Department of Computer Science

Submitted in part fulfilment for the degree of
MEng Computer Science with Artificial Intelligence.

Vagueness and Ambiguity Removal using Transformers

Adam Barr

May 2022

Supervisor: Dimitar Kazakov

To all the people who can't quite find the right words to say

Acknowledgements

There are a lot of people I'd like to acknowledge, too many to list here, but I should probably at the very least acknowledge Intuety for providing my data and Dimitar Kazakov for his support, so thank you!

Contents

List of Figures

List of Tables

Executive Summary

Vague and ambiguous text causes problems with most Natural Language Processing (NLP) products, in an industrial setting it can have a severe impact on a users experience with, and opinion of, a tool. Intuety is a company that uses NLP to improve the quality of Risk Assessments in the construction industry. Vague and ambiguous text in a risk assessment causes Intuety's product to make worse recommendations to the user, and given the domain of the product (safety on a construction site), identifying these problematic pieces of text can have a potentially life saving impact.

During the project, I explore and critique different NLP techniques that could be used to catch vague and ambiguous inputs before Intuety's product attempts to make recommendations on them. As I explore the different NLP techniques available to me, I settle on using the pretrained all-mpnet-base-v2 Sentence-BERT encoder to calculate sentence encodings that represent the semantic meaning of input text, these encoding are compared with my knowledge base using cosine similarity to provide an insight quality of the input sentence.

I decided measuring the "absolute" vagueness/ambiguity of the input was unfeasible and unnecessary, what I really care about is how the input compares to to to the topics that Intuety is able to make recommendations on. Therefore I am measuring the vagueness and ambiguity against my knowledge base, as even if a sentence is well worded, if it has nothing to do with construction it shouldn't be going through Intuety's product and wouldn't produce good recommendations, therefore should be flagged as vague.

Intuety were kind enough to allow me to use some of their data in my knowledge base, this data set is regularly updated and maintained by a team of humans and contains 5594 pieces of text you would reasonably expect to see in a construction risk assessment. Intuety also provided my testing data, it contains pieces of text extracted from real life construction risk assessments. Each piece of text in both my knowledge base and testing data has one of three labels, Activity, Risk, or Mitigation, this label sheds more information in how vague the text is as it can be measured against only the pieces of text with its label in the knowledge base.

The knowledge base and testing data comes under Intuety's intellectual property, so for demonstrations of my project and validation of any claims I make about that data please contact me directly using ab2474@york.ac.uk.

I defined four clear objectives to be fulfilled during my project, my first objective was to implement functions that allowed my knowledge base to grow to accommodate new pieces of text so that valid but previously unseen ideas could be

learned. I achieved my first objective by implementing a "GraphHelper" class that contained all the functions I needed to interact with my graph database. I also fulfilled my fourth objective that focused on performance time, I set myself a target of all results being returned within 5 seconds and surpassed this substantially with even my slowest testing runs returning results in under 3 seconds. My second and third objectives focused on vagueness and ambiguity being reliably detected. I found that comparing the semantic similarity of an input to a knowledge base could provide an insight into how vague a sentence was against a data set, and attempted to back up this result with information about my distribution, visualisations of the principal components of semantic matches, and anecdotal evidence. I felt that I had not seen any evidence to support the idea that semantic similarity could be used to identify ambiguity, however after identifying issues within my knowledge base I could not conclude whether the reason for the lack of evidence was the method used or my knowledge base.

I came to the conclusion that, despite being maintained and cleaned by humans, my knowledge base has too much variation of vagueness and ambiguity in itself to be able to provide meaningful information about another piece of text. There is potential in using my method with a cleaner data set to achieve better results, I speak about this and other further work that could produce better results.

My project can have serious ethical impacts, if it is ever used in industry I want to emphasise it should be done with proper care and that humans should always be made aware of any decision my algorithm comes to and have the power to overrule any decision it makes. This is a tool for checking documents and should not be considered a ground truth, I speak at length in my results and conclusion about issues I have encountered with the technique.

1 Introduction

1.1 Natural Language Processing and Text Similarity

Natural Language Processing (NLP) is an important field in Artificial Intelligence, it attempts to allow computers to communicate with humans the way humans have been communicating with each other for millennia, through language. The field of natural language is becoming increasingly industrialised and important to businesses, with a survey by Gradient Flow finding that a third of companies raised their NLP funding by over 30 percent from 2020 to 2021 [1].

Text Similarity is an important tool in the field of Natural Language Processing, it is used to find the closeness of meaning between two different pieces of text. Text similarity algorithms are the building blocks of a huge array of different natural language products, search engines use text similarity to compare a query to the contents of a website [2], in the Legal sector text similarity is used to compare different contracts [3], plagiarism detection software will even have analysed this report to make sure these words are not too similar to anything that can be found in academic papers or on the internet [4].

1.2 Problem Identification

Intuety is a company that is using Natural Language Processing to improve the quality of Risk Assessments in the construction industry [5]. Text similarity is used in their product to compare sentences taken from a users Risk Assessment to sentences in a knowledge base, if a sentence is similar to a sentence in the knowledge base it will provide recommendations on areas that may have been overlooked in the Risk Assessment. A big problem Intuety faces is attempting to provide recommendations for input sentences that are of a poor quality, these sentences may be too vague meaning the product struggles to find any one sentence in its knowledge base that means the same as. Additionally, the sentences may be too ambiguous meaning the product does not know which of the sentences the input is most applicable too. To paraphrase a famous saying in the field of Artificial Intelligence [6], a poor quality of input sentences results in a poor quality of recommendations.

In this project I attempt to find a solution to the problem of poor input sentences

producing poor recommendations, I have written an algorithm that can be used as a filter to catch vague and ambiguous inputs and notify the user before the poor quality recommendations are returned.

1.2.1 Risk Assessments

Every company in the United Kingdom is required by law to protect their employees and members of the public from harm [7]. Under the Management of Health and Safety at Work Regulations 1999 legislation [8] this responsibility means businesses in the United Kingdom must, at a minimum, identify the hazards related to their business, identify the risks related to the hazards, and take action to eliminate the hazard or control the risk. The process of completing these checks is called a "Risk Assessment". Writing down risks assessments is not compulsory, however it is the most common way for companies to demonstrate that they have fulfilled this responsibility and are complying with the law.

Risk Assessments have no official formatting rules, however, it is standard practice for them to be displayed in tables which often have columns that store information on the activities, hazards, risks, and risk controls related to a companies work. The columns may also store more information depending on a companies internal policy, for example many companies also assess the severity and likelihood of identified risks. Intuety has developed a method to extract information from these tables, and categorise the contents into one of three labels: activities, risks, and mitigations (section 1.2.2).The input text for my algorithm will be the extracted text from cells in Risk Assessment tables, and their individual labels.

1.2.2 Activities, Risks, and Mitigations

Intuety currently classifies text from risk assessment tables into one of three labels: an activity, a risk, or a mitigation. Definitions of what these labels mean may differ slightly for different companies and purposes therefore it is useful to define what an each label is in regards to my project. An activity is a thing that is happening or a task that is being completed, a risk is the threat posed to employees or members of the public by an activity, and a mitigation is the control used to eliminate the risk or mitigate its consequences. An example that helped me understand the labels is that of visiting a zoo, where visiting the lion exhibition is an activity, being eaten by a lion is a risk posed by the activity, and putting the lion in a cage is a mitigation used to control the risk.

1.2.3 Vagueness and Ambiguity

It is important to define exactly what vagueness and ambiguity mean as they are core ideas to the project. In general, I would define a vague piece of text

as a sentence or phrase with unclear meaning, and an ambiguous piece of text as a sentence or phrase with more than one possible interpretation. Whilst Wittgenstein suggested there is beauty to be found in the vague and unclear [9], and many linguists have argued that vagueness and ambiguity are useful tools in communication [10], vagueness and ambiguity in a construction site's Risk Assessment are definitely a bad thing and may lead to serious injury, or even death.

In the scope of my project, I will be measuring the vagueness or ambiguity of a sentence against a knowledge base. This knowledge base will contain information about different activities that take place on a building site, the risks associated with them, and the mitigations used to control the known risks. This means that in my project I will be tweaking my general definitions of vague and ambiguity. Below are the definitions of vagueness and ambiguity I have used throughout my project:

Vague A piece of text whose meaning is not in the knowledge base

Ambiguous A piece of text whose meaning could reasonably be interpreted to be the same as multiple items in the knowledge base.

1.2.4 Examples

Vague Example: "Site works" is a real world example of a vague piece of input text, the full extracted row is shown in appendix table .1 and has the risk label. Whilst the activity and mitigation from the example table could probably more descriptive their meaning is clear to people on a construction site. "Site works" is a vague input because it is not clear what it means, despite what some members of the media say work on a construction site is a daily activity and just referencing it is not clear at all to a reader. Looking at the definition of risk in section 1.2.2, "Site works" itself is not a clearly defined threat to anyone's safety. An example a good input to go in its place may be "Site vehicles hitting piling machinery", this example describes a clear threat to people on site.

Ambiguous Example: An example of an ambiguous input row would be the activity "Scaffolding", risk "Falling objects", and mitigation "Tie objects up", appendix table .2 shows what this would look like in the line of a risk assessment table. When put through the recommendation process these inputs will have a poor quality of recommendations. Each piece of text could relate to a variety of things that happen on construction sites and there is not enough information in them to determine what specifically they are referencing. For example, "scaffolding" could mean putting up scaffolding, working on scaffolding, or even having scaffolding structures around members of the public; "Falling objects" could refer to builders dropping objects off the scaffolding when they are working on it, or pieces of the scaffolding itself falling down; and "Tie objects up" may refer to tie worksite equipment to other worksite equipment, to the scaffolding structure, or to the builder themselves.

1.3 Goals, Objectives, and Criteria

In this section I will attempt to develop the general goal of my project beyond what I described in section 1.2 into a set of clearly defined goals. I will also create a set of objectives that when implemented contribute to the goals being met, and a set of quantifiable criteria that can be used to assess how well each objective was met.

1.3.1 Goals

The goal of my project is to be able to detect text that is vague or ambiguous when compared to my knowledge base. More specifically I want to develop an algorithm that can be used by Intuety to alert a user to problematic pieces of texts within their Risk Assessment before recommendations are made on them. Preventing vague and ambiguous text from being used to make recommendations would improve the quality of recommendations shown to a user, improving their opinion of the system as a whole. Fulfilling this goal will ultimately result in users of Intuety's system producing risk assessment that are clearer, easier to read, and less prone to misinterpretation.

I feel clearest definition of my goal comes in the form of a research question, this means in my conclusion I can easily assess whether it was met. Here is the research question that defines my goal.

Research Question If I measure the similarity of a piece of text against a data set, can I use this result to interpret information about the vagueness of the the input text against the data set?

1.3.2 Objectives and Criteria

In this subsection, I identify multiple objectives to reach these goals, and criteria that can be used. I will later use these criteria to assess whether my objectives have been fulfilled, and ultimately whether my goal is achieved.

Objective 1: Implement ability to grow the knowledge base to include previously unseen non-vague inputs that have no existing similar matches. The criteria for this objective is whether the relevant functions have been implemented and work. It is reasonable to assume that there are activities, risks, and mitigations are valid inputs but not in the knowledge base (section 3.1.2), therefore there will likely be input strings that are well worded but will return low similarity scores as against the knowledge base as there is no similar piece of text stored. This objective allows the data set to grow, meaning if the non-vague piece of text that initially scored a low similarity score is seen again, there is an item in the knowledge base similar to it.

Objective 2: Identify vague input text. This objective is fulfilled when a piece of input text can reliably be determined to be vague or not vague. There is

no set method of measuring vagueness so I will have to rely on my personal opinion and examples of results to determine whether the decision is reliable or not.

Objective 3: Identify ambiguous input text. This objective is fulfilled when a piece of input text can reliably be determined to be ambiguous or not ambiguous. Again there is no standard method of measuring ambiguity and I will have to rely on my personal opinion and examples of results to determine whether this objective has been met.

Objective 4: Vagueness and ambiguity be determined as quickly as possible. The criteria for this objective is that any input text to the algorithm being determined to be vague or ambiguous within 5 seconds, I would consider anything over this time limit to be a failure. This objective is important for a users experience of a system, the result for a piece of texts vagueness should be returned as quickly as possible so the user is not frustrated at the time it takes my implementation to process their product. If the results take too long to return a user may even choose to by pass my vagueness and ambiguity filter, defeating its purpose and likely resulting in a lower quality of recommendations.

2 Literature Review

In this chapter, I will explore pieces of work and literature that have helped me with my project. I will provide a background to the different methods I considered using and an explanation of the theory behind them.

2.1 Understanding Vagueness

Before exploring Natural Language Processing methods, I felt it was important to develop my understanding of vague language, both from a linguistics and computational standpoint.

In Linguistics the concept of vagueness has been studied for centuries, with linguists exploring both how its work and its importance to language. The book "Vague Language" by Joanna Channel [11] has been a core piece of material I used to develop my understanding of what vague language is and how it is viewed and understood in the field of linguistics. The book was incredibly useful in regards to defining vagueness and providing examples of vague sentences and explores many language features that occur in vague language. The book was an important resource in assessing the viability of a rules-based approach (section 2.4).

In NLP the most up to date work on detecting vagueness focus on specific domains, like fake news [12], privacy policies [13], instructional texts [14], and software requirements [15]. By limiting vagueness detection to a specific domain, vagueness itself becomes easier to define and data sets easier to collect, this was a key motivation behind limiting my project to the construction domain. There is no conventional method or architecture to detect vagueness in NLP, [12] [13] [14] [15] all use different techniques to varying degrees of success. WikiHowToImprove [14] is the piece of work that came the closest that I could find to being a "general purpose" vagueness detection tool. It used information about revised sentences in WikiHow articles to train a Biderectional LSTM model (section 2.6.3) to determine which sentences in a sentence pair were the original and revised text, with the assumption the original text was more vague than the revised text. The trained model was able to predict which sentence was the revised one with an accuracy of up to 75.4%, whilst the task it achieved these results on is different to mine it is encouraging to see a computer achieve statistically significant results when identifying which sentence is less vague.

