining an entailment pair. For example, "***the woman is cooking eggs***" and "***the woman is cooking something***" could be relevant, or similar, if "***something***" is related to "***eggs***". The truth is we would never know, given only such limited context. One suggestion is to make a program treat "***something***" as a wildcard and enable it to be relevant, or synonymous, to any entity in another sentence. In plain language, this means that "***the woman is cooking***" whatever is possible, and this includes "***eggs***". In short, "***something***" could be "***eggs***". A more prevailing case is the use of "***person***". A "***person***" can refer to male and female. "***Person***" has multiple hyponyms that it could be treated as a wildcard for a single human entity.

4.54 Active and Passive Voices

An error that might have happened in our proposed program is when one sentence is in the active voice while the other is in the passive voice. An example of this is "***a woman is slicing an onion***" and "***an onion is being sliced by a woman***". Both sentences use exact same content words but in reverse order. Although our program might allow this kind of situation to be correctly determined as an end result, the logic behind it might not align with the way the program works. One possible cure is to convert either voice into another, preferably passive voice into active voice because we are generally more familiar with the sequence subject followed by a verb predicate and a subject complement, respectively, and that there are more active voice sentences in overall. Another example is between "***a bee is clinging to***

a yellow flower" and "***a yellow flower is being clung to by a bee***". These two examples show a complete reverse order between subjects and objects.

4.55 False Logic

An example that requires careful attention and might prove our proposed method to fully work, or not, is "***a man is talking to the woman who is seated beside him and is driving a car***" and "***a woman is driving a car and is talking to the man who is seated beside her***". When extracted what each subject is doing, we would get "***a man is talking and is seated beside a woman***" and "***a woman is driving a car and is talking and is seated beside a man***". This is an entailment type sample. However, the proposed method does not see this. This is a circumstance that could have given a prediction from a false logic. This is because the proposed method might see these sentences as neutral as the appearance between "***man***" and "***woman***" are in reverse order; hence the subjects of the sentences are different, and so, no matter what the subjects do, they are considered irrelevant.

4.56 Rearranging words

The proposed program requires simplifying groups of words in many sentences. This is seen in a sentence like "***a man is participating a race for bmxs***". "A race for ***bmxs***" can be simplified into "***a bmx race***". This would promote higher level of confidence in determining entailment samples. For example, with a counter sentence of "***a man is participating in a bmx race***", the similarity would gain top score because this would result in exact match after the simplification. This kind of situation exists throughout the dataset and therefore could be worth handling with a converting mechanism.

4.57 Knowledge Assumption

A not-so-rare case occurs between sentences "***a child in a red outfit is jumping on a trampoline***" and "***a little boy in red clothes is jumping into the air***". Despite having a common action such as "***jumping***", this kind of sentence pairs may require special knowledge to infer with higher level of confidence. "***Jumping on a trampoline***" implies "***jumping into the air***" because when we jump on a trampoline, we are jumping into the air. However, "***jumping on a trampoline***" does not contain any information about air directly. This shows

that there are cases that must depend solely on context. There exists an ambiguous sample between "***some cheerleaders are taking a break***" and "***some cheerleaders are dancing***". This is labeled as neutral. However, as we know that dancing is the main activity for cheerleading, or in other words, they can be synonymous, "***taking a break***" could mean taking a break from dancing. If the context was explored at a deeper level, "***dancing***" and "***taking a break***" could convey opposite meanings, as for cheerleading. This would eventually lead to contradiction.

4.6 Semantic Relatedness vs. Textual Entailment

SICK-2014 dataset focuses on generic semantic knowledge and semantic compositionality as described in [5]. This is the reason for choosing this dataset despite all the aforementioned doubtful cases. Generic semantic knowledge is captured in the process of finding similar and opposite words. Although we handcrafted a list of synonyms and antonyms, we did not specify any domains. Generality is also seen where we set up a list of common negative markers so that any detection would reverse the polarity of the context. Semantic compositionality is reflected through handling multiword scenarios such as phrasal verbs. This is when a group of words is recognised as one constituent of a sentence. We explicitly turn them into single words for ease in handling. The main benefit for this is to compare the words to that of the other sentence so that the program could recognise synonyms, antonyms and possibly negative markers. Moreover, we believe that choosing to evaluate on entailment relation over semantic relatedness, which is also a value given by the dataset [5], should be a better indication on how well a system understands computational semantics at a more general level. The program is only required to identify the type of relatedness instead of a specific value of relatedness. Numerical semantic relatedness values between zero and five may give a more specific insight to the degree of similarity but textual entailment classes identify relevancy between entailment and contradiction. This is the difference that made us go with using the latter.

