Size Matters: Acquiring Vague Spatial Size Information from Textual Sources

By Benjamin J Eggelton

Supervised by Dr Robert Gaizauskas

COM3610

May 1, 2019

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science by Benjamin J Eggelton.

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Benjamin Jon Eggelton

Abstract

Common sense reasoning is a large segment of Artificial Intelligence that is concerned with coding autonomous systems to have the ability to make presumptions about problems they encounter. This dissertation has the objective of creating software that will allow users to find the absolute sizes of objects through the extraction of natural language text from the internet.

The final system searches through each object's Wikipedia page and uses information extraction methods to obtain sizes of any objects identified in the text. We created a list of objects using WordNet's hierarchy and consists of 24,000 entries. The success of the system is varied; it is clear that some relationships between objects and sizes are accurate but the large amount of noise indicates that the results are not reliable enough to be used by other commonsense reasoning software.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most significant problems humans will encounter with Artificial Intelligence is creating a 'common sense'; a type of knowledge that helps us solve all kinds of problems when we are not given contextual information. While this knowledge is taken for granted by humans, it is surprisingly difficult to implement into an autonomous robot because we learn a lot of this information from day to day experiences without acknowledging it. When computer scientists find a complete way to implement common sense reasoning to a robot, we will be one monumental step closer to developing a completely autonomous, human-like artificial intelligence.

This dissertation project looks in detail at an aspect of this type of reasoning, in particular the general sizes of objects. Humans can estimate the approximate sizes of a vast range of objects having not seen them in recent memory. This dissertation has the aim of using natural language processing to extract information from web pages in order to create a database spanning all common 'objects' (inclusive of animals; vegetation; buildings and other inanimate physical entities) with their approximate measured sizes. An example use for this system would to gain contextual information in a 2d image; software could use the database to decipher what objects are near to the camera and what is far away by their relative sizes. Another potential use could be for an autonomous robot lifting or moving objects around. The robot could approximate whether the object could be lifted by comparing the object's size to its own.

The effectiveness of a solution to this information extraction problem can range on a scale, due to the number of problems the system must resolve. The ideal solution would be to have a system that could detect an object's measurement even when the object and size are not mentioned in the same sentence, but this would require our solution to resolve monumental problems introduced by co-reference resolution (explained further in chapter 2). There must also be a reliable and consistent method to label objects and sizes in a sentence, keeping the noise introduced by mislabelled words as low as possible. Problems such as these will be discussed in more detail throughout the report, with a plan of action on how they will be solved.

Chapter 2 covers an extensive literature review which will be used to create a specification for this project. In this chapter the topics that will be covered include a comparison of classifier approaches; how annotators for natural language processing are made; the

methods used to evaluate the software; how difficult the aims of the project were and the extent of the implementations (how well their problems are solved). Having analysed all of the papers, the requirements of this project will be specified with the clear extent and feasibility to which certain features can be implemented in chapter 3. Furthermore, in that section the possible evaluation methods will be listed. A technical design for the system will then be presented in chapter 4, followed by the testing of the implementation (chapter 5). The system's results will be discussed in chapter 6, with the key findings highlighted in chapter 7 alongside any other conclusions drawn from our research.

Chapter 2

Literature Survey

While work on developing common sense in systems began as early as 1959 [1] and there have been successes in areas such as taxonomic and temporal reasoning [2], it appears there has not yet been an attempt to do the exact kind of information extraction research that will be performed in this project. Ostensibly, the information extraction program to be constructed will consist of a Named Entity Recognition classifier and a Relation Extraction classifier. The approaches to each classifier will be determined by evaluating the advantages to a rule-based; supervised learning and unsupervised learning method.

2.1 Classification Methods

The text that comes into the system first uses a classifier to detect objects and sizes, then each object and size pair can be classified into a positive relationship (where they are related), or negative relationship (where they are not related). Regardless of the classifier's objectives, the approaches to classification do not change. Classification methods range from rule-based; supervised-learning; semi-supervised learning and unsupervised-learning.

2.1.1 Rule-Based Classification

This rigid classifier consists of a set of rules as a way to predict the category of an input. These rules can be as simple as an IF-THEN statement [3], meaning this method does not need training data to learn from but the patterns must be known so the rules can be created. Rule Induction Algorithms can be used to automatically find relevant rules from the data [4]. Additionally, this method can be used with Rule Ranking Measures that tell the system which rules are the most important, increasing efficiency [3].

Rule-based classifiers run quickly and have reasonable efficiency but any patterns that are not written as rules will be missed. However, the classifier can be improved; rules can be edited and updated as more defining features are observed [5]. A corpus of formal text (such as Wikipedia [6]) typically has a consistent writing style compared to informal text from the likes of social media.

Therefore, this implementation is not versatile but still useful when the text is predictable or has an obvious structure. This could be the case with an object recogniser.

By looking at a single word a human could distinguish whether it is an object or not (for example, 'car'); although some words (such as 'duck') may require the context of the sentence.