2.2 Readability Metrics

Readability metrics attempt to assign a value to how easy a sentence is to read. They have a variety of industrial applications and there has been a push by many organisations to make text more accessible and readable [16] [17]. I focused on readability metrics that provided an absolute score the the readability of a sentence, not an estimate of the reading age (such as the popular FOG index [18]). Two widely used readability metrics that provide an absolute score for the readability of a sentence are the SMOG Readability Grading and Flesch Reading Ease Score.

SMOG, standing for "Simple Measure Of Gobbledygook", is an algorithm proposed by Prof. Harry McLoughlin in 1969 [19]. The algorithm takes 30 sentences from a piece of text and counts the number of words in these sentences that have more than 3 syllables, then add the square root of this number to three. Like Flesch, SMOG Readability Grading is widely used, one study [20] that looked at the accessibility of online healthcare information concluded that SMOG was the "Gold Standard" of readability metrics. The Flesch Reading Ease Score [21] assigns a score measuring the readability of a sentence, this score ranges between 1 and 100 with 100 being the highest (easiest to read) score. The formula (shown in appendix figure .1) was developed in the 1940s by Dr. Rudolf Flesch, who revised his PhD thesis into the simpler formula. It uses the average word length and the average sentence length of a piece of text to derive its readability. The score is widely used even today, with the original paper having 5403 citations it is widely regarded as a good metric for measuring readability, although is generally considered to be better at assessing children's reading material than material targeted at adults. It also came under criticism from the creator of the SMOG algorithm for not considering the whole length of a sentence.

2.3 String Simplification and Comparison

Words can take many forms depending on the contexts or tense they are being used. Words can also be derived into families that revolve around a core idea, such as *democracy*, *democratic*, and *democratise*. This observation can help us understand and compare pieces of text by simplifying the words in the text using the techniques of stemming and lemmatisation. Stemming reduces a word to its root by removing its suffixes. For example, the words "connected", "connection" are all stemmed to "connect". It is often used in information retrieval processes to reduce text complexity and focus on meaning. The Porter Stemming algorithm [22] is the most common tool used today, and it fragments words in consonant and vowel groups that are used in 5 different stemming steps to decide whether a rule should be applied (see appendix .3). Lemmatisation [23] is like stemming in that it breaks down words into their meaning, but instead of stemming to the "root" of the word (which often is not a word that exists), lemmatisation breaks down a word into a valid word.

Lemmatisation achieves this by looking at the meaning of the word as opposed to just its form (as in stemming). Once two strings has gone through the stemming or lemmatisation algorithm, Levenshtein distance can be used to calculate the similarity between them. The metric, which can be calculated using the formula in appendix figure .16, counts the number of characters that would need changing for two strings to match, meaning more similar strings return lower distance.

2.4 Rules-Based Approach

Before more recent improvements in machine learning and language models, rules-based approaches were the best way for a computer to handle natural language and they can still outperform trained networks in specific tasks. Rules-based approaches rely on huge libraries of language rules, these libraries are created by humans and each rule is only applicable to a specific grammar or word rule. Rules-based approaches generally perform best when looking for patterns in specific contexts, medicine is one of these contexts with many pieces of work focused on finding rules to unlock new insights [24] [25] [26]. These pieces of work adopt rules-based approaches as a lot of the text they analyse is standardised and they are searching for key terms that can easily be written into rules. Yao et al [26] used a rule based approach to NLP to find trigger words and phrases in a patient's discharge summary and classify them.

The book "Vague Language" by Joanna Channel [11] identifies multiple features of vague language that could be turned into rules for identifying it. For example, the book highlights how a vague sentence may approximate quantities with non-numerical quantifiers. A rule for detecting this kind of vagueness would be to negatively impact the score of pieces of text containing phrases like "a lot", "a bit", or "a few".

2.5 Word Embedding

Representing words as strings is problematic, the calculations computers are designed to do, such as addition and multiplication, cannot be meaningfully used on strings. Word embedding proposes a solution to this problem by storing words as vectors whose direction embed a semantic meaning. A team led by Tomas Mikolov revolutionised word embeddings when they proposed a way the train feed forward networks to create semantically meaningful vector representation of words using a technique they called word2vec [27]. Embeddings created networks trained using Mikolov's methods are so powerful that meaningful mathematical operations can be performed on the embeddings, for example subtracting the vector embedding of the word *Man* from the embedding of the word *King*, then adding the embedding of the word *Woman* results in a vector that is very similar to the embedding of the word *Queen*.

In word2vec, word embeddings are created by a feed forward neural network, Mikolov proposed two architectures to train networks to generate these embeddings (shown in appendix figure .4), Continuous Bag Of Words (CBOW), and Continuous Skip-Gram. The CBOW architecture trains a network to predict missing words form a sentence/phrase, the models input is a "Bag of Words", meaning a one hot encoded vector where each position represent a word in the dictionary, the output is a prediction of which word in the dictionary is most likely to be the missing word. The Continuous Skip-Gram architecture trains a network to predict the words surrounding an input word, where the input is the one hot encoded vector of the word and the output is a probability that each of the words in the dictionary come before in specific position before or after the input word. An analogy the helped me understand this architecture is that the training process is like a teacher performing an "egg drop" challenge with their class [28], the teachers ultimate goal is not to have all the eggs survive, instead it is for the class to learn the fundamental physics ideas that can help prevent the egg from cracking. The same idea is applied in these architectures, the task they are evaluated on is not to represent words as effectively as possible, but by improving their performance on the task the hidden layers of the network learn how to represent each of the possible input words.

2.6 Transformers

The transformer is an architecture that can be used to solve sequence to sequence tasks like language translation and time series analysis. Attention Is All You Need [29] is the seminal piece of work on the topic, it presents the novel transformer architecture and the self attention mechanism as a solution to problems faced when using other language processing models such as a Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM). The paper achieved groundbreaking results when translating English to German and French, both in the quality of translation achieving 28.4 BLEU on an English to German translation task, and in the reduction of training time and resources to achieve these state of the art results.

2.6.1 Self Attention

Transformers use the mechanism of self attention to achieve these ground breaking results. Attention mechanisms focus on parts of the input sequence and compare their importance with parts of an output sequence, an attention mechanism for English to French translation would say an input word "red" is important for the output word "rouge". Self attention builds on the idea, but instead of the attention being between an input and output sequence, the focus is relating parts of the input sequence to other parts of the input sequence (the focus stays within the input). The self attention mechanism computes a representation of the input sequence that describes the relationship between different parts of of the input sequence. Vaswani et al introduced Scaled

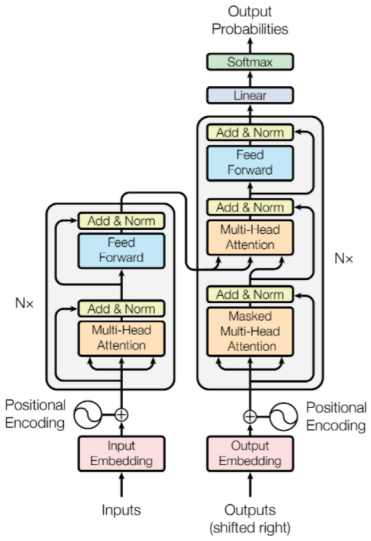
Dot-Product attention in [29], a multi-headed version of scaled dot-product attention is used by the transformer architecture to implement the idea of self attention, how this works is explored in sections 2.6.2.

2.6.2 Transformer Architecture

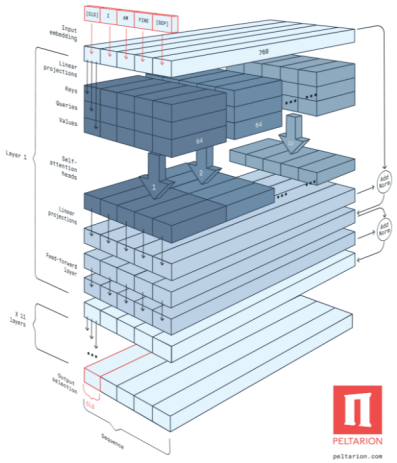
The transformer architecture (shown in figure 2.1a) consists of two distinct parts, an encoder and a decoder. For my project I am only concerned with using the encoder to create a meaningful encodings of an input string. The encoder architecture consists of multiple feed-forward neural networks that are responsible for individual tasks within the process breaking down a string to a meaningful encoding. First word embeddings are calculated for each of the words in the input string, these are combined with positional encodings of the sentence that provide the model with information about where in a sentence each word appears. The combined word embeddings and positional encodings are fed into a multi-head attention layer, this is a series of parallel feed-forward networks use the self attention mechanism to weight each word's importance relative to other words in the sentence, figure 2.1b includes a brilliant visualisation of this layer. Having multiple heads means attention can focus on different things within the sentence, for example one head may focus on how a word relates to the words in its immediate vicinity, whilst another head focuses on how a word relates to words further away. The multiple attention head outputs are concatenated, normalised, and fed into a feed forward network whose main purpose it to transform the attention vectors into a form that is acceptable for their next purpose, for example a form that is acceptable for the decoder in a full transformer network.

2.6.3 Transformer Alternatives

There are alternatives to transformers when training networks to handle natural language, however these do not perform as well as. Recurrent Neural Networks (RNNs) [31] are feed-forward networks that enable previous outputs to be used as inputs, this means that an input word can be considered in relation to all words that came before it. RNNs do not perform as well as transformers as they cannot consider word positions in ahead of the input word, their architecture also often results in exploding and disappearing gradients during training. Long Short Term Memory (LSTM) [32] neurons were proposed as a solution to RNNs, they still use previous inputs to get meaning about the context of a word in a sentence, however they perform fewer calculations on the previous inputs meaning exploding and disappearing gradients are less common. Bidirectional LSTM [33] improves on LSTM by working through a sentence from both start to end and end to start, meaning it can consider the how words that come later in a sentence impact earlier ones. However, the self attention mechanism still performs better than Biderctional LSTM when gathering information about a word relative to other words. LSTM is also slower



(a) Transformer Model Architecture [29]



(b) BERT Encoder architecture [30]

Figure 2.1: Diagrams of the transformer architecture

to train than RNNs, which themselves are slower to train than transformers.

2.7 BERT

BERT, meaning Bidirectional Encoder Representation for Transformers, takes advantage of the transformers self attention mechanism by using only the encoder to generate a language model [34]. The "Bidirectional" part of BERT is a terminology continued from RNNs and LSTMs, it would be more proper to say BERT is "non-directional" as the self attention mechanism means it considers all parts of a sentence at once. The architecture of BERT is based on the encoder model of transformers described in section 2.6.2.

BERT was groundbreaking in how the framework could be trained in two steps, first a model is "pre-trained" with unlabelled data to develop a general understanding of language, next the model is "fine-tuned" by using labeled data to tweak the model for a specific task. This framework meant that incredibly powerful language models could be trained using vast amounts of resource, then tweaked and applied to specific tasks using much smaller data sets and without the need for the language model to be learned from scratch. BERT has been applied to a variety of tasks, such as sentiment analysis [35], next sentence prediction [36], and question answering [37], a downfall of BERT however is that to perform text comparison tasks (such as semantic similarity), it requires two sentences to be fed into the network to so that they can be compared, as a result semantic lookup tasks become time consuming, computationally expensive processes [38].

2.8 Sentence-BERT

Sentence-BERT [38] (or SBERT) is an implementation of the BERT algorithm that only requires one input sentence, compared with BERT which requires two input sentences for tasks that involve comparison. SBERT is much quicker at performing a semantic similarity look ups. Reimers et al [38] provided the example of finding the most similar sentence pair in a collection of 10,000 sentences, using a standard BERT model each sentence pair would have to be evaluated together, requiring around 50 million inference computations and taking about 65 hours on the right hardware (e.g. a V100 GPU), on the other hand the SBERT can calculate the encodings for each sentence in the collection individually then compare them, taking around 5 seconds.

SBERT adds a pooling operation to the output of a BERT network to create a fixed sized vector (normally 768 dimensions), the paper experimented with three pooling operation, using the output of the CLS-token, calculating the mean of all output vectors, and calculating the max-over-time value of the output vectors. A "siamese" network is used to fine tune the BERT network, shown in appendix figure .5. The exact structure of the network varies on the available training data (this is why there are two separate architectures in the figure), the network may use a Classification Objective Function (left of figure) to optimise cross-entropy loss whine fine-tuning BERT for tasks like sentence classification, or a Regression Objective Function (right of figure) to optimise the mean-squared-error loss for tasks like measuring semantic similarity. A triplet network can also be used for tasks like sentiment analysis, however this is not relevant to my project. Cosine similarity is the recommended function to calculate the semantic similarity between two sentence encodings. It compares the angle between two vectors returning a score between -1 and 1, where 1 means the vectors identical. The formula for calculating cosine similarity is shown in appendix figure .2.

SBERT outperforms state of the art models, both in measuring semantic similarity and in computing performance. As previously mentioned it drastically improves BERTs computational needs, however it is also faster than other state of the art methods, it is 9% faster than InferSent [39], and 55% faster than Universal Sentence Encoder [40]. Fine-tuning pretrained BERT models using the siamese network architecture also outperformed other state of the art techniques at Semantic Text Similarity (STS) benchmark tasks. A fine tuned SRoBERTa-NLI-large model performed the best overall on the STS tasks, narrowly beating a fine tuned SBERT-NLI-large model. Scores were calculated using Spearman rank correlation ρ between the cosine similarity of sentence representations and the "gold labels", SRoBERTa-NLI-large scored an average of 76.68 across the tasks, and SBERT-NLI-large scored 76.55, beating Universal Sentence Encoder which scored 71.22 and InferSent which scored 65.01, as well as a non fine tuned BERT model.

3 Data Analysis

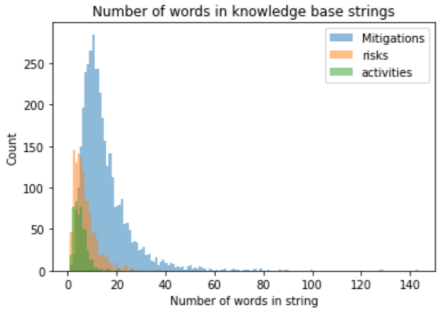
In this chapter, I will break down my project through the lens of the data I will be using to complete it. I have two data sets, one which I will be referring as the "Knowledge Base" (section 3.1) will be used in my algorithm as the source of knowledge about the construction domain. My second data set is my "testing data" (section 3.2), I will be using this to test my algorithm. Unfortunately, I am unable to submit the full knowledge base or testing data as it is under Intuety's intellectual property, however I can be contacted using ab2474@york.ac.uk to verify any claims and small samples of the data can be found in the submitted jupyter notebooks and appendix.