There is a subtle difference between how relatedness score and textual entailment classification are deduced. At first glance, relatedness score might seem to only be a numerical range for the discrete entailment classes. This means higher relatedness scores imply higher level of confidence for a sentence pair being contextually related. Conveniently, the other end of the spectrum would imply neutral as this is where sentences are irrelevant.

We can see that relatedness scores do not differentiate between entailment and contradiction because both would imply relevance and hence we cannot use relatedness scores as our primary indicators of similarity. Whether this is clearly defined and widely accepted in general, we observed that relatedness scores work this way in the dataset; scores of 4 to 5 are usually of either entailment or contradiction classes, otherwise neutral. Simple examples of neutral with low relatedness score include "***a dog is barking noisily***" and "***a jet is flying***". However, there exists samples that are less obvious. "***There is no man playing a guitar***" and "***a man is playing a piano***" are originally labelled as neutral and gain a relatedness score of 3.2 which is considered high for a neutral type. One may view these sentences as contradiction as the main difference is that one implies a presence of a man while another implies an absence. There is still relevancy as both sentences are referring to a "***man***" playing a musical instrument, just only different types. On the other hand, the sentences could be considered as neutral when one views "***guitar***" and "***piano***" are completely unrelated. However, with the relatedness score of 3.2, there must be some relevancy between the two sentences. This is an example of label controversy and, possibly inconsistency, which illustrates that when we agree with relatedness score, we might not agree with the labelled textual entailment class as given by the dataset. Our program would have predicted this as contradiction for the reason mentioned and this would only result in incorrect prediction.

Table 4.7 *Sentence pair samples*

ID	Sentence A	Sentence B	Relatedness Score	Class
5618	A man is severing the toe of an empty leather boot with a sword.	A woman is severing the toe of an empty leather boot with a sword.	4	Neutral
6781	A black dog in the snow is jumping off the ground and catching a stick.	There is no dog jumping for a Frisbee in the snow.	2.65	Neutral
8130	Several young people are posing for a photo and holding beers.	Several old people are posing for a photo and holding beers.	3.265	Neutral
8199	A bride with a white dress is looking down.	A bride with a white veil is looking down.	4.435	Neutral
8206	A bride with a black veil is looking down.	A woman is looking down and is wearing a wedding veil.	3.8	Entailment
8742	Three people are walking across a rope and wood bridge over a river.	Three people are walking across a rope and steel bridge over a river.	4	Neutral
9576	A dog is fetching a stick out of very clear water.	A dog is fetching a stick out of very dirty water.	4	Neutral

Consider Table 4.7., the question seems to be what level of similarity we are focusing. Is ***stick*** not related to ***Frisbee*** (id 6781), as for ***wood*** to ***steel*** (id 8742), ***clear water*** to ***dirty water*** (id 9576), and ***young*** to ***old*** (id 8130) that these sentences made neutral? Does ***a bride with a white dress*** (id 8199) not imply ***a bride in white veil*** that the label is neutral, while ***black veil*** (id 8206) and ***wedding veil*** are basically interchangeable that the label is entailment? It is interesting to note that ***white dress*** and ***white veil*** in sentence pair id 8199 is of neutral type with a relatedness score of 4.435, whereas ***black veil*** and ***wedding veil*** in sentence pair id 8206 is of entailment type with lower relatedness score. Another doubtful label exists between a ***man*** and a ***woman*** in sentence pair id 5618. The sentences only differ by the gender of the subject. The relatedness score is at the high end but is also classified as neutral.

Chapter 5

FUTURE WORK

Our proposed encoding method converts a sentence into one-dimensional vector space. This represents word code for each pre-processed word. This feature may not suffice in many cases. Therefore, adding more features could be a possible improvement to the program. For example, the second dimension could be a row of binary values indicating action or stative verbs, or other types of verb, and the third dimension could indicate subject, abstract, or collective nouns, or other types of noun. Such features could make a sentence representation more unique and hence strengthens the level of confidence of being classified as such classes.