2.1.2 Supervised Classification

Supervised-learning methods apply a set of pre-written features on training data (a list of labelled sentences) in order to determine the characteristics of text in each category. An example of a feature could be the position of the text in a sentence, or whether the text has a capital letter.

The simplest supervised-learning method is the Naive Bayes classifier; although modifications can rapidly increase the complexity [7]. Another approach is to transform a document into a vector; known as a Support Vector Machine (SVM). This is considered to be more powerful than Naive Bayes but the performance is not always superior. Additionally, it's observed that SVMs have a worse resilience to missing data [8], which is highly likely in this project.

The creation of annotated data to train a supervised-learning classifier costs time, unless there is one readily available. However, very strong classifiers can be trained which are able to generalise (scale) well on large amounts of unlabelled data.

2.1.3 Semi-Supervised Classification

Semi-supervised learning, also known as 'bootstrapping', is where a large amount of raw data is given for the classifier to learn from alongside a small number of known patterns or labels. For example, if the system was given a select few labels at the start, the classifier would infer patterns from these labels in the data. Then the system would apply these patterns back over the data in order to find new labels, and use the new labels to find new patterns. This is done recursively until no more patterns or labels are found.

An advantage to this method is that a small amount of human intervention is required compared to the time taken creating training data using supervised learning. However, this approach suffers from a problem known as semantic drift [9]; where one incorrect pattern can lead to a new set of incorrect labels, which in turn lead to more incorrect patterns. A large amount of erroneous patterns would cause the system to be largely ineffective when trying to classify text, because many incorrect relationships will be found.

2.1.4 Unsupervised Classification

This type of classifier predicts the class of an input without training from pre-defined labels. The machine typically trains from a large corpus where the specific relationship pairs are known. From there, the classifier must detect the patterns on its own. One

unsupervised learning algorithm is clustering; where inputs are grouped together based on their characteristics, allowing unseen data to be distinguished into a category.

No annotated training data needs to be made but there must still be raw data for the classifier to learn patterns from. A significant advantage is that hidden patterns can be found that would not be obvious to a human. However, it could be prone to making errors and the creators cannot influence its judgements.

2.2 Source of the Data

The end goal of this project is to have a classification system that scans a large amount of text online. Before extracting the data, the source of the data must be located. The system could either use a web crawler that reads several pages from different websites, or it could scan only one website if there are enough pages to cover the sizes of a wide range of objects. There are thousands of possible sources to use over the internet but Wikipedia stands out from the rest from an informative perspective. Wikipedia is an extremely large, free online encyclopedia that's well-structured containing 5,757,574 articles [10] increasing at around 500,000 articles per year [11].

Several risks [12] come with using Wikipedia as our only source, such as inaccuracy in the information due to it being editable by the general public. Furthermore, the information can be volatile and biased; for example the houses in the USA [13] are typically 1000 square feet larger an average (in floor space) than that of a UK home [14], meaning that in the past a UK resident may have edited the web page to remove the USA values and instead add the UK values. This would change the reliability of the data significantly. However, it is also noted that generally the accuracy of articles Wikipedia is similar to that of Encyclopedia Britannica [15]. Wikipedia's category system and disambiguation pages [11] allows a program to easily access related pages to the one it is currently on. Additionally, no other website is as likely to contain the relevant information we are looking for. This leads us to the decision to extract information solely from a Wikipedia dump, or extract it live from the website.

2.3 Datasets

Extensive research has shown that there is no existing corpus to use as a dataset in this system. The datasets to train and test classifiers would therefore need to be created manually in-house, although ideally the original sentences should be found from a similar corpus to the one that the final system will process. Annotation tools such as BRAT [16] are used to label data for training and testing, although it can be done manually.

Problems can arise from creating training and testing data in-house, such as overfitting and underfitting [17]. Overfitting is where the system might pick up noise in the training data as patterns; weakening the classifier's ability to generalise. Alternatively, the system might not be able to classify the training data well because the patterns have

not been picked up correctly (known as underfitting). This can be detected by testing the classifier on the same sentences it was trained from. A classifier should ideally find the point of equilibrium between the two, maximising the ability to generalise on unseen data.

Cross validation would be useful to combat overfitting [17] but would require a large labelled corpus that may not be possible to create over the course of this project due to time and economic constraints. Generally, K-fold cross validation is an effective method to training classifiers [9]. The training data is divided into k subsets and the classifier uses k-1 sets to train on, and the final unused data set becomes testing data. This is repeated until every data set is used as the testing set. Training the classifier uses every k-1 combination so no training data goes unused. Therefore, the classifier has the opportunity to learn every pattern available.

It is noted that the train test split on a smaller dataset (up to 200 relationships) is recommended to be at around $2/3 : 1/3$ [18]. This provides the best balance for training and evaluation, but as the dataset increases the system would benefit from a larger proportion to train from like 80:20 or 90:10.