3.1 Knowledge base

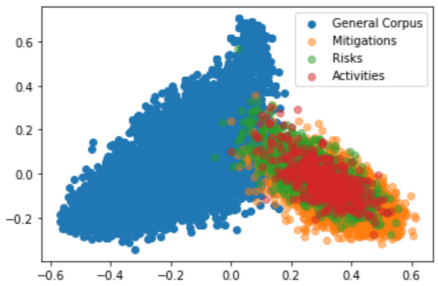
My knowledge base has kindly been provided by Intuety, it consists of 5594 pieces of text that can reasonably found in a construction site's Risk Assessment. These pieces of text are categorised into Activities, Risks, and Mitigations as described in section 1.2.2, and the data set is regularly updated and maintained by a team of humans with construction domain knowledge. I am using a partial copy of Intuety's data, my data is stored in a Neo4j Aura graph database [41] (section 3.1.1). Intuety themselves use a different graph database solution so if they were to adopt my algorithm some of the queries would need to be updated to a new query language. My graph database is made up of 5594 connected nodes (figure .6), each node (figure .8) is labeled as an "Activity", "Risk", or "Mitigation", it stores a string containing a piece of text that could reasonably be in a construction site Risk Assessment, as well as a 768 dimensional vector encoding of the string. The text was taken from Intuety's database whilst the encoding was calculated using SBERT (chapter 4.3.1). The connections (relationships) (figure .7) between nodes can have one of two labels, "Risk_Of" or "Mitigation_Of", for this I am not storing any information on the connections.

3.1.1 Graph Databases

As my knowledge base is stored in a graph database I think it is important to explain what one is and why they are useful way of storing data. Graph Databases store data in nodes, then store relationships between nodes as connections that can be traversed. Nodes and their connections can store multiple pieces of information. Being able to store information in nodes and



(a) Graph of the number of words in the strings in my knowledge base



(b) Plot of PCA of Knowledge Base and General Corpus of text

Figure 3.1: Visualisations of my knowledge base

the connections between them is incredibly powerful. For example a graph database about footballers may have two nodes representing Joe Allen and Ryan Shawcross, a connection between the two nodes could then have the label "teammates" and store information about how they both played for Stoke City, and the number of times they passed the ball to each other. Intuety use a graph database as they store information for recommendations on these connections, however the connections themselves are not important for my implementation.

3.1.2 Knowledge Base Contents

Each node in my knowledge base stores a string for the text it represents in the field "text" (appendix figure .8), as well as a label of what the text is, e.g. an activity, risk, or mitigation. Each string has been stripped of unnecessary punctuation and put into lower case, this is because extraction from documents can often leave unusual deformities in the text, such as random full stops or capital letters. These deformities are remove by putting the string into lower case and removing unnecessary punctuation, therefore it makes sense to store the knowledge base data in the same way.

The knowledge base was built in two stages, initially a huge amount of text was ingested in an unsupervised fashion, after this the knowledge base grew naturally, where previously unseen pieces of text are put into a separate database to be evaluated by a human and either discarded or added to the knowledge base. The majority of string in my knowledge base are under 20 words long, figure 3.1 shows the distribution of the number of words in each string, a break down of this graph can be found in the appendix .9 .10 .11. My activities and risks and generally much smaller than my mitigations, this is to be expected as mitigations often have to be more detailed, describing exactly how the risk should be controlled.

My text almost entirely relates to the construction domain, this can be demonstrated by performing Principal Component Analysis (PCA) on the sentence

embeddings (calculated in my implementation 4.3.1) and generic corpus of data [42], then plotting the outcome (figure 3.1). This plot show the knowledge base data is clustered with a similar semantic meaning, which is also different to the majority of text within the general corpus. Strings within the database are regularly checked and pruned, users of Intuety have the ability to report problematic strings to a customer support team who will either update the string or remove it from the knowledge base. I feel confident that this constant human monitoring means the knowledge base is a reliable source of construction specific text. PCA of the knowledge base and a general corpus of text shows the semantic meaning of the of my text is also unique to the construction domain and similar across labels.

Despite being regularly checked by humans, there are issues with my knowledge base I feel it is important to identify. One of these issues is that my knowledge base does not contain every activity, risk, or mitigation from a construction site possible. There will be perfectly valid inputs that are not vague but will be flagged as vague by my algorithm as there is no similar piece of text in the knowledge base. For this reason my knowledge base needs the ability to grow to include these previous unseen triples. Another issue is that poor quality pieces of text are only ever really highlighted when they are recommended, this means there is likely some data in my knowledge base that is of such a poor quality that it may never be recommended and therefore highlighted for removal. Unfortunately Intuety does not have the time and resources to go through every single piece of text in its knowledge base and evaluate it manually to remove the problematic one that haven't been recommended. I am hoping that as they have never been recommended, these problematic piece of texts will also not be too similar to anything I am testing, however it is still something I needed to be aware of when developing and testing my algorithm.

3.2 Testing Data

My testing data is taken from a database of text that has been extracted from Risk Assessments, more specifically it is taken from a the subset of all extracted rows that contain text that is not in the knowledge base. This extracted text is in the format that would be used to provide recommendations, and it is labeled as an activity, risk, or mitigation based on the column it was extracted from. In total my testing data is made up of 55063 pieces of texts, which break down 3652 activities, 11090 risks, and 40321 mitigations.

The testing data was extracted in an unsupervised fashion and has not been checked by a human, this means that compared with my knowledge there is a higher probability that data is mislabelled and contains spelling mistakes. However my final algorithm's goal is to catch poor quality inputs, regardless of whether the inputs were extracted wrong or written badly initially. Therefore, I consider it to be a good thing that my testing data is not perfect, as it is these imperfections I am hoping to catch.

4 Design and Implementation

In this chapter, I will describe the various approaches I considered to fulfill the goal and objectives I set out in section 1.3, as well as describing in detail my final implementation.

4.1 Considered designs

I considered multiple possible implementations that I felt could fulfill my goals, in this section I will describe and critique the different implementations I considered. I will only focus on how the design would be implemented to suit my goals, the literature review (chapter 2) contains more information on the background of the considered techniques.

4.1.1 Readability metrics

I explored using readability metrics to measure vagueness, thinking that the easier to read a sentence is, the less vague it is. Initially I assumed this fit the definition of vague I provided in section 1.2.3, as the harder text is to read, the harder it is to interpret its meaning. However I found that in regards to my project, readability metrics are not useful, scores can easily be manipulated by choosing shorter words, and they contain no real information about the meaning, and as a result vagueness, of a piece of text, this is demonstrated in appendix figure .12. I have chosen not to use the Flesch Reading Ease Score or the SMOG Readability Grading in my project. Readability metrics may still provide some use to an extension of the work done in my project. Well written risk assessments should both be readable and not vague. If further work were to explore using the metrics identified in my literature review, I recommend it uses the Flesch Reading ease score, this is because construction risk assessments may unavoidably contain technical polysyllable words, which SMOG would punish. On top of this my knowledge base contains many small phrases and even single word items, this is acceptable in risk assessments, however SMOG requires larger pieces of text and full sentences to perform at its best.

4.1.2 String Simplification and Comparison

String simplification methods could be used in my project to simplify input text and my knowledge base items before they are compared, this would likely result finding more similar sentences than just comparing non-simplified pieces of texts. I tested both stemming and lemmatisation techniques on my knowledge base to see how they each performed. These tests highlighted the issue of stemming breaking down technical words into roots that no longer represented its meaning, lemmatisation only breaks down words into words that exist so it is far less likely to encounter this problem. This issue is demonstrated in appendix figure .13 where the activity "piling" is broken down by the stemmer in to a root "pile", completely changing the meaning of the activity, as a result I came to the conclusion if I were to use a string simplification technique it would be lemmatisation. The design I eventually considered involved leammatising a word then comparing it to the knowledge base using levenshtien distance to see which item it was most similar too. This technique works well on strings that use the same wording and sentence structure, but it is unreasonable to assume all pieces of text I see will uses similar wording and have similar writing styles. The technique does make sentences less complicated and easier to handle, but on its own it does not provide any information about the semantic meaning of a sentence, if two phrases mean the same thing but use different words the simplification and comparison won't pick up on it. After realising this I chose not to use string simplification techniques.

4.1.3 Rules-Based Approach

My project is limited to a specific domain (construction), and in theory a lot of risk assessment text should follow similar patterns, meaning a rules-based approach should preform well on a task like mine. Unfortunately when exploring a rules-based approach to fulfilling my goals I found it was difficult to write rules to detect vagueness. Whilst I thought the language features Channell identifies [11] would be useful, I found that they were not for my goal. They can be used to detect general vague language, however fall short when it comes to the specific domain of construction. On top of this many of my data points are single words or short phrases, there is not text in most of them to activate any general rules. A piece of text is vague when it has no meaningful features, making writing any form of rules an incredibly difficult task. An alternative to writing rules that detect vagueness would be to write rules that determine whether an input matches the type of text in my knowledge base, and reject any inputs that don't. Unfortunately the task of writing rules to encompass my entire knowledge base would require more time and resources than I have available to me. Additionally, my first objective is to have the ability to grow my knowledge base to include valid pieces of text that are not already in it, a rules-based approach that looked for patterns similar to the knowledge base would require a new rule every time the knowledge base is expanded, making maintenance a constant task.

4.1.4 Sentence-BERT

A big issue with my previous designs was that they could not evaluate and compare semantic similarity. SBERT is currently one of the best performing models at extracting semantic meaning from a sentence, and its output is a fixed sized vector which can be compared to other output vectors using cosine similarity. Many pretrained models have even been trained using a fitness function that maximises how accurately it can perform semantic similarity. I had the choice between training a SBERT model from scratch, or importing a pretrained model. My data set is far too small to train a useful SBERT model on its own, meaning I would have to find a useful general corpus to train a model with. The time and resources available to me are very limited, especially when compared to the resources used to train some of the available pretrained models, therefore I feel it makes sense to use a pretrained model. Other people have far more resources available to them to train language models than I do, they will also already have found a large and useful text corpus to train an SBERT model. This will result in a model that performs better than anything I would be able to train with my resource and time restrictions. Fortunately there are resources that compare the best performing SBERT models [43], meaning I can easily assess which models are perform the best and are suited to my task.

Using an SBERT model also presents me with the opportunity to fine tune a pretrained model for my specific task, which theoretically results in a better performing model. One of the advantages of fine-tuning a pretrained model is that it does not require a huge amount of data, however the data typically needs to be labeled for the task at hand, which my current knowledge base is not. I do not have the time and resources to label my data, so will have to assess how a pretrained model performs, it would be interesting to see how any future work that uses a BERT model fine tuned to identify vagueness against a data set compares.

4.2 Data Preprocessing and Analysis

I had to ingest the data Intuety provided into my Neo4j Aura instance, I took this opportunity to clean the data ensuring all text has the same form. This included stripping the text of most punctuation, removing any additional white space (including new lines), and putting all text in to lower case. Generally you want to leave punctuation and casing in a piece of text when passing it into BERT as it can provide information on the context of words within the sentence, however, due to the extraction process that gets the texts from risk assessment documents, Intuety's data has a lot of "dirty" punctuation in sentences. This "dirty" punctuation looks like punctuation that occurs in places it should not, e.g. a hyphen at the end of a sentence. The reason for the error may be down the original document containing grammatical errors or the extraction process making a mistake, but as it occurs so often in the data set I

was given I decided it is better to remove most forms of punctuation leaving only full stops, commas, and apostrophes. This prevents any encoder was not being thrown by it. The same rules that were used to clean my knowledge base data were also used to clean any input text to the algorithm, however this cleaning was not permanent and only took place for the lookup. Once my knowledge base had been entirely populated with the data Intuety provided, I got the encoding for each individual piece of text and stored it in the node, this encoding was calculated with using the same model I went on to use in my final implementation.

4.3 Final Implementation

In this section, I will describe my final implementation, the code for my final implementation can be found in the supplementary files of my final submission, as my knowledge base and testing data include Intuety's intellectual property the code will not be connected to a functioning Neo4j aura instance, for a demonstration of the working code please contact ab2474@york.ac.uk . My final implementation was programmed in python, it used the pretrained all-mpnet-base-v2 SBERT encoder to calculate sentence embeddings, a Neo4j Aura Instance to store my data, and cosine similarity to calculate the similarity between vectors.

4.3.1 SBERT Encoder

My final implementation used the pretrained all-mpnet-base-v2 SBERT encoder [44], the encoder is a based on the microsoft mp-net base model [45] and was fine tuned on a data set containing over a billion sentence pairs (1,170,060,424 to be precise). The fine-tuning was done with a contrastive learning objective meaning the model was given a sentence from a pair and told to select its partner from a randomly sampled set of sentences. The model was trained on 7 TPUs v3-8, it was trained during 100k steps using a batch size of 1024 (or 1028 pre core), the hardware, parameters, and data set used to fine tune the mp-net model hopefully show how far off any model trained on my resources in the time I had available to me would have been far from state-of-the-art, and given the target of my project was investigate whether transformer similarity could be used to give information about the vagueness of a sentence, and not to build and train my own encoder from scratch, it makes sense to use a state of the art model. The model uses mean pooling [46] to pool the BERT output into a 768 dimensional vector, and has a maximum input sequence length of 384 words, which is longer than any string in my knowledge base.

The only other pretrained model I considered using was multi-qa-mpnet-base-dot-v1 [47], the model also fine tuned the mpnet-base pretrained model. However, it was tuned specifically for semantic search by training on a set of 215

million question/answer pairs using multiple negative ranking loss function [48] and uses CLS-Token pooling [46]. The multi-qa-mpnest-base-dot-v1 model performed slightly better than the all-mpnet-base-v2 model on semantic similarity tasks [43], scoring 57.60 where all-mpnet-base-v2 scored 57.02, however the model was considerably outperformed on the sentence embedding tasks scoring 66.76 to all-mpnet-base-v2's 69.57, both models are the same size and run at the same speed on identical hardware. The multi-qa-mpnet-base-dot-v1 model has a maximum input length of 512 which is longer than all-mpnet-base-v2, however both are able to accomodate the longest string in my knowledge base and as a result capable of handling the majority of input strings my algorithm will encounter.