Table 5.1 *The Penn Treebank syntactic tag set*

ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Subordinate clause
SBARQ	Direct question introduced by *wh*-element
SINV	Declarative sentence with subject-aux inversion
SQ	Yes/no questions and subconstituent of SBARQ excluding *wh*-element
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Constituent of unknown or uncertain category
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace of wh-Constituent

New features could be identified by using the results of constituency parsing, specifically, a parsing tree. A parsing tree reveals relations between parent and child. Our program could benefit from this by grouping words under the same parent. There would be multiple forms extracted depending on the parent and its tree depth. The parent could be a clause-level or a phrase-level syntactic tag, as seen in Table 5.1. Like the POS tag set from Penn Treebank, the syntactic tag set is derived from the Penn Treebank [43]. In this section, we investigate the usefulness of Penn Treebank syntactic tag set and address any challenges that must be handled for the tags to be used.

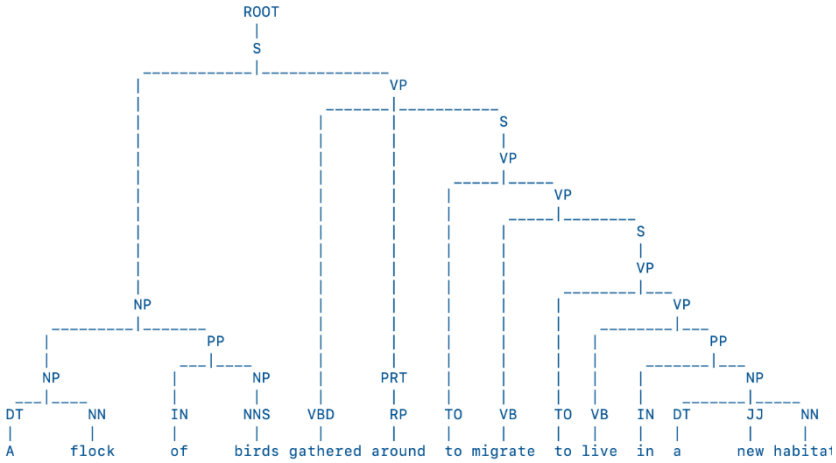


Figure 5.1 *NP VP under S tag*

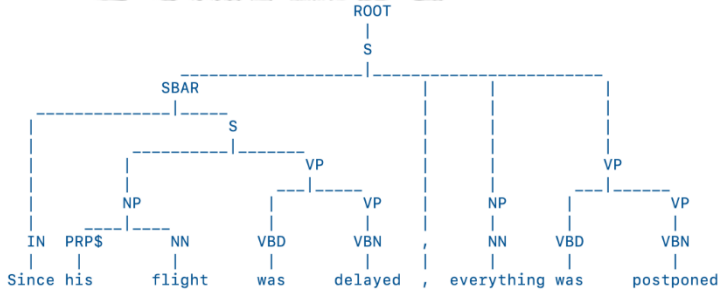


Figure 5.2 *Subordinate clause SBAR*

Phrases such as ***a flock of birds*** in Figure 5.1 would be identified under a noun phrase and we can further derive that ***flock*** is the main entity instead of ***birds***, or vice versa, if this is informative. A modifier-entity pairing problem stated in the discussion section would be solved, as a noun phrase would conveniently distinguish and group them under the same hood. A subordinate clause would be identified, as well as, the main clause. For example, ***his flight was delayed***, in Figure 5.2, is ultimately under SBAR which tells that this is not the main message of the sentence. The main message is in the form of noun phrase followed by verb phrase which is typical of a complete independent sentence. Once these are known, the main verb-predicate could be extracted and indicated in one of the additional dimensions of the vector space. There are several possibilities to benefit from such tree. Different types of verbs, such as, infinitives and gerunds, could be filtered and marked as different descriptive features such as reasoning. For example, ***to migrate*** and ***to live in a new habitat***, in Figure 5.1, give reasons, or purposes, to the main verb-predicate ***gathered***.