2.4 Named Entity Recognition

Named Entity Recognition (NER) involves creating an annotator that identifies key words in text. In the case of this project, the named of objects and any sizes must be labelled by the NER. A machine-leanring method we've considered to implement this uses the IOB (inside-outside-beginning) tagging format first introduced by Ramshaw and Marcus [19]. Each word is prefixed with an 'I' if it is inside a 'chunk' (named entity); 'O' if it is irrelevant to the recognition system and 'B' if it is the beginning of a chunk (for example, a name or a place). Example 1 is a sentence split up showing in more detail how this works:

Example 1.

Red [**B**; Name]
apples [**I**; Name]
are [**O**; Part of Speech]
typically [**O**; Part of Speech]
sized [**O**; Part of Speech]
at [**O**; Part of Speech]
5 [**B**; Measurement]
inches [**I**; Measurement]

In example 1, 'Red' and '5' is prefixed with B because they are the beginning of a chunk.

'Apple' is inside the same chunk as 'red' and therefore is tagged with 'I'. This would be useful to pass into the relation extractor because it can then easily pick out the entities and their positions in the sentence. Objects and units could be recognised using gazetteers as part of a rule-based system, or instead use a supervised-learning classifier. Below are some ideas that would help create a Named Entity Recogniser.

2.4.1 Gazetteers and WordNet

In order to help identify all objects and measurements in the source we could use a gazetteer as a reference. A gazetteer, used often in Named Entity Recognition [9, 20], is simply a list of words and phrases that can be matched to the text in order to pick out entities. Wikipedia has a page stating all units of length [21] that could be entered into a list. Text from a sentence would then be checked against the list, and then any match would tag the word as a unit.

Using the gazetteer as the sole feature for the classifier would make it rule-base. Research shows that WordNet [22] contains a hierarchy of all 'objects' which could be used as a gazetteer. Through using the Natural Language Toolkit [23], we can access WordNet's interface on Python to find all instances of an object on the website. While this might not cover all objects in existence and there may be some noise coming from words that are not specifically the objects we are looking for, there should be enough to provide a foundation for the work this project is going to produce. As a list of words, the objects could each be linked to a Wikipedia title. A significant drawback to this is the possibility of ambiguity; for example the Wikipedia page 'Apple Inc' could be mistaken to be a relevant page connected to the object 'apple'.

WordNet [22] has been used in previous natural language processing projects [24] in conjunction with unsupervised learning to classify unknown words based partially upon their context (surrounding words) with success. Due to the hierarchy system, WordNet could enable us to try and connect similar objects in an uncommon case of not finding the size. For example, if the program could not find the size of a Gala apple, we could find the size of a Granny Smith apple. While this might not be exactly correct, a majority of the time it is certainly a stronger solution than not giving a size to the entity.

2.4.2 Spacy

A quantity recogniser has been done before as part of Spacy's NER library [25]. Spacy is an open-source natural language processing library in Python that has preprocessing and NER features including tokenization; lemmatization; a stop list and part-of-speech tagging.

Tokenization is the process of breaking a corpus down into sentences or individual words, while lemmatization is reducing a word to its default form (for example, 'ran' would have a lemma of 'run). A stop-list is a list of words that add no valuable information to a sentence and can therefore be removed. These words include 'there'; 'when' and 'are'. It must be noted that certain stop words are useful to this system; for example,

'to' and 'in' would typically be removed from the sentence 'a human baby is 18 in to 22 in long'. In that sentence, 'to' infers a range of sizes and 'in' refers to the unit 'inches'. Part-of-speech tagging assigns word types to tokens, such as 'location' or 'organisation'.

There are three different supervised-learning models that Spacy can use to perform NER on text. They are trained from different corpora, and range in storage size, inevitably affecting run times. At 10MB, *en-core-web-sm* is the smallest; followed by *en-core-web-md* at 91MB and *en-core-web-lg* at nearly 800MB. Interestingly, the smallest model only has a performance drop of up to 1%. It remains to be seen what the most useful one could be for this project but *en-core-web-sm* appears to be the most feasible in terms of efficiency.

Using a library like Spacy could be beneficial to this project because the model is more sophisticated than anything that could be created manually. However, it cannot be said whether it would out-perform a basic rule-based system that looked for a number followed by a match to a gazetteer of units.

2.5 Relation Extraction

'Relation detection and classification' involves linking together two or more entities that have a relationship. In the instance of this project, the relationship that is being detected is solely the object to its corresponding sizes in the text. This relationship could then be broken up further into the mean size (if only one size is stated in the text); the range of sizes and the relative size. For example, the sentence *'Red apples are typically sized at 5 inches'* would have the relationship *size(red apple, 5, inches)*.

Some features in relevant sentences include the position of the object; distance from the object to the size and the number of objects or quantities between the pair the system is investigating. The best approach to classification for this part is unknown. This is because the sentences are distinguishable through the features specified (making it in favour of supervised-learning) but the sentence structure is notably repetitive, in that it follows the general pattern of 'a [object] has a length of [SIZE]'. This means a rule-based classifier could be superior.