I decided to use the all-mpnet-base-v2 model in my final implementation as it was the best performing model for sentence encoding tasks. Other implementations performed better at semantic similarity tasks I was concerned about the obscurity of my domain (construction risk assessments), unfortunately no publicly available model has been fine-tuned using specifically construction industry text. I felt it was better to use a model that has been fine tuned using as many pieces of text as possible, no model has been trained explicitly on construction domain text however both models I identified were trained on text taken from public forums, there is a slim chance some of these discussions contained information about the construction industry, training on more text from public forums increases the possibility the model is exposed to text similar to what is found in risk assessments.

4.3.2 Similarity Lookup

I am performing similarity lookups against my knowledge base, this is achieved by comparing the input sentence encoding to the sentence encodings of the items in the knowledge base using cosine similarity. As mentioned in section 4.2 I have precalculated encodings for ach of the items in my knowledge base, this saves time and computing power on each lookup as only the encoding for the input sentence needs calculating, The result of the two highest similarity matches are saved to be used to determine if the input text is vague or ambiguous.

Identifying Vagueness: By applying my definition of vagueness from section 1.2.3 to this implementation, I decided that a vague sentence would be a sentence that when compared to all of the sentences in the knowledge base, does not have a high cosine similarity score for any them. I implemented this with the plan of finding a threshold for when a sentence can be considered vague, this threshold is discussed in section 5.2.3.

Identifying Ambiguity: By applying my definition of ambiguity to the implementation, I decided that an ambiguous sentence would be a sentence that when compared to the knowledge base had a considerably high cosine similarity score for more than one of items, where the score was above the vagueness threshold and within a certain limit of each other. I discuss the limit on the

closeness of similarity score in section 5.2.3. The score for the ambiguity of the sentence would be $s_1 - s_2$ where s_1 is the highest cosine similarity match and s_2 is the second highest similarity match.

4.3.3 Graph Database Functions

My knowledge base was stored in a Neo4j aura instance, this is a free hosted service provided by Neo4j and uses the cypher query language [49], an example of a cypher query is shown in appendix figure .17. All of my graph database actions are implemented in a class called "GraphHelper", the class is instantiated with the connection settings of an aura instance, this decision improves the security of my implementation as connection details only need to be accessed once, it also improves the ease of development as every lookup can have a consistent style and be accessed within the same object. The GraphHelper class has functions to fetch every node with label activity, risk, or mitigation, ingest new text using a staging label of its type, and update nodes with a staging label to have the production label of their type. I chose to ingest new text with a separate label to those used in my implementation look ups as so that it can be stored for review, a node with label "StagingActivity" may be a piece of text not previously seen by the system, once it is reviewed by a human it can be deleted if it is a bad piece of text, or my function that changes staging labels to production labels can be run, this means the label of the node will be changed to "Activity" and it will be returned in future look ups and used. When new text is ingested, I first check there is no identical text within the production database, then utilise the cypher "MERGE" function to prevent duplicates within my staging text. In my example code I store look up results locally because the lookup to the graph database is one of the most time consuming parts of my implementation, this is the equivalent to caching lookup results in a production environment, the performance improvements of caching database look ups can provide are discussed in section 5.4.

5 Results

In this chapter I will analyse the performance of my final implementation using the testing data set (section 3.2). All graphs and data are further supplemented in the appendix.

5.1 Analysis of Evaluation Methods

Unfortunately, there are no existing metrics for evaluating vagueness and ambiguity or how good a method is at detecting them. Standard metrics exist that evaluate the performance of an NLP implementation's performance on specific and general tasks using large public data sets, however none of these evaluate vagueness or ambiguity specifically and are of no use to this project. BLEU [50] is a popular metric that evaluates the performance of a model at language translation tasks, it has also been applied to general reading comprehension tasks [51], however there is no way I to adapt this metric to evaluate vagueness or ambiguity, therefore it provides no insight into the accuracy of my implementation.

GLUE [52] and SuperGLUE [53] are benchmarks that evaluate a language models performance on a variety of tasks to gain an insight into the models general natural language understanding. The metrics heavily influenced my decision of pre-trained language model, however a break down of the individual tasks used in both GLUE and SuperGLUE shows that none of the tasks evaluate a models ability to identify vagueness or ambiguity. Furthermore it would make little sense to run a pretrained model that has not been fine tuned through any of these tests as I already know what the outcome will be.

My testing data is not labelled with information about each strings vagueness, if it was I could have calculated information about the precision of my algorithm using its accuracy at predicting the information on vagueness, supplementing this insight with figures such as the root mean squared error or mean average precision.

Initially I had planned to conduct a survey to evaluate how my implementation performs in comparison to a human understanding of vagueness and ambiguity, however due to exceptional circumstances I ran out of time to the check the survey with Intuety, apply for ethical approval, and gather a reasonable number of responses.

I have settled on evaluating the distribution of similarity and ambiguity scores to gain an insight into my implementation's accuracy and to help me decide

where any thresholds should go. I utilised Principal Component Analysis visualisations to gain an understanding of what vagueness and ambiguity look like. Finally I used anecdotal evidence to explore issues relating to measuring vagueness and ambiguity using semantic similarity against a data set.

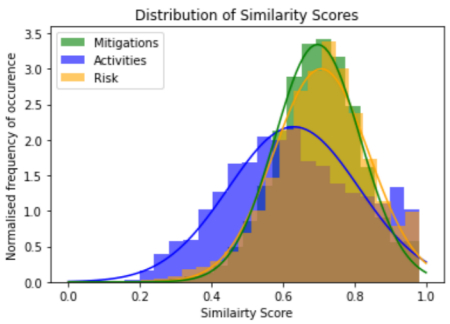
5.2 Distribution of testing results

Analysing the distribution of my testing results will give me an idea of the mean and variance of the scores I can expect to see during everyday use of my implementation. In this section I will analyse the distribution of similarity and ambiguity scores, discuss its implications regarding my criteria, and attempt to find a threshold for vagueness and ambiguity.

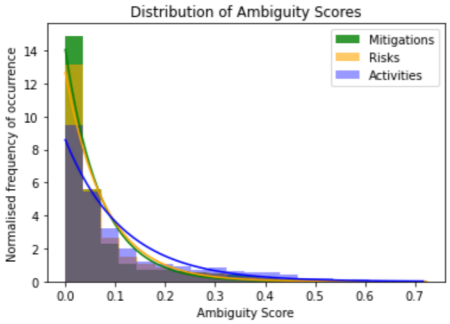
5.2.1 Distribution of vagueness scores

Vagueness will be detected using the similarity score of an input piece of text against the knowledge base (section 4.3.2) and a threshold (section 5.2.3). The distribution of similarity scores is shown in figure 5.1a, and the individual graphs are shown in appendix .18, .19, and .20. Before creating the graphs I removed any identical matches from the testing data, in total 442 activities were removed whilst 3210 were plotted, 1173 risks were removed whilst 11090 were plotted, and 4074 mitigations were removed whilst 36247 were plotted. I considered an identical match to be any string that matched an item in the knowledge base with a similarity score of more than 0.98, whilst this technically means the matches were not always completely identical strings, I found that the majority of strings that that produced a similarity score of at least 0.98 could be considered identical except for some punctuation or a single unimportant word. For example the string *"confined space – communication during confined space working"* matched with the string *"confined space communication during confined space working"* with a similarity score of *0.9805507063865662*. Identical matches would never be considered vague or ambiguous, so by removing them I could focus more on the kinds of matches that would make a difference to a deployment of my implementation.

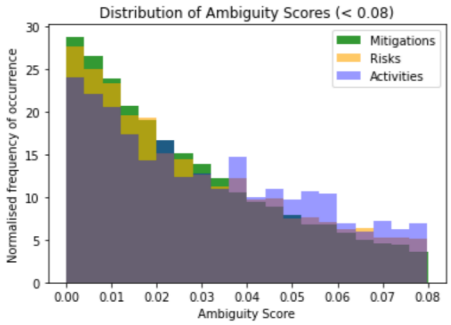
The plots for each of my labels resemble a bell curve, therefore I have fit the data for each label to a normally distributed Probability Density Function (PDF) and have included this plot in the same figure. The PDF also gives me an insight into the mean (μ) and variance (σ^2) of my results. Analysing the plots, it is clear that the distribution of my activity results has a greater variance and lower mean that the distribution of my risk and mitigation results, and the risk and mitigation results have a similar variance and mean. These observations are supported by the PDF for each label, for my Activity results $\mu = 0.6299968953305316$ and $\sigma^2 = 0.0333450242330677$, for my Risk results $\mu = 0.7067183941163951$ and $\sigma^2 = 0.017657975674763116$, and for my Mitigation results $\mu = 0.6965272417503459$ and $\sigma^2 = 0.014226491868421554$.



(a) Distribution of similarity scores



(b) Distribution of ambiguity scores



(c) Distribution of ambiguity scores < 0.08

Figure 5.1: Distributions graphs

The reason testing data with the label activity has a lower mean and greater variance is likely due to issues faced during the extraction process, in practice not all Risk Assessment tables contain an activity column and when this is the case the Intuety product looks for a "Global Activity", meaning an activity that is applicable to the entire document. The process of finding a global activity is not as reliable as extracting risks and mitigations from Risk Assessment table columns, resulting in incorrect pieces of texts being extracted with the label activity much more often than they are with the labels risk or mitigation. Taking this into consideration it is actually a positive result that the more problematic testing label has a lower mean. It is evidence that can be used to support the hypothesis that similarity results can be used to identify problematic inputs, as the label with the most problematic inputs has the lowest mean.

5.2.2 Distribution of ambiguity scores

The process for calculating ambiguity scores is described in section 4.3.2. Figure 5.1b shows the distribution of ambiguity score results, it appears to show that the shape of the distribution for each label is similar to that of e^{-x} (a decreasing exponential function), where the most common ambiguity scores are those closest to zero. In the histogram of the full distribution (figure 5.1b) the majority of ambiguity scores are in the "bins" for scores less than

0.1, therefore I took the time to plot only the ambiguity scores less than 0.08 (figure 5.1c) which showed a similar pattern where ambiguity scores were more likely to be closer to 0. The plotted distributions also showed that the testing data labelled as activities produces a greater variance in results and on average has a higher ambiguity score than the risk and mitigations. This observation is supported by fitting my results to an exponential PDF, and testing inputs with the label activity produces an exponential distribution where $\mu = 0.11637902279003949$ and $\sigma^2 = 0.013544076945564532$, whereas text with the label risk fits an exponential PDF where $\mu = 0.07896982506925292$ and $\sigma^2 = 0.006236233271468407$ and text with the label mitigation fits a PDF where $\mu = 0.07112153582516054$ and $\sigma^2 = 0.005058272858129593$. The reasons behind this observation are most likely explained by the fact that there are considerably more mitigations and risks in the knowledge base than there are activities, which means there are more potential matches for an input string to have. The mean and variance of the test activity inputs will also be affected by previously highlighted issues with the extraction of activities.

I feel this distribution does not support the idea that ambiguity can be measured successfully using the difference between similarity scores of input text against my knowledge base, instead it is evidence of the contrary. The majority of my testing data is not ambiguous, however the shape of my histogram implies the most probable ambiguity scores are those closest to 0.

5.2.3 Attempting to find a threshold

Any production deployment of my implementation would likely require a threshold to determine whether a score was vague/ambiguous or not. This threshold will be important as it decides the leniency of what is considered to be vague or ambiguous, on one hand setting the threshold too high may result in user frustration as a large amount of their document may be flagged as vague/ambiguous, on the other hand setting the threshold too low could result in lots of vague/ambiguous inputs being used in the system, defeating the purpose of checking the vagueness and ambiguity in the first place.

I used a random sample of similarity scores from across the distribution to influence my decision of where a vagueness threshold should go, appendix figure .15 is a snapshot of the scores I used to influence this decision, these scores were taken into consideration along with the hundreds of similarity scores I encountered throughout development. I decided that my threshold should be between the similarity scores of 0.6 and 0.65, this decision was based entirely on my personal opinion and experience. There is no set technique for deciding where a threshold should go and no mathematics or statistical analysis that could be relied upon to make the decision for me. I investigated how these scores fit into the distributions of my similarity results and found that for both my risk and mitigation scores the bounds of the lower quartiles were within my desired range. Due to the test activity input distribution's high variance I decided not to use it to influence my decision

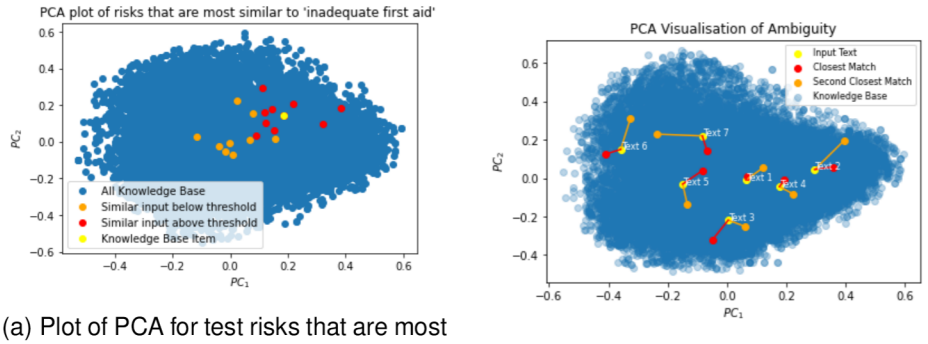
on where the vagueness threshold should be, this decision will likely result in more activity inputs being flagged as vague, but I feel this is due to an issue with the problematic extraction process not the threshold itself. As the mean and variance for the similarity score distributions of the test risk and mitigation inputs were so similar I decided to concatenate the two scores and use the lower quartile of the distribution of scores for both label as a global threshold for vagueness that will be used for every label, this threshold is a cosine similarity score of 0.6223 to 4 significant figures.