Another benefit of using a constituency parsing tree is the ability to identify whether a given clause is a complete sentence, or not, by looking at the syntactic direct child tag of ROOT. Typically, a complete sentence would have S as the top of the tree under ROOT, or SQ if the sentence is in a question form; otherwise, the whole clause would descend from SBAR or SBARQ. There are occasional fragment tags that precede all the other constituent tags but we are only concerned with whether the ancestor is S, or SQ. If that is not the case, our program could immediately discard the given clause and refuse to proceed with the classification task. This is because our primary assumption in this work is that given samples must be complete sentences.

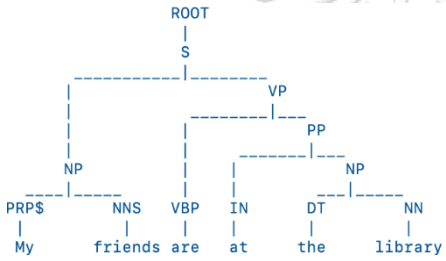


Figure 5.3 *Locational PP tag*

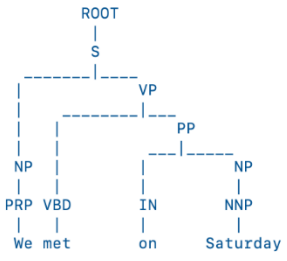


Figure 5.4 *Temporal PP tag*

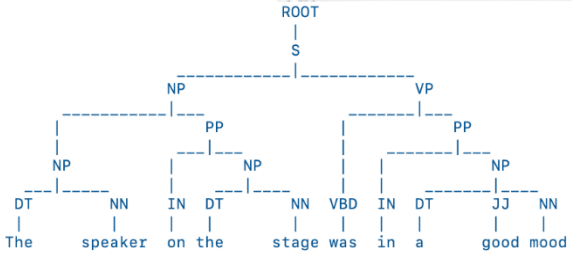


Figure 5.5 *Different PP functions*

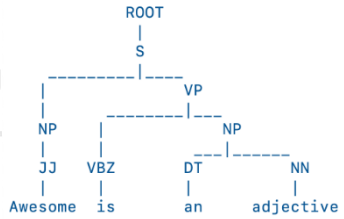


Figure 5.6 *An adjective as entity*

Nouns that are considered secondary to the main entities, such as, subject and object, can be extracted by those under prepositional phrases, PP. For example, ***habitat*** in Figure 5.1 and ***library*** in Figure 5.3. Prepositional phrase implies locational or temporal information and can be used to describe main entities, ***on the stage*** in Figure 5.5. It can also be used as a subject complement, Figure 5.3 and 5.4, when resides on the right of the main verb-predicate. For example, this applies to ***in a good mood*** but not ***on the stage*** in Figure 5.5. When acting as a subject complement and there is no object in the sentence, the entities in the prepositional

phrase may be included as one of the main entities of the sentence as applied to ***library*** in Figure 5.3. Note that entities are commonly but not limited to nouns as seen ***awesome*** in Figure 5.6.