2.6 Data Decisions

A significant problem with information extraction from a large corpus involves removing (or simply not extracting) sizes that reference a similar object with an extremely different size, compared to any other sizes identified. The cause of this is ambiguity in the named entity [9] and potential solution is to use a logarithmic scale [26] that will be able to recognise that a size difference of 30cm is significant when the size of the object is typically 10cm long; however when the object is 20m long the difference would be negligible.

2.7 Reference Resolution

Reference resolution entails a complex method of detecting anaphoric references in text (for example, representing the word 'apples' as 'they'). Detecting these instances [9] is difficult because the expressions used are also extremely common throughout the text with other objects, meaning they couldn't be classified manually using a rule-based algorithm.

Example 1. **Refrigerators** are commonly found across all households in the United Kingdom. **They** typically range in height between 4 and 6 feet.

A program that could build and maintain a set of mental representations named a discourse model [27] would solve this problem. The model consists of the connected words in the text and the relationships they're linked with. Types of anaphora, first investigated and discussed in Computer Science in 1981 [28], include *Pronominal* ('they'); *noun phrase* ('the footballer') and *one* ('Simon only ate one').

2.8 Databases

The information found after searching through a corpus must be stored in a light, simple database. The findings could be stored as a .csv file, although there would be little structure or maintainability to this (like a .txt file). MongoDB has more features than sqLite, but a simple storage design would not make use of these. Furthermore, MongoDB is not suited to highly transactional systems, making it an inferior choice compared to sqLite.

2.9 Evaluation Methods

A commonly used evaluation method to measure the consistency of natural language processing is finding the precision, recall and f-measure [9]. These measures can be found through first finding the number of true positives and negatives in addition to the false positives and negatives. A true positive is when something does belong to a class and the system assigns it correctly to that class and a true negative is where the system correctly predicts that something does not belong to that class. On the contrary, a false positive is where the system assigns something to a class when in reality it doesn't belong to that class and a false negative is where the system doesn't assign something to the class which it belongs to. Figure 2.1 shows this in a more straight-forward manner.

Sentence 1

Precision is the percentage of correctly predicted positive classifications; recall is the ratio of correctly predicted classifications to all observations in the class and the F1 score (F-measure) is the harmonic mean of these two values. The exact equations are

	Positive	*Negative*
Positive	TP	FN
Negative	FP	TN

Table 2.1: Table how a true positive; false positive; true negative and false negative is found. Italics along the top represent the predicted classification and the bold along the left represent the actual classification

below.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.2}$$

$$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{2.3}$$

Chapter 3

Requirements and Analysis

3.1 System Requirements

Through evaluating other research papers, we have developed a sense of what is feasible and what is out of our time constraints. Needless to say, there are several different approaches and techniques that could be used to solve this problem because it has never explicitly been explored before, but unfortunately we cannot implement them all. This project is investigating the feasibility of obtaining sizes of objects through examination of a large corpus, in addition to observing the performance of the NER and RE classifiers.

If the sole objective was to create an accurate database of sizes for the end user we could take advantage of Google's ability to extract key sentences or measurements and display them at the top of search results. Figure A1 (from the appendix) shows an example of searching 'size of dog' in Google. The system could simply just obtain objects and sizes from the few sentences Google can extract, but having Google extracting key sentences would undermine the objective of this project; to search through a large corpus (such as Wikipedia) filled with relevant and irrelevant sentences to obtain sizes of objects using intelligent NER and RE classifiers.

3.2 Tasks

3.2.1 Obtaining a List of Objects

Chapter 2.4.1 explains how WordNet can be manipulated to obtain a large, complete range of objects. This list could be used as a gazetteer for the Named Entity Recogniser (discussed further in this chapter). Another idea would be to use it in conjunction with the Wikipedia package [29]. This way the system could search through Wikipedia looking for the names of objects. An advantage of this would be that the system ignores several thousand irrelevant pages that are unlikely to have the sizes of objects. Therefore, this method of searching Wikipedia would save a lot of time as less data is processed.

3.2.2 Preprocessing

Although preprocessing techniques are used often in natural language processing, the extent to which they should be used here remains unknown and will be investigated in the development of this system. While fewer irrelevant words are processed, they may cause problems with the object gazetteer. The gazetteer is discussed further in this chapter, but if the gazetteer were to contain a word such as 'trainer' (which the WordNet hierarchy for an object does), when lemmatized the NER would detect the verb 'train' as an object. This is detrimental to the system because the NER will output more noise, but without preprocessing any plural of an object would not be detected (such as 'cars') because the gazetteer would only contain the lemma.

Other preprocessing techniques could potentially be used on the data. For example, stop words can be removed but it must be noted that certain stop words (such as 'to') would be useful to the Named Entity Recognisers when searching for quantities. An example of this is in the sentence, 'Red apples are 5 *to* 7 inches'. Any preprocessing techniques used in the implementation must be tested to ensure they work as part of unit testing.