I decided that a threshold for ambiguity was not worth calculating, section 5.2.2 highlights issues with the distribution of ambiguity scores that show my technique for measuring ambiguity is problematic, my distribution would suggest that the distribution of ambiguity scores tends to 0 meaning any threshold would cut off the most probable ambiguity scores, an outcome that doesn't make sense to me.

5.3 Visualisation of vagueness and ambiguity

In figure 5.2 I attempt to visualise what vagueness and ambiguity matches look like using Principal Component Analysis. To create the visualisation I performed Principal Component Analysis (PCA) on the 768 dimensional sentence encoding for each of the items in my knowledge base, and as well as the encoding of the input text I am plotting. I then plot the first two principal components from the PCA calculation. The first two principal components explain approximately 8.718% of all variance within the sentence encodings, (PC_1 explains 4.750309% and PC_2 explains 3.967546% of the variance). The visualisations are not perfect tools, as over 91% the variance is unexplained the plots appear to show closer matches in the knowledge base than the highlighted items. However these are only closer matches across the two principal component directions, it is no realistic to expect the matches across the 768 dimensional sentence encodings to be perfectly portrayed over two dimensions. I did attempt to visualise the components over three dimensions, however these plots were very difficult to interpret with a lot of the plotted points obstructed other ones. The two dimensional PCA visualisations are still useful tools for understanding my results as long as their limitations are acknowledged and taken into consideration.

Figure 5.2a shows a plot of all the test risk inputs that are more semantically similar to "inadequate first aid" than any other item in the knowledge base. The figure shows how the vectors for semantically similar sentences cluster around the knowledge base item they are similar to, it is also noteworthy that test inputs with lower similarity scores (those that are below the threshold) are further away from the knowledge base match than those which have higher similarity scores (are above the threshold). Figure 5.2b shows test input sentences and their two closest matches, it again shows how similar pieces of text cluster around one another, and again it is noteworthy that closer matches are plotted closer to each other in the PCA visualisation, the plotted sentences



(a) Plot of PCA for test risks that are most semantically similar to "inadequate first aid"

(b) Plot of PCA for sentence that closely matches two items in knowledge base

Figure 5.2: Principal Component Analysis Visualisations

and the similarity scores can be found in appendix figure .21.

5.4 Performance

Table 5.1 contains the average time in seconds it takes for for activity, risk, and mitigation look ups to take place and vagueness/ambiguity results be returned on my laptop. The table has a column for when the knowledge base needs to be fetched and a column for when the fetch is stored in local storage (cached). A full break down of my results, the strings used to collect them, and the individual times can be found in appendix .14. The tables show that performing look ups on a cached knowledge base is drastically quicker than having to fetch the entire knowledge base at the start of each process. The tables also show that for longer strings and inputs with labels that have more knowledge base items, the process of getting its vagueness/ambiguity takes more time, this is likely because longer sentences take more time to encode, and if there are more items for it to be compared in the knowledge base, more calculations need to happen.

In section 1.3 I set the objective: "The score for vagueness and related sentences should be returned as quickly as possible", with the criteria stating that if results took longer than 5 seconds to return I would consider it a failure. Table 5.1 shows I have met and surpassed my criteria for this objective, with activities taking as little as 0.6275 seconds to be processed when the knowledge based is cached. I am meeting my objective even when I perform a lookup on the knowledge base at the start of the process, with the slowest average time to return a result for processing an input mitigation when the knowledge base is not cached, taking an average of 2.29875 seconds. Not a single one of my testing look ups took over 5 seconds (appendix .14). I have assumed any deployment of my implementation would have access to at least the equivalent computing resources to my laptop.

Table 5.1: Average time taken in seconds to get vagueness and ambiguity score for each knowledge base label type

Label	Avg. Time Taken (not cached)	Avg. Time Taken (cached)
Activity	2.03625	0.6275
Risk	2.156875	0.670625
Mitigation	2.29875	0.709375

5.5 Identifying Problems

An anecdotal look through the semantic matches of my testing data shows many of my the strings in my knowledge base contain multiple distinct ideas, for example the input sentence *"Major injury, Fire, Environmental damage Poor Training"* matched with knowledge base item *"injury to operators, other trades and bystanders fire / explosion, injury from debris, noise"* with a score of 0.7643582821 (above the threshold for vagueness). This example shows that despite being constantly maintained by humans, and regularly groomed for issues, my knowledge base may have too much variation of vagueness and ambiguity in itself to be able to provide meaningful information about the vagueness and ambiguity of another piece of text.

I also encounter a problem where a knowledge base match for an input contains a small amount of additional information that despite having a small impact on the semantic meaning of a sentence, has a larger impact on its context in a construction domain. For example input sentence *"access and egress into excavation"* matches with knowledge base sentence *"access in to the excavation, slips trips and falls"* with a score of 0.7200915813. The issue with the example is that the input has not actually specified what the risk is whereas the knowledge base match identifies *"slips, trips, and falls"*, the input sentence should have been flagged as vague as the additional information is important to properly identify the risk. The example raises a bigger issue with using semantic similarity to measure vagueness, small details can mean a lot in a construction risk assessment, however they may make a small impact on the overall semantic meaning of the sentence.

Breaking down the language used in a sentence gives an insight into language features that cause problems, I have anecdotally identified that a lot of my problematic sentences are compound sentences that contain multiple clauses. Compound sentences will impact the sentence encoding as the 768 dimensional vector has to represent two distinct ideas at once. I further investigated this observation using a simple clause detection algorithm and spaCy [54], finding that only 8 of my 493 knowledge base activities and 45 of the 1208 knowledge base risks contained more than one clause, both relatively low proportions. However, I found that 1066 of my 3893 (27.4%) knowledge base mitigations contained more than one clause, with one string containing 7 clauses. The mitigation results suggest compound sentences may be a prevalent issue within my knowledge base. Activities and risks are also more likely to be paraphrased and not have a proper sentence structure; I suspect a

more advanced clause detection algorithm would find that more of my activity and risk texts contain multiple clauses than the simple detection algorithm suggests. Given the opportunity to tackle the problem proposed in this project again I would like to have broken down my knowledge base sentences into individual clauses during my data preprocessing.

5.6 Further Work

Whilst working on my project and analysing my results I identified multiple pieces of further work I would be interested in seeing the results of given the opportunity. I feel there is an opportunity to utilise graph databases by using connections between nodes to identify the vagueness/ambiguity of an entire row of a risk assessment table as well as the individual pieces of text, and using connection may also provide more information about the vagueness/ambiguity of a piece of text itself. I would also like to take the time to complete a survey to assess the success of my algorithm, [14] provides a framework for writing survey questions that assess a models ability to detect vagueness.

In March 2022, Ghosal et al. published an article on novelty detection in NLP [55], unfortunately this was too late for me to use in my project, however I would be interested to see what insight novelty detection could provide into the vagueness/ambiguity of a piece of text against a knowledge base. I am interested to see whether vague text would be considered novel because its content cannot be represented by the knowledge base, or considered not novel as the vagueness means it contains no new information.

Given the time and resources I would also like to further clean and process my knowledge base data, as mentioned previously this could include breaking down text into its individual clauses. I would also like to label my testing data with information about each items vagueness/ambiguity, this would allow me to more easily assess the success of using semantic similarity to detect vagueness/ambiguity as well as explore fine-tuning a pretrained SBERT model for the task and a construction domain. BERT and SBERT are so powerful because they do not require a huge amount of training data to be fine tuned for a specific task, I am disappointed that I have not explored fine-tuning a model and take advantage of its power for my project.

6 Conclusion

Throughout this project I have discussed multiple potential methods of measuring the vagueness and ambiguity of a piece of text, eventually deciding to experiment with comparing the semantic similarity of sentence encodings calculated with a pretrained all-mpnet-base-v2 SBERT model.

My final implementation met the criteria required to fulfill my first objective, this was achieved in the GraphHelper class I wrote (section 4.3.3). The class contains functions to ingest new strings with any label, storing them with a "Staging" label until they can be reviewed by a human. There are also functions to change the label of any ingested text from staging to it's production equivalent meaning after a new piece of text is reviewed by a human it can be included in similarity look ups. My ingestion functions also prevent duplicates by checking for identical matches in the production data, and using the cypher "MERGE" function to ingest staging data. Furthermore, I met and surpassed the criteria required to fulfill my fourth objective, with even my slowest test runs returning results well within my 5 second time limit (section 5.1).

The criteria for objectives 2 and 3 relied on my personal opinion and anecdotal evidence, throughout my results (chapter 5) I attempt to find evidence to influence my opinion on whether either objective was met. Considering all of my results, I feel that semantic similarity using SBERT sentence encodings are a feasible way of detecting vagueness, however I have not seen evidence to suggest it is a useful way of measuring ambiguity. I discuss at length issues within my data that I feel hinder my algorithms ability to detect both vagueness and ambiguity, and would recommend that any data is at least tidied to fix the problems identified in section 5.5 before any implementation of my project is deployed in a commercial setting.

To conclude by relating back to my research question, I misjudged how clean a data set had to be to interpret information about the vagueness and ambiguity of an input. Despite this my knowledge base was still able to provide some information about the vagueness of a sentence. I have made suggestions on methods for cleaning the data that may allow further work to determine whether the ambiguity can be identified using text similarity, or whether the requirements on the knowledge base are impossibly high.

Bibliography

[1] P. N. Ben Lorica. 'Gradientflow: 2021 nlp survey report.' (), [Online]. Available: https://gradientflow.com/2021nlpsurvey/. (accessed: 07.04.2022).

[2] L. Page, S. Brin, R. Motwani and T. Winograd, 'The pagerank citation ranking : Bringing order to the web,' in *WWW 1999*, 1999.

[3] A. Mandal, R. Chaki, S. Saha, K. Ghosh, A. Pal and S. Ghosh, 'Measuring similarity among legal court case documents,' Nov. 2017, pp. 1–9. DOI: 10.1145/3140107.3140119.

[4] M. T. Sultan A Meo, 'Turnitin: Is it a text matching or plagiarism detection tool?,' 2019.

[5] Intuety. 'Intuety website.' (), [Online]. Available: https://www.intuety.io/. (accessed 22.04.2022).

[6] J. P. Tingström. 'Activate conference closing talk.' (), [Online]. Available: https://findwise.com/blog/activate-conference-2018/. (accessed 08.04.2022).

[7] Health and S. Executive. 'Managing risks and risk assessment at work.' (), [Online]. Available: https://www.hse.gov.uk/simple-health-safety/risk/index.htm. (accessed 25.04.2022).

[8] 'The management of health and safety at work regulations 1999.' (), [Online]. Available: https://www.legislation.gov.uk/uksi/1999/3242/contents/made. (accessed 25.04.2022).

[9] L. Wittgenstein. (1953), [Online]. Available: http://fs2.american.edu/dfagel/www/Class%20Readings/Wittgenstein/Philosophical%20Investigations%20(1st%20100).html. (accessed 22.04.2022).

[10] S. Leitch and S. Davenport, 'Strategic ambiguity in communicating public sector change,' *Journal of Communication Management*, vol. 7, pp. 129–139, Apr. 2003. DOI: 10.1108/13632540310807340.

[11] J. Channell, *Vague language*. Oxford University Press, 1994, ISBN: 0194371867.

[12] P. Guélorget, B. Icard, G. Gadek *et al.*, *Combining vagueness detection with deep learning to identify fake news*, Oct. 2021.

[13] L. Lebanoff and F. Liu, 'Automatic detection of vague words and sentences in privacy policies,' in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018. DOI: 10.18653/v1/D18-1387. [Online]. Available: https://aclanthology.org/D18-1387.

[14] T. Anthonio, I. Bhat and M. Roth, 'Wikihowtoimprove: A resource and analyses on edits in instructional texts,' May 2020.

[15] B. D. Cruz, B. Jayaraman, A. Dwarakanath and C. McMillan, 'Detecting vague words & phrases in requirements documents in a multilingual environment,' in *25th International Requirements Engineering Conference (RE)*, IEEE, 2017, pp. 233–242.

[16] 'Sentence length: Why 25 words is our limit.' (), [Online]. Available: https://insidegovuk.blog.gov.uk/2014/08/04/sentence-length-why-25-words-is-our-limit/. (accessed 25.05.2022).

[17] 'Plain writing initiative.' (), [Online]. Available: https://www.sec.gov/plainwriting.shtml. (accessed 25.05.2022).

[18] readabilityformulas.com. 'The gunning's fog index (or fog) readability formula.' (), [Online]. Available: https://www.readabilityformulas.com/gunning-fog-readability-formula.php. (accessed on 24.05.2022).

[19] P. H. McLoughlin, 'Smog grading —– a new readability formula,' 1969.

[20] P. Fitzsimmons, B. Michael, J. Hulley and G. Scott, 'A readability assessment of online parkinson's disease information,' *The journal of the Royal College of Physicians of Edinburgh*, vol. 40, pp. 292–6, Dec. 2010. DOI: 10.4997/JRCPE.2010.401.

[21] FLESCH, 'A new readability yardstick.,' 1948.

[22] M. Porter. 'An algorithm for suffix stripping.' (), [Online]. Available: https://tartarus.org/martin/PorterStemmer/def.txt. (acessed 27.04.2022).

[23] J. Kanerva, F. Ginter and T. Salakoski, 'Universal lemmatizer: A sequence to sequence model for lemmatizing universal dependencies treebanks,' *CoRR*, vol. abs/1902.00972, 2019. arXiv: 1902.00972. [Online]. Available: http://arxiv.org/abs/1902.00972.

[24] N. Kang, B. Singh, Z. Afzal, E. M. van Mulligen and J. Kors, 'Using rule-based natural language processing to improve disease normalization in biomedical text,' *Journal of the American Medical Informatics Association : JAMIA*, vol. 20, Oct. 2012. DOI: 10.1136/amiajnl-2012-001173.

[25] A. Hardjojo, A. Gunachandran, L. Pang *et al.*, 'Validation of a natural language processing algorithm for detecting infectious disease symptoms in primary care electronic medical records in singapore,' *JMIR Medical Informatics*, vol. 6, e36, Jun. 2018. DOI: 10.2196/medinform.8204.

[26] L. Yao, M. Chengsheng and Y. Luo, 'Clinical text classification with rule-based features and knowledge-guided convolutional neural networks,' *The Sixth IEEE International Conference on Healthcare Informatics*, Apr. 2019. DOI: 10.1186/s12911-019-0781-4.