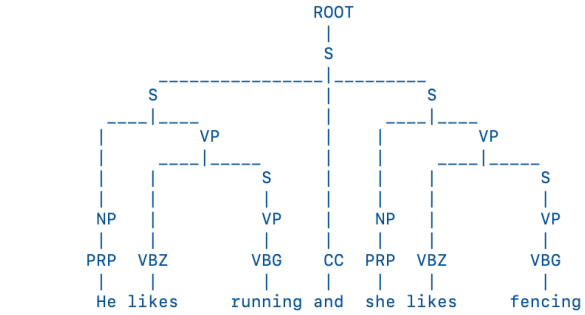


Figure 5.7 *Balanced CC between Ss*

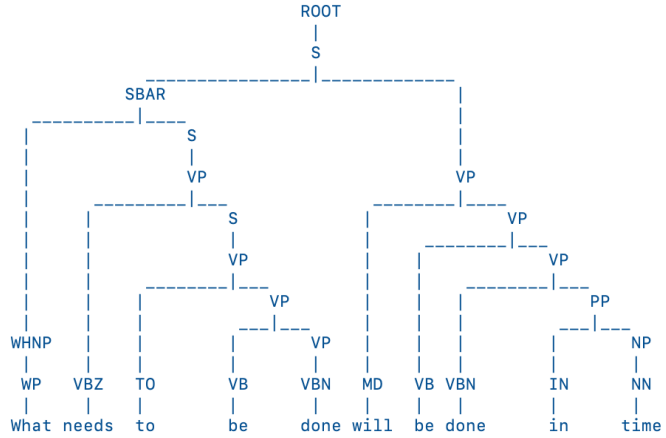


Figure 5.8 *SBAR as subject*

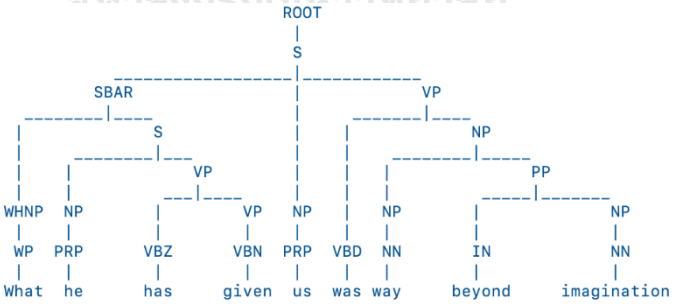


Figure 5.9 *Co-level SBAR and NP*

Another useful tag is coordinating conjunction, or CC. When two independent sentences are joined by a coordinating conjunction, a CC tag can be used to recognize that there will be two sets of main subjects, or expletive there, and two main verb-predicates, as shown in Figure 5.7.

Constituency parsing trees may not be deduced the way we expect them to be. In order to derive main subjects and main verb-predicates, we ought to look for noun phrases and verb phrases at lower depths. However, main subjects may not explicitly be in the form of noun which could lead to a false extraction of main entities. A main subject could be in the form of SBAR, such as, ***What needs to be done will be done in time*** in Figure 5.8. Here, the

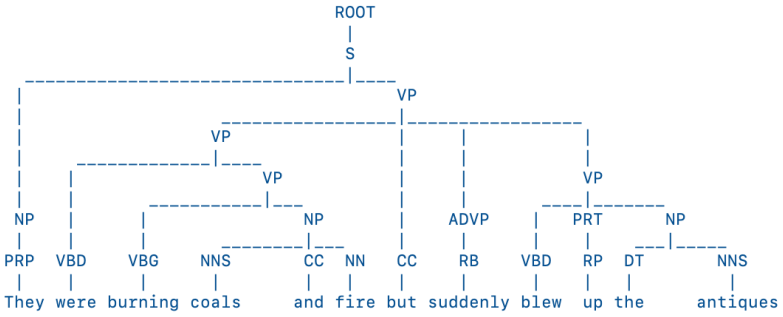


Figure 5.10 *Balanced constituents around CC*

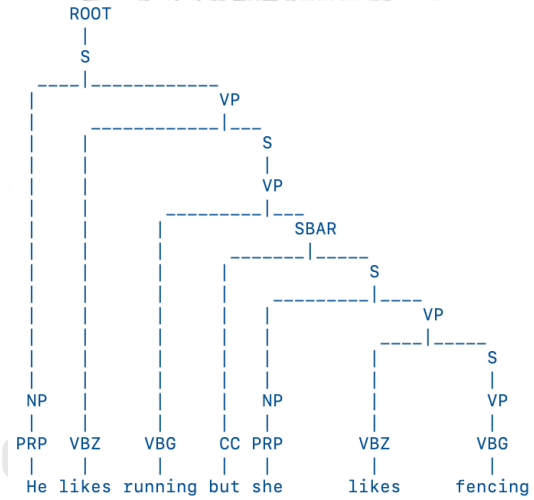


Figure 5.11 *Unbalanced constituents around CC*

main subject is the phrase ***what needs to be done***. It is under SBAR and to make matter worse, there is no noun phrase residing in any branches. Although there is a WHNP, which is a noun phrase for an interrogative word, ***what***, the whole phrase must be considered as the subject as ***what*** alone is not enough to give a meaning. The same kind of example which contains a co-level noun phrase is ***what he has given us was way beyond imagination*** in Figure 5.9. Here, ***us*** can be used as the main entity as the parent NP is right under S which is the head of the tree, but that would not give much context without the co-level SBAR.