3.2.3 Named Entity Recogniser

Fundamentally, the system comprises of a Named Entity Recogniser which processes sentences to provide an input for the Relation Extractor classifier. The NER classifier should use as a minimum an object gazetteer to detect objects in a sentence. The gazetteer should be extracted directly from WordNet to get a more complete list than anything that could be created manually. This will risk the gazetteer having unwanted words or phrases because the extraction will be large and automated, but it will still be the most effective method to get a gazetteer that covers many objects. The extraction process should start at the word 'object' in the WordNet hierarchy, and recursively extract every direct hyponym beneath it. Filters could be used to stop particular words from being extracted, such as verbs. An example of this would be that 'catch' is a direct hyponym of 'object', but 'catch' is certainly not an object we are attempting to determine the size of. Using manually created filters come with risks; eliminating verbs would also prevent objects such as a 'duck' from being in the gazetteer.

For size recognition in the NER, another gazetteer could potentially be used as part of a rule-based classifier to determine whether a size is mentioned in a sentence. Using the page 'Unit of length' from Wikipedia [21], we can list every possible unit of measure; inclusive of every prefix (such as 'centi' or 'milli' or the unit 'metre'). Alternatively, the system could make use of Spacy's supervised-learning Quantity Recogniser which can be trained on three different models. Ideally the best performing model could be used but the difference in size of the models (from 10MB to around 700MB) means that the efficient solution is to use the smallest model.

The object recognition must be combined with the size recognition to produce a labelled sentence, ideally containing chunks of each entity. This can then be passed into the Relation Extractor to classify relationships between the entities. Example 1 from

chapter 2.4 explicitly demonstrates what the output of this classifier should be.

3.2.4 Relation Extractor

Given that there could be a lot of noise from the NER being passed into this, a strong classifier must be created that can easily filter out incorrect relationships between sizes and entities. For example, given the sentence 'Red apples are typically sized at 5 inches', the system must be able to determine that there is a connection between the object 'red apple' and the size '5 inches'. A more complicated sentence, such as 'Red apples are sized at 5 inches, while green apples tend to be 4 inches', require the relation extractor to detect that red apples are connected to the first size ('5 inches') but not the second in the sentence, whereas green apples are connected to the second size ('4 inches') but not the first. It is sentences like these where a sophisticated classifier shows its strengths.

From the findings in chapter 2, it seems that a supervised-learning approach such as a Naive Bayes classifier would appear useful for linking objects to entities because of the different features in the sentence that are displayed. These features include the positioning of the object and quantity and the number of other entities in between. Naive Bayes would be straight-forward to implement compared to other methods and the literature suggests it is typically effective in classification tasks. Although the sentence structure appears fairly repetitive across different sentences, a rule-based classifier could be likely to struggle when working across a corpus as large as Wikipedia, because there is no way to determine every single pattern used in a website written by hundreds of different authors with a range of writing abilities and styles.

3.2.5 Baseline System

A baseline system will consist of another RE classifier that is simpler to create. It will be tested against the more sophisticated relation extractor in order to determine which one is more powerful.

The baseline relation extraction system will check if any annotated object appears in the same sentence as annotations that are sizes, ranges of sizes sizes. If this rule proves true, the Information Retrieval system will consider them connected and add the size to the database. It is anticipated that the baseline extractor will obtain every true positive relationship in the test data but at the cost of finding every single false positive possible.

3.2.6 Data Decisions

Once the data has been extracted from the text, it could be processed further before being stored in a database for the end user. A perfect system would know a way to detect classifier results that match a result found on a different web page, or a measurement that is the same as another but in different units. For example, if two relationships were found for the object X; one being 1 metre and another being 3.28 feet, the system should

be able to detect that this is the same relationship meaning that only one gets stored.

Another data decision is whether anomalous relationships could be filtered from the database, by comparing the size found to the other recorded sizes of a specific object. For example, if the size of an object 'bed' was found to be 75" long several times, and then a new relationship was found claiming that the same object 'bed' was 3.5km long, an ideal system would be able to detect that this size is drastically different and ignore it. Problems could arise with this when looking at different measurements of the same object. An example of this includes finding the length of a car to be 6 metres but the width 2 metres. The difference between them for this object is significant, but they are still both valid measurements, leading to another data decision: what measurements should be stored?

Unfortunately, data decisions such as these are out of the scope of this project, as this project is looking more at whether web crawling to find sizes is possible. It is already assumed that the perfect system will not yet be created in the first iteration of software, so we cannot expect to be able to solve every data decision problem when two classifiers must be created in addition to training data; test data and software that can web scrape from Wikipedia.