[27] T. Mikolov, K. Chen, G. Corrado and J. Dean, *Efficient estimation of word representations in vector space*, 2013. DOI: 10.48550/ARXIV.1301.3781. [Online]. Available: https://arxiv.org/abs/1301.3781.

[28] R. Miles. (), [Online]. Available: https://www.youtube.com/watch?v=gQddtTdmG_8. (last accessed 03.05.22).

[29] A. Vaswani, N. Shazeer, N. Parmar *et al.*, 'Attention is all you need,' Jun. 2017.

[30] peltarion.com. 'Bert encoder.' (), [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/bert-encoder. (accessed on 24.05.2022).

[31] J. Elman, 'Finding structure in time,' in Feb. 2020, pp. 289–312, ISBN: 9781315784779. DOI: 10.4324/9781315784779-11.

[32] S. Wang and J. Jiang, 'Learning natural language inference with LSTM,' *CoRR*, vol. abs/1512.08849, 2015. arXiv: 1512.08849. [Online]. Available: http://arxiv.org/abs/1512.08849.

[33] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao and B. Xu, 'Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling,' *CoRR*, vol. abs/1611.06639, 2016. arXiv: 1611.06639. [Online]. Available: http://arxiv.org/abs/1611.06639.

[34] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding,' Oct. 2018.

[35] H. Xu, B. Liu, L. Shu and P. S. Yu, *Bert post-training for review reading comprehension and aspect-based sentiment analysis*, 2019. DOI: 10.48550/ARXIV.1904.02232. [Online]. Available: https://arxiv.org/abs/1904.02232.

[36] Y. Liu, M. Ott, N. Goyal *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. DOI: 10.48550/ARXIV.1907.11692. [Online]. Available: https://arxiv.org/abs/1907.11692.

[37] Y. Zhang and Z. Xu, 'Bert for question answering on squad 2.0,'

[38] N. Reimers and I. Gurevych, 'Sentence-bert: Sentence embeddings using siamese bert-networks,' Aug. 2019.

[39] A. Conneau, D. Kiela, H. Schwenk, L. Barrault and A. Bordes, *Supervised learning of universal sentence representations from natural language inference data*. [Online]. Available: https://research.facebook.com/downloads/infersent/.

[40] D. Cer, Y. Yang, S.-y. Kong *et al.*, *Universal sentence encoder*, 2018. DOI: 10.48550/ARXIV.1803.11175. [Online]. Available: https://arxiv.org/abs/1803.11175.

[41] (), [Online]. Available: https://neo4j.com/cloud/platform/aura-graph-database/. (last accessed 04.05.22).

[42] L. Tyermean. 'The life and times of the rev. samuel wesley: Rector of epworth.' (), [Online]. Available: https://www.gutenberg.org/ebooks/67980. (last accessed 04.05.22).

[43] (), [Online]. Available: https://sbert.net/docs/pretrained_models.html. (accessed on 08.05.2022).

[44] Microsoft. 'All-mpnet-base-v2.' (), [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2. (accessed 08.04.2022).

[45] K. Song, X. Tan, T. Qin, J. Lu and T.-Y. Liu, 'Mpnet: Masked and permuted pre-training for language understanding,' *arXiv preprint arXiv:2004.09297*, 2020.

[46] sbert.net. 'Sbert pooling types.' (), [Online]. Available: https://www.sbert.net/docs/package_reference/models.html#sentence_transformers.models.Pooling. (accessed on 10.05.2022).

[47] ——, 'Multi-qa-mpnet-base-dot-v1.' (), [Online]. Available: https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1. (accessed on 09.05.2022).

[48] ——, 'Multiple negatives ranking loss algorithm.' (), [Online]. Available: https://www.sbert.net/docs/package_reference/losses.html#multiplenegativesrankingloss. (accessed on 10.05.2022).

[49] neo4j. 'Cypher querly language - neo4j.' (), [Online]. Available: https://neo4j.com/developer/cypher/. (accessed on 10.05.2022).

[50] K. Papineni, S. Roukos, T. Ward and W. J. Zhu, 'Bleu: A method for automatic evaluation of machine translation,' Oct. 2002. DOI: 10.3115/1073083.1073135.

[51] N. Craswell, B. Mitra, E. Yilmaz, D. Campos and J. Lin, 'Ms marco: Benchmarking ranking models in the large-data regime,' Jul. 2021, pp. 1566–1576. DOI: 10.1145/3404835.3462804.

[52] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy and S. Bowman, 'Glue: A multi-task benchmark and analysis platform for natural language understanding,' Apr. 2018.

[53] A. Wang, Y. Pruksachatkun, N. Nangia *et al.*, *Superglue: A stickier benchmark for general-purpose language understanding systems*, May 2019.

[54] M. Honnibal and I. Montani, 'spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,' To appear, 2017.

[55] T. Ghosal, T. Saikh, T. Biswas, A. Ekbal and P. Bhattacharyya, 'Novelty Detection: A Perspective from Natural Language Processing,' *Computational Linguistics*, vol. 48, no. 1, pp. 77–117, Apr. 2022, ISSN: 0891-2017. DOI: 10.1162/coli_a_00429. eprint: https://direct.mit.edu/coli/article-pdf/48/1/77/2006641/coli_a_00429.pdf. [Online]. Available: https://doi.org/10.1162/coli%5C_a%5C_00429.

Appendix

Table .1: Example of a vague table row in a risk assessment

Activity	Risk	Mitigation
Sheet piling	Site works	Use proper safety signs to highlight activity

Table .2: Example of an ambiguous table row in a risk assessment

Activity	Risk	Mitigation
Scaffolding	Falling objects	Tie objects up

Figure .1: Flesch Reading Easy Formula

$$206.835 - 1.015(\frac{\text{total words}}{\text{total sentences}}) - 84.6(\frac{\text{total syllables}}{\text{total words}})$$

$$\text{cosine similarity} = cos(\theta) = \frac{A.B}{||A||||B||}$$

‘

Figure .2: Cosine Similarity formula for two vectors A and B

Figure .3: Logic used by the porter stemming algorithm to represent words as consonant vowel groups

```
- A consonant is a letter other than A E I O or
U and other than Y preceded by a consonant
- A consonant is denoted by v, and a vowel is
denoted by v
- A string ccc... of length greater than 0 (a
list of one or more consecutive consonants) is
denoted by C, a string vvv... of length greater then
0 is denoted by V, therefore any word can have
one of four forms:
    - CVCV ... C
    - CVCV ... V
    - VCVC ... C
    - VCVC ... V
- This can be represented in the single form
[C]VCVC...[V] where the square brackets denote
the arbitrary presence of their contents (a
word doesn't have to start with a constant or
end with a vowel)
- This can be rewritten as [C]($VC^m$)[V]
```

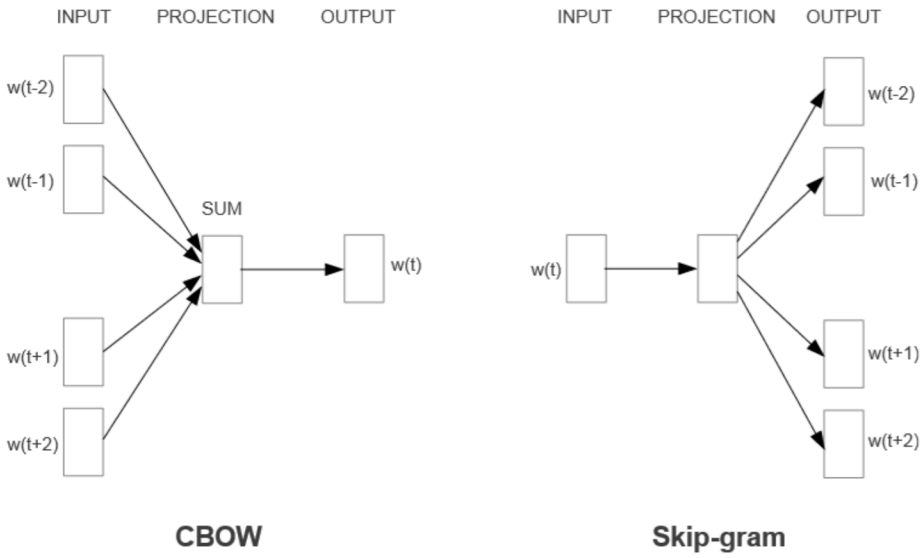


Figure .4: Architecture of CBOW and Continuus Skip-Gram word embedding models [27]

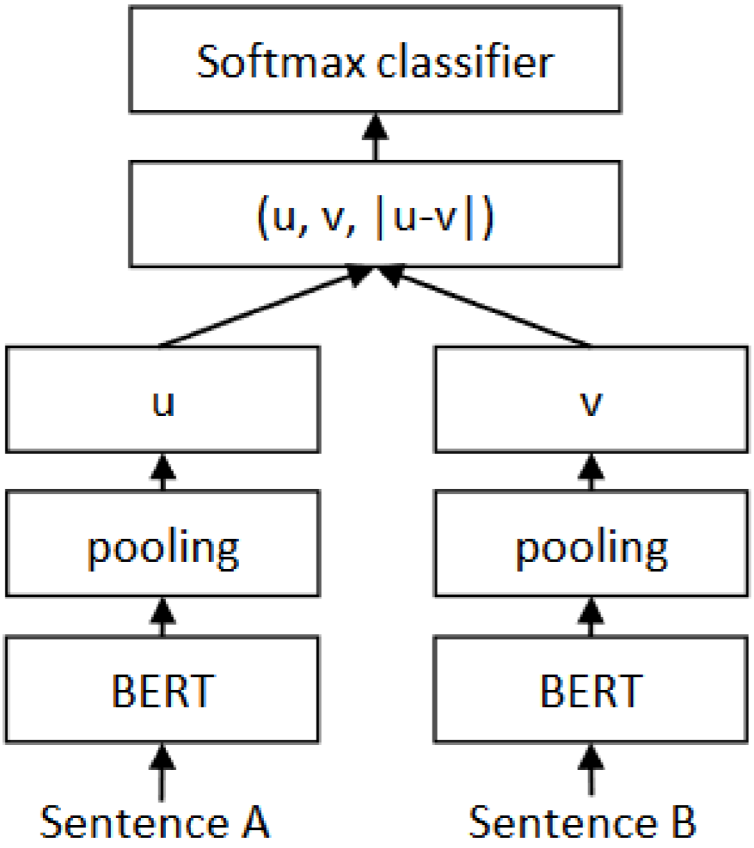


Figure .5: Siamese network architecture used to train SBERT [38]