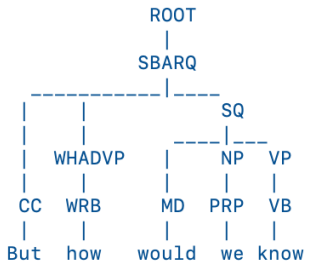


Figure 5.12 *Conjunction at head of sentence*

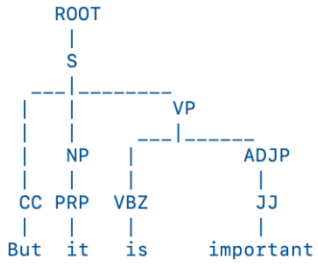


Figure 5.13 *CC at head of sentence*

Another inconsistency lies in the presence of a CC tag. There are two situations where a CC tag may occur. This is when a CC is delivering a set of balanced constituents such as two noun words are separated by ***and*** to indicate multiple items, or entities, such as ***coals and fire*** in Figure 5.10, and two Ss are separated by ***and*** to indicate a compound sentence, such as that in Figure 5.7. The constituents on both sides are balanced because they are at the same depth level. Another situation is when a CC tag is under SBAR, such as ***but*** in Figure 5.11. Although S resides both on the left and right of CC, the two Ss are not at the same depth level as in the first situation. This case normally results in tree branches cascading down from left to right. From observation, this is due to the fact that the word under the CC tag is a subordinate conjunction. To add more of this type of POS tag, a situation, where a complete sentence is falsely found, is by having a head S tag with the leftmost child being a CC tag, Figure 5.13. In English, a sentence must not start with a conjunction, also shown in Figure 5.12. From our observations, we conjecture the rule is broken as there exists an NP and VP pair, and the constituency parser just ignores the fact that a CC tag occurs on the leftmost side. We can see that having an ancestor S still requires an extra verification in order to deduce that a sentence is complete.

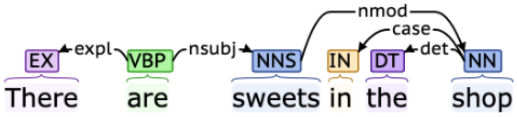


Figure 5.14 *Expletive there at head of sentence*

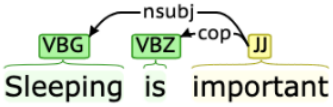


Figure 5.15 *Gerund as subject*

Figure 5.16 *Dependency parsing with CC tag*

Apart from constituency parser, a dependency parser would also benefit in adding more features and could be a faster and better way to obtain subjects and verb-predicates.

Again, there are inconsistencies and exceptions in the parser that require careful handling for extracting useful information. Examples of dependency parsing are shown in Figure 5.14, 5.15 and 5.16. Hypothetically, if any of these features are useful, the proposed method could be having plausible indicators in interpreting the results and explaining such classifications.

The idea of using a constituency parser to aid in adding features to the vector space could be very problematic for low-resource languages. This is because it is unlikely to find at least one substantial constituency parser for those languages. Nevertheless, we believe that our proposed method is worth testing on different languages, by creating a word code table and handcrafted resources.

Furthermore, training all encoders and classifiers to output all three types instead of two types could be a possible improvement. This would bypass the stage for detecting antonym pairs. By using trained models to classify contradiction samples, this could lessen any false predictions. In our proposed method, words in proximity to antonym pairs and negative cues are determined as entities being modified. This is not always true. In the case of ***red roses***, the rule can be applied since ***roses*** is in proximity to ***red***, especially that it is on the immediate right of the modifier. However, for ***roses which are not red***, ***roses*** is not considered as the entity being referred if proximity means within two tokens to the right of ***red***. In fact, preliminary results on performing a 3-class classification confirmed that filtering contradiction samples would produce better accuracies. We speculate that this might be due to the fact that the given contradiction samples mostly contain explicit negative markers or antonym pairs which are presumably more important attributes than relations between modifiers and entities. Also, contradiction samples without explicit cues can be very similar in structure to that of entailment samples. Further investigation could be made.

Finding an ultimate common optimal encoding size could be another possible work. In doing so, all encoders would have the same *x*, where *x* is the optimal encoding size for each encoder. When *x* is the same for all encoders, a set of classifiers can be simplified to one classifier. The input to the classifier is the same. This promotes in training the model since the classifying module will have much more samples to one classifier and hence could lead to higher accuracy.