3.2.7 Data Storage

At a minimum the system should store several measurements for each object along with the Wikipedia page they were obtained from. The database should then be accessible by an end user that can simply search for an object. The output to the search could either be every size stored in the database or an average of the sizes. Storing too many measurements for the objects would make the database extremely large, given that the WordNet object hierarchy contains several thousand objects. This means that the search process could be very slow and storage of the database on an autonomous robot would be inefficient. Therefore, the ideal storage of sizes would be around two or three sizes per object as this would allow room for error with one incorrect measurement. Databases straight-forward in design but large in storage space should use lightweight database software, in order to keep the searches fast and efficient. Therefore it is suggested from the literature review that sqLite3 should be used due to its performance and reliability.

3.3 Datasets

Sentences containing valid relationships between an object and a size must be collected for the system to train from and test on. These sentences are collected from informative; formally-written sources such as Wikipedia among others [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42]. This is because Wikipedia is the source the final system will scrape information from, meaning the sentence patterns should be similar to the sentences that the system will scrape from. Sentences can either be extracted manually or through using software discussed in chapter 2, such as BRAT.

3.3.1 Dataset Issues

The lack of a corpus readily available means the sentences must be found and labelled manually. Due to the lack of variation in the sentences, it could be the case that not much data actually needs to be collected to train the classifier. This would be beneficial because this project does not have the time and money to carry out this task to a large extent. Some of the problems that can come from use of a small dataset include overfitting; underfitting and data sparsity. The lack of a large amount of balanced training data means that bias could be introduced into the classifier, and using k-fold cross-validation on such a small amount of data would not prove to be useful when a 90:10 split (which is typically recommended) means that the '10' would consist of under 20 relationships.

Additionally, a dataset split such as 90:10 or even 80:20 might not be an accurate way to test this system because the NER might not be effective enough at locating objects and quantities in every sentence. This means that, if there are 100 relationships in the total dataset, a 90:10 split would mean 10 relationships are being searched for and the NER missing entities in just two or three sentences significantly changes the precision and recall. A more suitable option within this scope would be a 60:40 split, where classifiers still train on a majority but there is room for improvement. There must also be an equal number of complicated sentences in the training and test data to ensure that the system can effectively deal with sentences containing a large number of object and size entities.

Due to the system's final use on Wikipedia, it may be beneficial to train and test the system on sentences that do not contain a relationship between entities. This would allow us to observe possible flaws in the classifiers when used on a large corpus containing both relevant and irrelevant sentences. This is a lower priority because sentences containing an object and a size measurement usually means there is at least one connection between the entities, so the classifier training off too many of these as part of a small dataset could introduce a bias.

3.4 Evaluation

A test data set must be used in order to evaluate the effectiveness of both the NER and RE classifiers. A precision; recall and F1 score can be calculated in order to factually evaluate each of the classifiers, but there also must be observations made to the outputs in order to see the specific scope of their efforts. This is because the outputs might not exactly match the gold-standard, but they could still be useful to store in the final database. For example, given the sentence, 'A red apple is 5-6 inches tall', the gold-standard pair would be *('red apple', '5-6 inches')* but if the output of the system found *('red apple', '6 inches')* it would count as a false positive in the calculations even though it is still useful and accurate information.

The NER and RE classifiers should be evaluated separately in order to observe the flaws in each. It is possible tweaks could be made to improve the scores this way and individual evaluation allows us to explore the specific scope of improvement for each

classifier. In chapter 6, it will be discussed what specific tweaks benefited the system the most. The most useful features will also be discussed if a supervised-learning approach is implemented. Generally it would be beneficial to the system to have the maximum precision possible because this would minimise the noise of the database's entries.

The only feasible way to evaluate the effectiveness of the final system used in a Wikipedia corpus is to look through the data stored and observe the general accuracy. This is because there is no easy way to automatically judge the accuracy of every single object and size stored. Although a factual 'score' cannot be placed on this, it will be relatively straight-forward to observe whether this system's outputs could be used in another autonomous robot or not by perceiving the number of incorrect entries in the database. An overwhelming amount of noise in the database would mean that this system is not yet fit for outside use, even if the F1 scores on the test data appear promising.

3.5 Scope For Improvement

After evaluating each of the classifiers, their scope for improvement will be specified. However, it is already predicted that a larger dataset than what we can create will certainly be beneficial, and more data decisions (specified earlier in this chapter) will make the final database contain more accurate information. Other improvements are not quite as predictable until the outputs of the classifiers are observed. As this is the first project to investigate this specific area of natural language processing, unfortunately not every problem can be solved. Therefore, it is expected that even a sophisticated implementation will still have a wide scope for improvement. The main expectation of this project is to create classifiers that are capable of extracting a large amount of relevant relationships between object and size entities.