Figure .6: Screenshot of my data using Neo4j Bloom

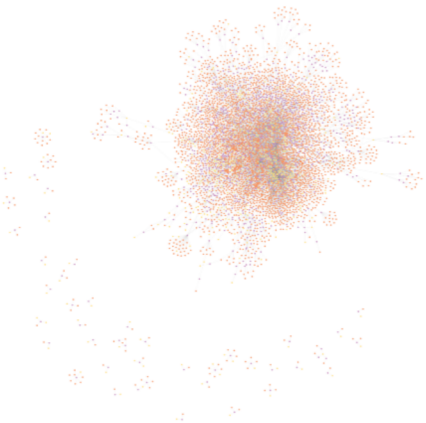


Figure .7: Example of connected nodes in my graph database

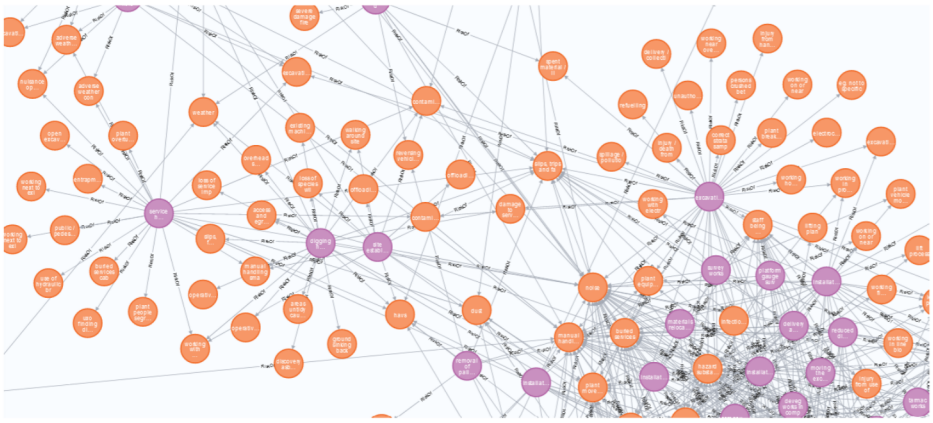


Figure .8: Example of the contents for a node in my graph database

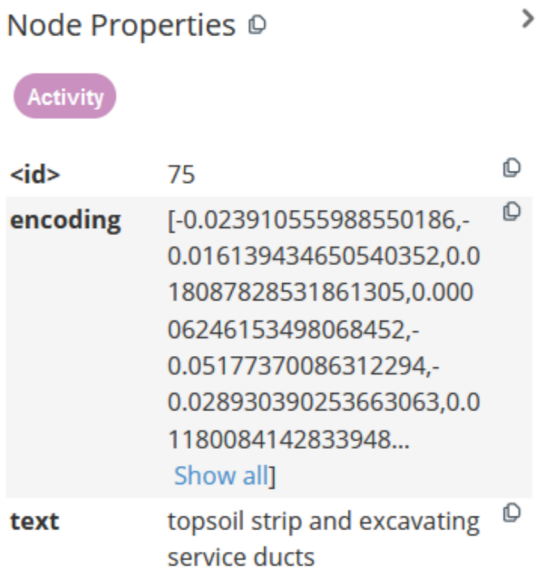


Figure .9: Number of words in the mitigation strings of my knowledge base

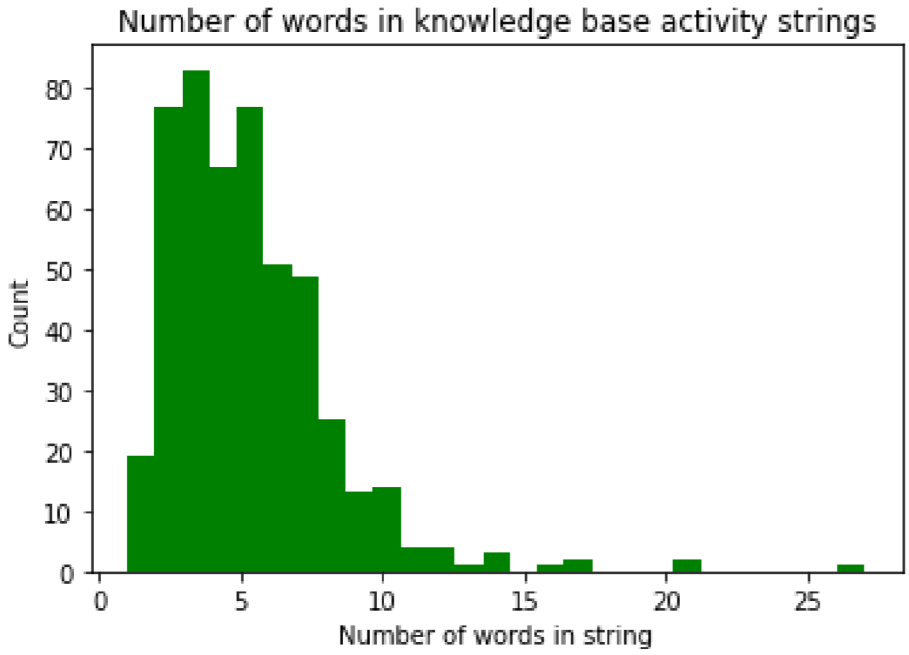


Figure .10: Number of words in the risk strings of my knowledge base

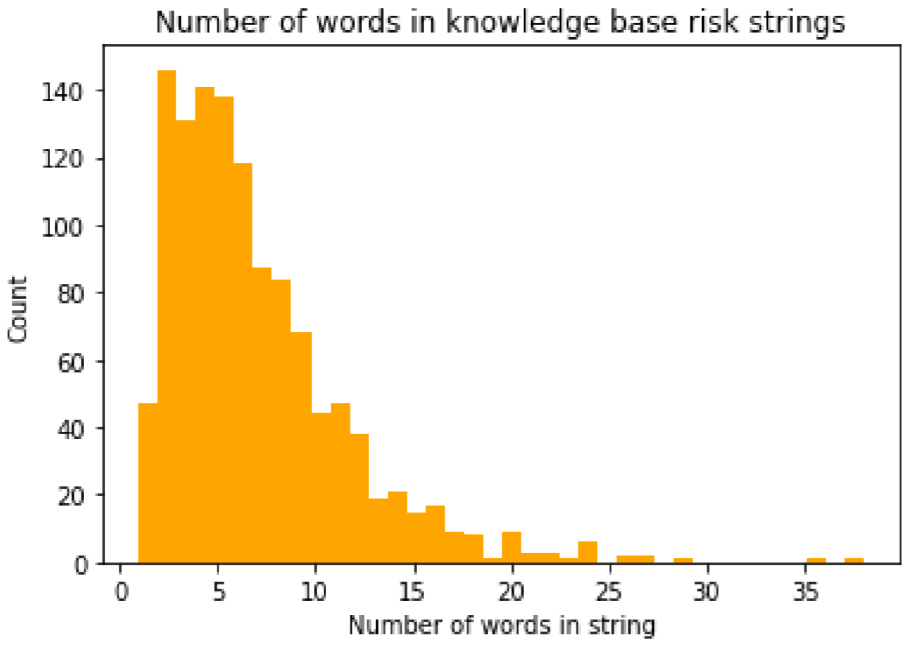


Figure .11: Number of words in the activity strings of my knowledge base

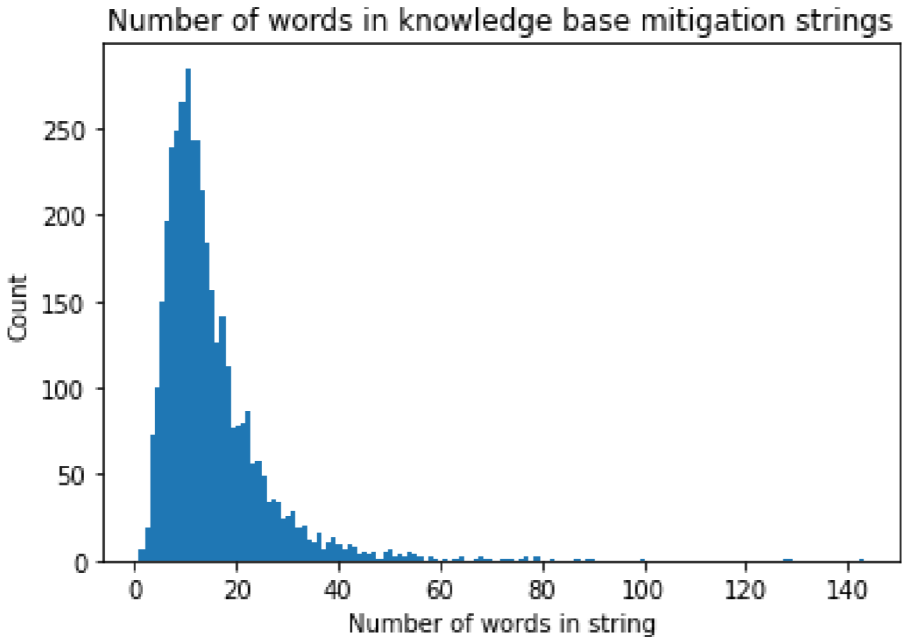


Figure .12: Example of Flesch Reading Ease Score for two different sentences

```
Sentence 1: The cat in the hat ate fish
Flesch Reading Ease Score: 100

Sentence 2: The feline wearing the hat consumed fish
Flesch Reading Ease Score: 81.3
```

Figure .13: Demonstration of Lemmatisation and Stemming of construction specific words

```
>>> import nltk
>>> from nltk.stem import WordNetLemmatizer, PorterStemmer
>>> nltk.download('wordnet')
[nltk_data] Downloading package wordnet to /home/adambarr/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
>>> lemmatizer = WordNetLemmatizer()
>>> stemmer = PorterStemmer()
>>> lemmatizer.lemmatize("piling")
'piling'
>>> stemmer.stem("piling")
'pile'
```

		Time Taken (not cached)						Time Taken (cached)					
Label	**String**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Avg.**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Avg.**
	relocation of materials	1	1.984375	2.046875	1.96875	1.9375	1.7875	0.625	0.46875	0.640625	0.546875	0.625	0.58125
	installation and commissioning of new flowmeter at noble ln tps (including ducting to kiosk)	2.21875	2.140625	2.046875	2.15625	2.375	2.1875	0.625	0.84375	0.59375	0.875	0.625	0.7125
	Excavation of Cofferdam	2.203125	2.078125	2.03125	2.1875	2.03125	2.10625	0.671875	0.609375	0.625	0.734375	0.65625	0.659375
	Driving around Sites	1.875	1.96875	2.1875	1.9375	2.046875	2.003125	0.59375	0.546875	0.640625	0.46875	0.5	0.55
Activity	installation of sheet piles to pumping station	2.171875	2.078125	2.109375	1.96875	2.15625	2.096875	0.5	0.734375	0.734375	0.578125	0.625	0.634375
	Dermatitis, cuts and bruises COSHH Damage to skin, water contact. Mechanical lifting	2.328125	2.0625	2.078125	2.15625	2.21875	2.16875	0.625	0.609375	0.5	0.71875	0.40625	0.571875
	Operatives electrocuted whilst using electrical tools and excavating Electrocution	1.859375	2.21875	2.046875	2.140625	2.15625	2.084375	0.875	0.625	0.625	0.75	0.71875	0.71875
	Sewage Personal hygiene, risk of disease	1.96875	2.296875	2.0625	1.90625	2.140625	2.075	0.6875	0.6875	0.5	0.625	0.5625	0.6125
	Joints, valves or fitting failure. Hose failure. Pipeline rupture / localised flooding.injury by high pressure water discharge Hydrostatic Testing	2.078125	2.171875	2.34375	2.140625	2	2.146875	0.84375	0.625	0.75	0.75	0.71875	0.7375
Risk	multiple injuries crushing musculoskeletal disorders death damage to vehicles and structures	2.296875	2.34375	2.3125	2.1875	2.40625	2.309375	0.75	0.75	0.75	0.625	0.6875	0.7125
	Operatives to wear correct rate dust mask and be face fit tested before use	2.25	2.25	2.109375	1.984375	2.375	2.19375	0.625	0.734375	0.625	0.640625	0.625	0.65
	All operative to be inducted before works and sign to relevant RAMS Provide safe access into works area Fence off works area & provide warning signs Keep works tidy & clear of materials Be aware of any works being undertaken by others Protect against unauthorized entry Good personal hygiene & washing facilities	2.46875	2.375	2.46875	2.421875	2.296875	2.40625	0.859375	0.84375	0.859375	0.703125	0.84375	0.821875
	Silt sock to installed pre-discharge	2.515625	2.171875	2.171875	2.265625	2.1875	2.2625	0.625	0.625	0.625	0.75	0.75	0.675
	Work should be planned to reduce the incidence of trip hazards where cables or hoses are expected in the work area	2.34375	2.421875	2.34375	2.171875	2.140625	2.284375	0.75	0.625	0.859375	0.71875	0.625	0.715625
Mitigation	Deliveries are to be booked into site via the Site Agent and onto the traffic management plan A banksman MUST be present when deliveries enter the site	2.4375	2.296875	2.34375	2.546875	2.109375	2.346875	0.75	0.609375	0.625	0.828125	0.609375	0.684375

Figure .14: All performance results collected (measured in seconds)

Input Text	Knowledge Base Match	Score
Pollution'	pollution'	0.9999999403953552'
weather and winter working'	weather and winter working'	0.9999999403953552'
local landscape, ecological features and inhabitants concrete and cement products	local landscape, ecological features and inhabitants concrete and cement products	0.9999997615814209'
local landscape, ecological features and inhabitants concrete and cement products'	eye injuries'	0.9999995231628418'
excavations.'	excavations'	0.9789394736289978'
unplanned cutting down and damage to trees and hedgerows working around trees and hedgerows'	unplanned cutting down and damage to trees and hedge rows working around trees and hedge rows'	0.9761753678321838'
underground/overhead services'	overhead & underground services'	0.9544980525970459'
Corona Virus'	coronavirus covid'	0.8994283676147461'
loading platform and bridge with plant and materials.'	loading platform with plant and materials'	0.8890511989593506'
Manual handling — injury to staff'	injury from manual handling'	0.879616916179657'
Stability of bridge and embankment'	embankment stability'	0.8399794697761536'
syringes / sharps'	encountering syringes'	0.8197046518325806'
Personal injury Pulled muscles/ back problems'	pulled muscles / back problems or personal injury'	0.8086856007575989'
Use Of Plant'	working with plant'	0.759066104888916'
Use of Plant'	working with plant'	0.759066104888916'
Underground services'	striking underground services'	0.755462646484375'
Leptospirosis - Reportable disease from contact with rats and other vermin within the work area'	leptospirosis'	0.7394660115242004'
Site Accidents incidents'	accidents near site entrance off'	0.7342627644538879'
Use Of saws'	use of cutting equipment'	0.7226788997650146'
Operational Railway'	working within railway corridor'	0.7125051021575928'
Excavations and TTRO street works'	excavations'	0.6911819577217102'
Crane access and egress'	working with floor crane'	0.6673873066902161'
Cutting'	use of cutting equipment'	0.6563196182250977'
Cutting and burning'	use of cutting equipment'	0.6488978266716003'
Lifting Loler'	lifting'	0.6378306150436401'
Welding-Burning'	hot works electrofusion welding'	0.6288377642631531'
not working to government guidelines spread of the virus'	coronavirus / covid19'	0.5895196199417114'
Working on a Live Operational Wastewater Treatment Works Weil's Disease Traffic Accident Collision Crushing'	working with or near mobile plant crush injuries / collision'	0.5879546999931335'
Pulling PE pipe'	lifting of pipework'	0.5829757452011108'
Repeat works / additional costs incurred Incorrect concrete specification'	additional; costs and rework setting out by competent engineers'	0.57550448179245'
LCE/RA/040 Use of machine mounted hydraulic breakers'	use of hydraulic breaker / air lance struck by flying debris contact with pressurised air'	0.5218774676322937'
manual handling of bitumen blocks'	impact dropping bitumen on foot'	0.4972438812255859 4'
not to correct spec or alignment completing the task right first time'	not to specification incorrect fall'	0.4850475788116455'
operating dumper'	loading & unloading of vegetation & material to dumpers & stockpile'	0.4585532248020172'
Works on operational platform'	failure of temporary works platform not built to temporary works design'	0.43938133120536804'
7'	age'	0.43839308619499207'
6'	age'	0.40805166959762573'
reworking freezing temperatures'	weather change'	0.3802826404571533'
assessment by tap'	surveying live services and pipes'	0.3751987814903259'
name'	attitude'	0.335456520318985'
Name'	attitude'	0.335456520318985'
sika'	havs stihl saw'	0.3173011243343353'
risk that not all the files necessary will be captured N'	incorrect storage of materials'	0.28928232192993164'
hi-viz waist coat or jacket to'	wintery conditions'	0.2673531472682953'
De Veg-Install Terrafirma'	contamination to the land / water silt protection'	0.23653578758239746'
Install Vortok'	entrapment in vacex'	0.2295534908771515'
no.'	trespass	0.20106105506420135'

Figure .15: Snapshot of results used to influence threshold decision

Figure .16: Formula for calculating Levenshtein distance between two strings a and b with lengths |a| and |b| respectively

$$\text{lev}(a,b) = \begin{cases} |a|, & if|b| = 0 \\ |b|, & if|a| = 0 \\ lev(tail(a), tail(b)) & ifa[0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & otherwise \end{cases}$$

‘

```
MATCH (r:Risk)
RETURN r.encoding as encoding, r.text as text
```

Figure .17: Example of cypher query that fetches all risks in my knowledge base, returning their text and encodings in easily accessible fields