Chapter 6

CONCLUSION

In this work, a method for textual entailment is proposed to classify among entailment, neutral and contradiction classes. The input to the framework is a pair of sentences and the three output signals can be found in either the early stage for contradiction pairs or the final stage for the other two types. At the early stage, a database is used to detect synonyms and phrasal verbs. Phrasal verbs were generalised into single words in order to promote finding synonyms between sentence pairs. A database of opposite words is also used to find antonym pairs between sentence pairs. Any antonym pair detection with same entity reference will trigger a signal for contradiction type and the process will be terminated. This is also true if a negative marker is found in one sentence but not the other. Otherwise, subsequent processes will be executed. These include a set of encoders and classifiers. All encoders are configured in an *y-x-y* manner where *y* is the length after two input sentences are concatenated and *x* is the optimal encoding size. All classifiers follow the same network architecture and output the probabilities of being entailment and neutral types.

The proposed encoding scheme gives over 90% accuracy for all types when evaluated on SICK-2014 dataset [5]. This indicates that it is possible to train and run samples on a lightweight framework such as the proposed method. By lightweight, we mean that no layers in any model of size over 1000 are involved and training on large-scale datasets is omitted. In fact, no complex neural structure was employed. We may say that our proposed model has low dimensionality in overall.

Moreover, one possible improvement to the encoding scheme could be to associate more features to each word. Features such as types of verb (action, stative, etc.) and types of noun (abstract, collective, etc.) could be added as a new dimension to the input of the encoders. This could lessen any predictions with false logic, regardless of correctness. The inference could be deduced with more specific reasonings making the classification more interpretable.

REFERENCES

1. Pennington, J., R. Socher, and C.D. Manning. *Glove: Global vectors for word representation*. in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.
2. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
3. Devlin, J., et al., *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv preprint arXiv:1810.04805, 2018.
4. Vaswani, A., et al., *Attention is all you need.* Advances in neural information processing systems, 2017. **30**.
5. Marelli, M., et al. *SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment*. 2014. Dublin, Ireland: Association for Computational Linguistics.
6. Farouk, M., *Measuring sentences similarity: a survey.* arXiv preprint arXiv:1910.03940, 2019.
7. Martinez-Gil, J. and M. Pichler. *Analysis of word co-occurrence in human literature for supporting semantic correspondence discovery*. in *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*. 2014.
8. Levenshtein, V.I. *Binary codes capable of correcting deletions, insertions, and reversals*. in *Soviet physics doklady*. 1966. Soviet Union.
9. Mihalcea, R., C. Corley, and C. Strapparava. *Corpus-based and knowledge-based measures of text semantic similarity*. in *Aaai*. 2006.
10. Pawar, A. and V. Mago, *Calculating the similarity between words and sentences using a lexical database and corpus statistics.* arXiv preprint arXiv:1802.05667, 2018.
11. Fernando, S. and M. Stevenson. *A semantic similarity approach to paraphrase detection*. in *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*. 2008. Citeseer.
12. Wang, Z., H. Mi, and A. Ittycheriah, *Sentence similarity learning by lexical decomposition and composition.* arXiv preprint arXiv:1602.07019, 2016.
13. Abdalgader, K. and A. Skabar. *Short-text similarity measurement using word sense disambiguation and synonym expansion*. in *Australasian joint conference on artificial intelligence*. 2010. Springer.
14. Farouk, M. *Sentence semantic similarity based on word embedding and WordNet*. in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*. 2018. IEEE.
15. Lee, M.C., J.W. Chang, and T.C. Hsieh, *A grammar-based semantic similarity algorithm for natural language sentences.* The Scientific World Journal, 2014. **2014**.
16. Batanović, V. and D. Bojić, *Using part-of-speech tags as deep-syntax indicators in determining short-text semantic similarity.* Computer Science and Information Systems, 2015. **12**(1): p. 1-31.
17. Ji, Y. and J. Eisenstein. *Discriminative improvements to distributional sentence similarity*. in *Proceedings of the 2013 conference on empirical methods in*