One feature that would make the overall system more capable would be the detection of relative sizing. Relative sizing is a low priority in this project because sentences including it are generally rarer than sentences containing absolute sizes. It would also require a completely different classifier to detect it, costing a significant amount of time. A benefit from its inclusion would allow for objects to have their sizes estimated, but the most significant benefit would be a basic validity check on relationships extracted from text. For example, if there was a relationship found claiming that the Earth has a diameter of 1 metre, but it is already known that the Earth is larger than the moon, a validity check can run to see if the absolute measurement found of the Earth is larger than that of the Moon's. In this case it would not be, so the '1 metre' relationship would be considered an anomalous result and it would not be stored in the final database. This would be a difficult but worthwhile feature to implement.

Chapter 4

Design

4.1 High Level Overview

Through investigating the different classifier approaches outlined in chapters 2 and 3, a high level overview for the system can be made. A simplified visualisation of the system is displayed in Figure 4.1 below. Chapter 4.2 Describes how the data (such as the list of words) is obtained and 4.3 describes how the findings from Wikipedia will be stored. Chapter 4.4 provides exact information on how the Named Entity Recogniser will work and 4.5 explains how the Relation Extractor makes use of the output of the NER to detect relationships between objects and sizes. 4.6 shows what the system will look like for the end user when they are accessing the database, and 4.7 states how the system is evaluated to give a fair outlook on the performance.

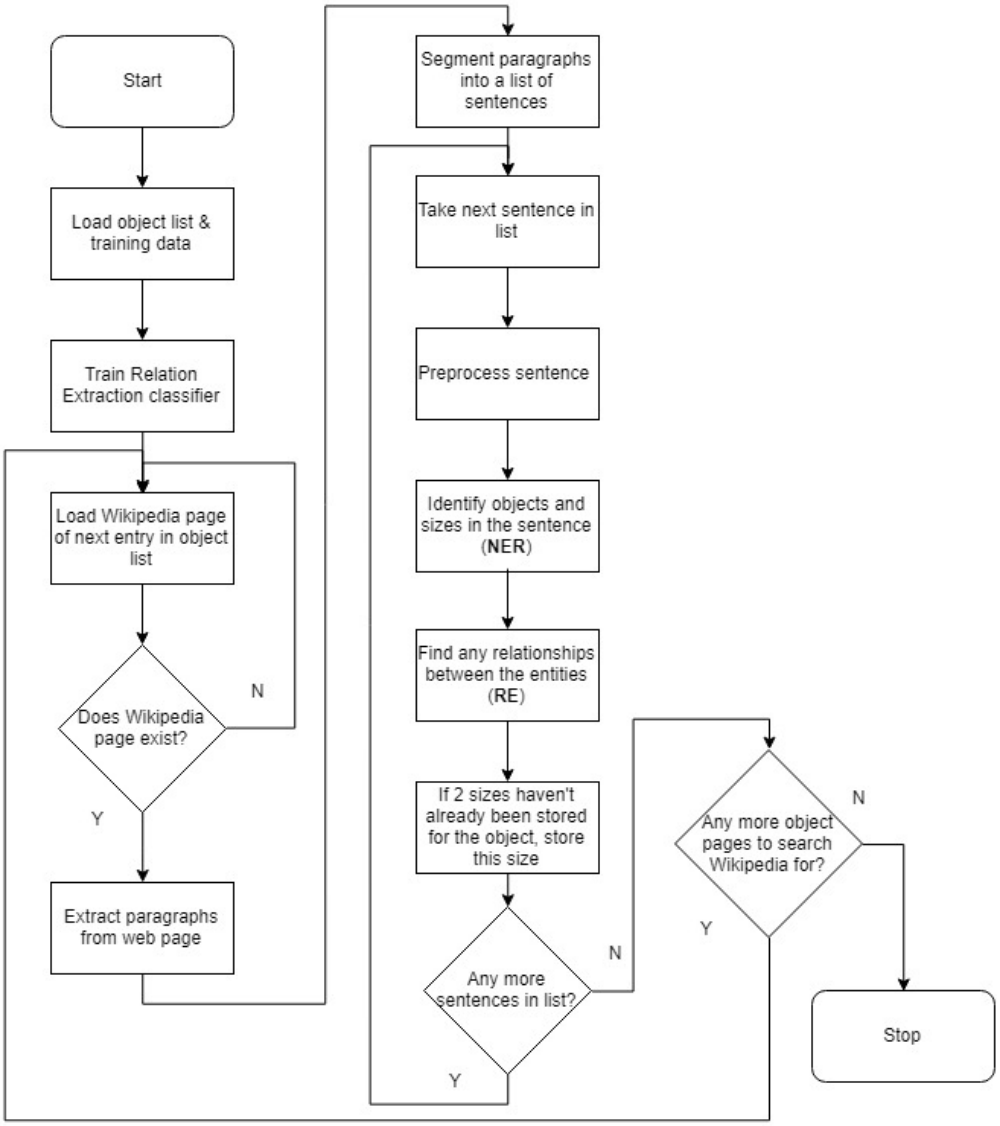

Figure 4.1: Diagram representing the flow of the system

4.2 Source of the Data

Before any classification can be done by the system, a list of objects must be created. This list will be used as a gazetteer for objects in the NER and as a list of Wikipedia pages to search through when extracting information from the web. WordNet appears to be the most appropriate source to extract the list from because it covers a wide range of living and inanimate objects. WordNet contains the noun 'object'; from there we can recursively search for the direct hyponyms (more specific words) of each following noun. This list will be lemmatized and then stemmed in order to minimise the size of it. This also means it will be able to match words in the sentences after they are stemmed during preprocessing (explained in 4.3.2).