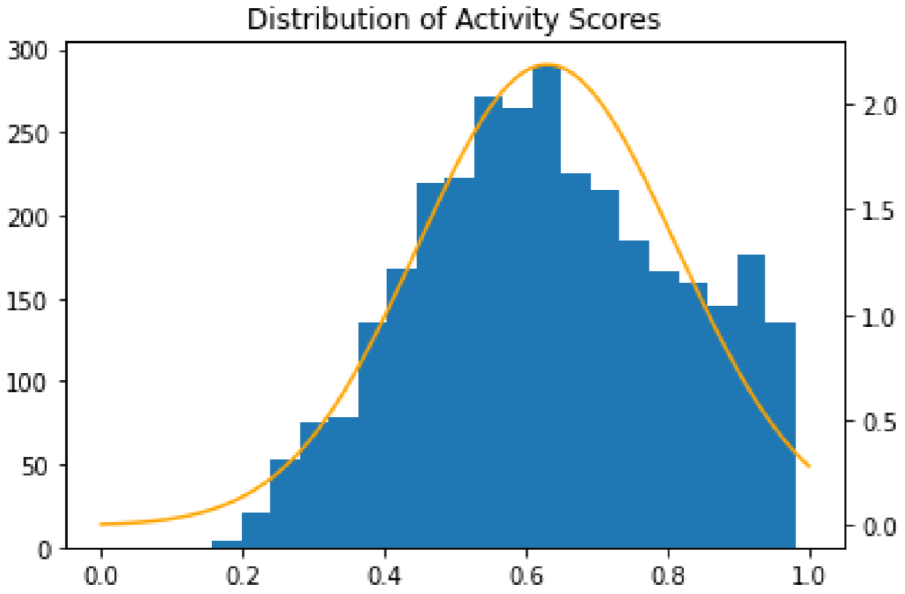


Figure .18: Distribution for results for test inputs with activity label

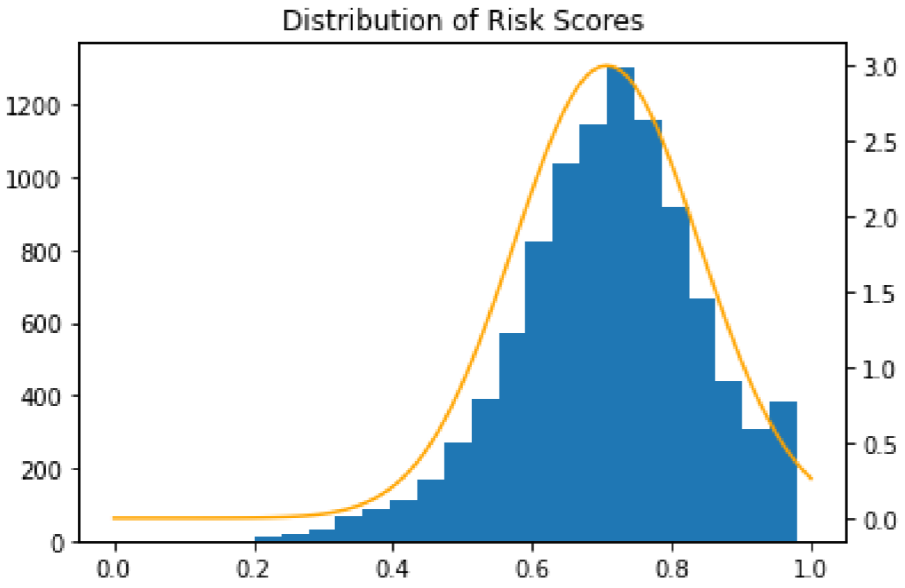


Figure .19: Distribution for results for test inputs with risk label

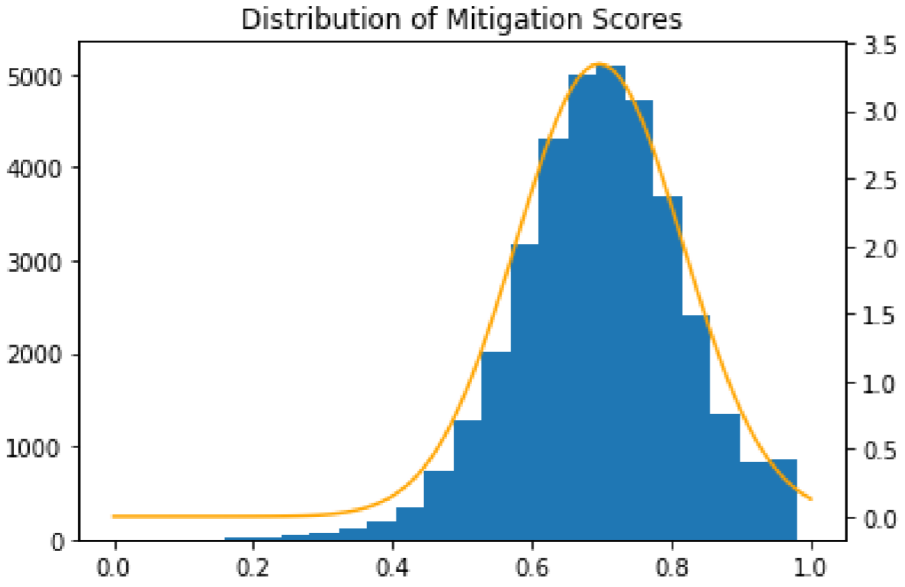


Figure .20: Distribution for results for test inputs with mitigation label

		Best Match		Second Best Match	
ID	Text	Text	Score	Text	Score
Text 1	New shower to be fitted in Mess Room	shower to be fitted in mess room	0.9622101784	sufficient wc's and sinks with hot and cold water, hand wash system (debcare or similar) and hand towels	0.4231586158
Text 2	Correct PPE to include non- perforated gloves, foot-wear and goggles to be worn	nonperforated gloves, foot-wear and goggles to be worn	0.823544383	correct ppe to be worn including gloves	0.7766017318
Text 3	Ensure there adequate fencing securely fastened when at all times	where perimiter fencing is not adequete, ensure additional local fencing is in place	0.7912466526	visual risk assessment to be carried out before carrying fencing panels or feet good housekeeping standards to be maintained at all times	0.7247407436
Text 4	test atmosphere within confined space for 10 minutes before entry using calibrated gas monitor (a minimum of 3 staged checks to be carried out as per c&g training)(only proceed with entry is safe to do so).	atmosphere testing should be continuously during entry	0.7810162306	gas monitors will be used (and fully charged before use) and the area well ventilated for 5 minutes, and tested prior to entry	0.7304663062
Text 5	twc shall complete a design brief or design justification for all dewatering operations	complete a design brief or design justification for all dewatering operations	0.8319923282	a temporary works assessment to be carried out in advance and if required twd to be implemented	0.5691119432
Text 6	area cordoned off; permit to lift to be issued and briefed; lift plan to be issued; loading supervisor or appointed person in place; all lifting equipment and accessories to be checked against safe working loads (swl); correct ppe; control weather condition; trained and competent operatives;	area cordoned off permit to lift to be issued and briefed lift plan to be issued loading supervisor or appointed person in place all lifting equipment and accessories to be checked against safe working loads swl correct ppe control weather condition trained and competent operatives	0.924505949	all lifts in accordance with lift plans no lift plan = no lift designated slinger / signaller pre use inspection on all lifting accessories minimal lifting required for activity	0.8007183075
Text 7	rotate tasks as required	rotate the task between operatives	0.6733644009	work to be rotated	0.5764371157

Figure .21: Data for PCA visualisation for sentences that closely match two items in the knowledge base

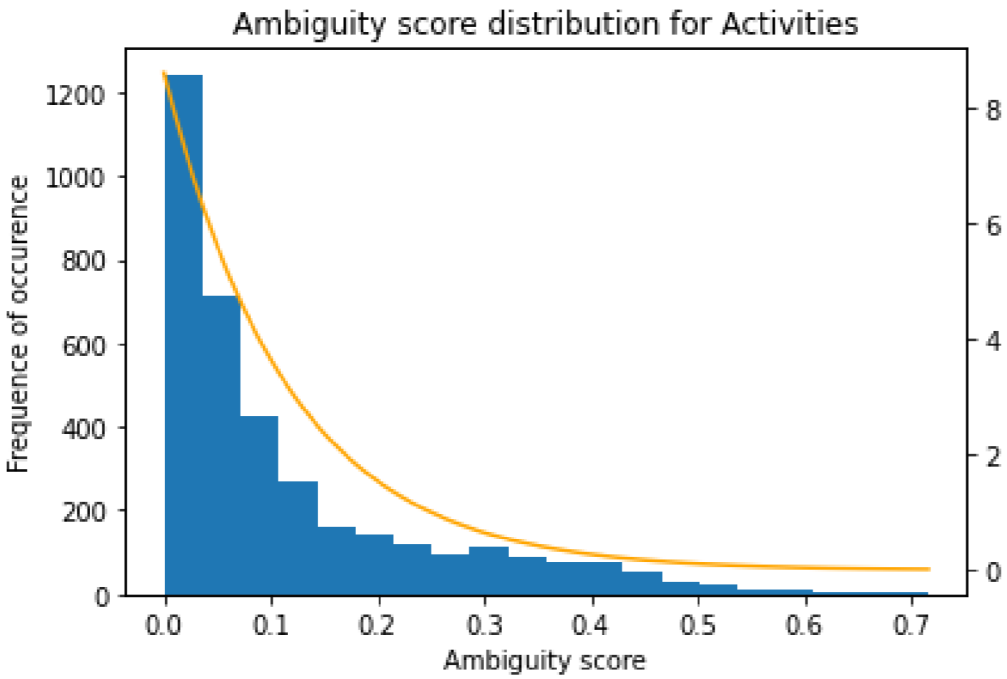


Figure .22: Distribution of ambiguity scores for label Activity

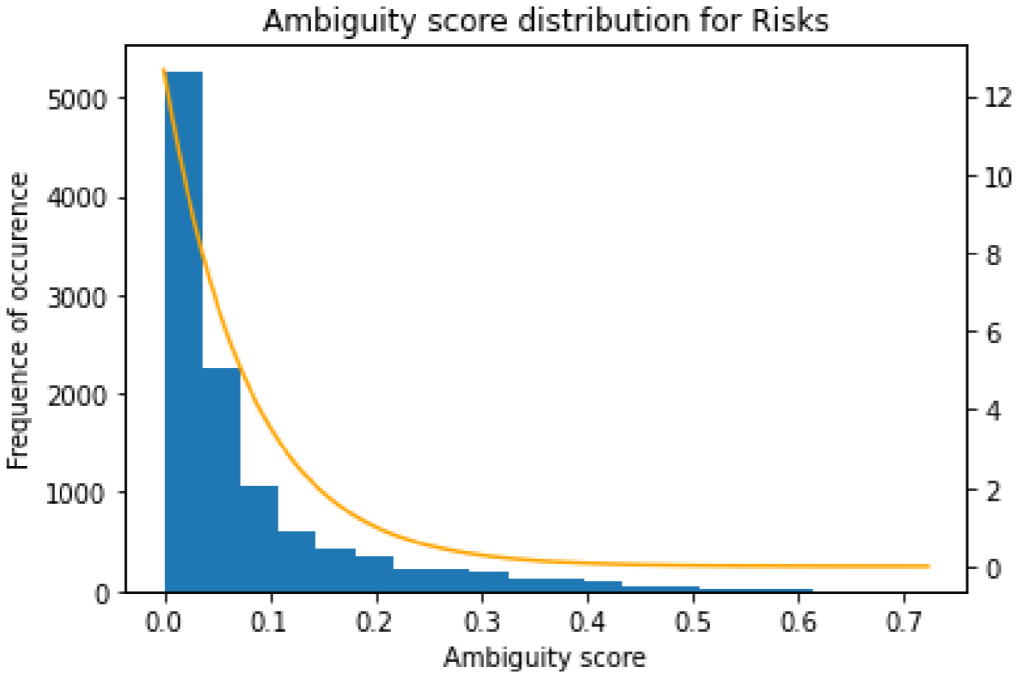


Figure .23: Distribution of ambiguity scores for label Risk

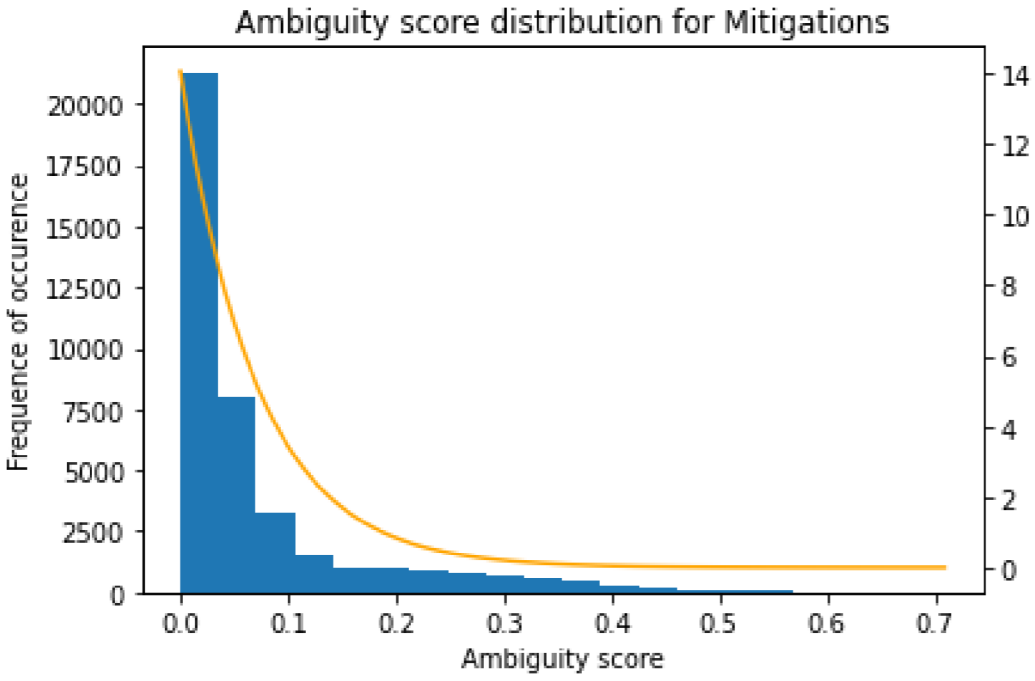


Figure .24: Distribution of ambiguity scores for label Mitigation