natural language processing. 2013.
18. Mikolov, T., et al., *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013.
19. Mueller, J. and A. Thyagarajan. *Siamese recurrent architectures for learning sentence similarity*. in *Proceedings of the AAAI conference on artificial intelligence*. 2016.
20. Almeida, F. and G. Xexéo, *Word embeddings: A survey.* arXiv preprint arXiv:1901.09069, 2019.
21. Turian, J., L. Ratinov, and Y. Bengio. *Word representations: a simple and general method for semi-supervised learning*. in *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010.
22. Mikolov, T., W.-t. Yih, and G. Zweig. *Linguistic regularities in continuous space word representations*. in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013.
23. Harris, Z.S., *Distributional structure.* Word, 1954. **10**(2-3): p. 146-162.
24. Collobert, R. and J. Weston. *A unified architecture for natural language processing: Deep neural networks with multitask learning*. in *Proceedings of the 25th international conference on Machine learning*. 2008.
25. Salton, G., A. Wong, and C.-S. Yang, *A vector space model for automatic indexing.* Communications of the ACM, 1975. **18**(11): p. 613-620.
26. Turney, P.D. and P. Pantel, *From frequency to meaning: Vector space models of semantics.* Journal of artificial intelligence research, 2010. **37**: p. 141-188.
27. Mikolov, T., et al. *Neural network based language models for highly inflective languages*. in *2009 IEEE international conference on acoustics, speech and signal processing*. 2009. IEEE.
28. Mikolov, T., et al., *Distributed representations of words and phrases and their compositionality.* Advances in neural information processing systems, 2013. **26**.
29. Bojanowski, P., et al., *Enriching word vectors with subword information.* Transactions of the association for computational linguistics, 2017. **5**: p. 135-146.
30. Joulin, A., et al., *Bag of tricks for efficient text classification.* arXiv preprint arXiv:1607.01759, 2016.
31. Deerwester, S., et al., *Indexing by latent semantic analysis.* Journal of the American society for information science, 1990. **41**(6): p. 391-407.
32. Lund, K. and C. Burgess, *Producing high-dimensional semantic spaces from lexical co-occurrence.* Behavior research methods, instruments, & computers, 1996. **28**(2): p. 203-208.
33. Rohde, D.L., L.M. Gonnerman, and D.C. Plaut, *An improved model of semantic similarity based on lexical co-occurrence.* Communications of the ACM, 2006. **8**(627-633): p. 116.
34. Lebret, R. and R. Collobert, *Word emdeddings through hellinger pca.* arXiv preprint arXiv:1312.5542, 2013.
35. Peters, M.E., et al. *Deep Contextualized Word Representations*. 2018. New Orleans, Louisiana: Association for Computational Linguistics.
36. Liu, Q., M.J. Kusner, and P. Blunsom, *A survey on contextual embeddings.* arXiv preprint arXiv:2003.07278, 2020.

37. Radford, A., et al., *Improving language understanding by generative pre-training.* 2018.
38. Radford, A., et al., *Language models are unsupervised multitask learners.* OpenAI blog, 2019. **1**(8): p. 9.
39. Manning, C.D., et al. *The Stanford CoreNLP natural language processing toolkit*. in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014.
40. Lai, A. and J. Hockenmaier. *Illinois-LH: A Denotational and Distributional Approach to Semantics*. 2014. Dublin, Ireland: Association for Computational Linguistics.
41. Bowman, S.R., et al. *A large annotated corpus for learning natural language inference*. 2015. Lisbon, Portugal: Association for Computational Linguistics.
42. Chen, Z., Q. Gao, and L.S. Moss, *Neurallog: Natural language inference with joint neural and logical reasoning.* arXiv preprint arXiv:2105.14167, 2021.
43. Marcinkiewicz, M.A., *Building a large annotated corpus of English: The Penn Treebank.* Using Large Corpora, 1994. **273**.

VITA

NAME	Thanaporn Jinnovart
INSTITUTIONS ATTENDED	University of New South Wales
PUBLICATION	Jinnovart, T., Lursinsap, C. (2023). Y-X-Y Encoding for Identifying Types of Sentence Similarity. In: Wah, Y.B., Berry, M.W., Mohamed, A., Al-Jumeily, D. (eds) Data Science and Emerging Technologies. DaSET 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 165. Springer, Singapore. https://doi.org/10.1007/978-981-99-0741-0_37
AWARD RECEIVED	Best Paper Award and Best Presentation Award at International Conference on Data Science and Emerging Technologies (DaSET 2022)