From observations made in chapters 2 and 3, the decision has been made to extract test solely from the Wikipedia website. The library named 'wikipedia' in Python allows

the program to effortlessly obtain paragraphs from a Wikipedia page it searches for. The system will iterate through the object gazetteer, searching for a Wikipedia page with a title matching that object. If one is found then the paragraphs will be obtained from that page, and the NER and RE process will begin on each sentence. Any the size of any object can be found from any of the pages. The searching will be real-time and not from a Wikipedia dump. Searching the internet will take a large amount of time but it will ensure that the system extracts only from the relevant pages. The end user will only interact with a database which can be used offline.

4.3 Storing the Data

When a relationship is found, the object; size and Wikipedia page it was found from will be stored into a sqLite3 database structured as specified by the UML diagram below. Up to 2 sizes will be stored for any one object and there will be no verification process to check if the size is correct or similar to other current entries because that is a lower priority than improving the classifiers. The ideal database format would store a range of sizes and a commonly found mean size, but there is no guarantee of finding any of these sizes in an inconsistent corpus as large as Wikipedia and as a result of this we decided to store only the final 2 sizes found. This simple database is visualised in the UML diagram below.

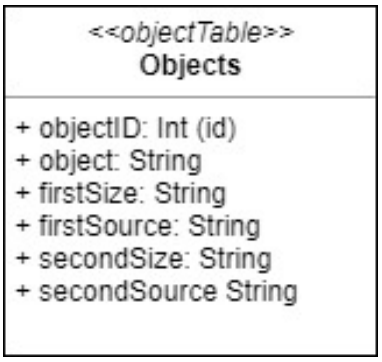

Figure 4.2: UML diagram of the data stored by the system

4.4 Named Entity Recogniser

4.4.1 Overview

The Named Entity Recogniser must take in a block of text and output labelled data that can be used for classification of any objects and quantities. The way this will work is by breaking a block of text into paragraphs. The sentence/words will then undergo a scan for any measurements using Spacy's supervised learning model, followed by a rule-based object recogniser (or 'lookup') using a gazetteer. Once all of the sizes and objects are detected they will be added to a collective list of entities. This list will be passed into the Relation Extractor, along with the preprocessed sentence.

4.4.2 Pre-Processing

NLTK will be used to segment each paragraph into sentences. From some experimentation this does not appear to cause any inconsistencies compared to using Spacy, which occasionally linked together two sentences thinking that they were one. Every sentence will undergo stop word removal (where useless words such as 'they' and 'with' are removed); all text will be made lowercase and all words are also stemmed. Preprocessing will minimise the number of words going into the system.

4.4.3 Size Recognition: Spacy

Sizes will be detected using the Spacy library [25]. While it is uncertain that using a supervised-learning approach will be superior given how few patterns there are for quantity phrases, it is certain that using Spacy's well-trained library on thousands of sentences will be superior to one we could make with limited hand-crafted training data made over this development process. Spacy will detect any size as a quantity entity and will not pay attention to whether it is detecting a range of sizes ('5-10 cm') or an individual size ('10cm'). This means that the gold standard we create must contain the longest one (range) as it cannot contain both entries without disrupting the evaluation calculations. Furthermore it detects sizes as a magnitude and unit of measurement combined, rather than separating the two, which is not a problem unless it occasionally leaves out the units.

The Spacy model is not just looking for sizes but also other forms of quantity types such as weights of items. To eliminate these unwanted entities we can prevent any quantity entity that contains a weight unit from being added to the chunks (discussed in more depth later in this chapter). Spacy also cannot detect exactly what kind of size it has found. Although this is arguably a Relation Extraction problem, it should be addressed that we will consider every size quantity to have a possible relation to an object entity because just obtaining the height and not the width provides a better estimation than having extracted nothing at all. Being able to distinguish the type of each quantity would be out of the scope of the project, so we will store any size as a match if the relation extractor decides upon it.

Spacy uses pre-trained statistical models to identify entities in the text. We will experiment with all three available models (SM; MS; LG) to determine which is the superior model to extract quantities from text, and thus which model will be used as part of the extraction from Wikipedia web pages. It will recognise more than just quantity entities such as locations, money and dates, and we have decided to remove these from the final chunks in order to reduce the time wasted when the relation extractor is scanning through the NER's output.

4.4.4 Object Recognition: Gazetteer

Words in a sentence will be compared to the list created by the WordNet hierarchy, referred to as the object gazetteer. This means that the NER is 'rigid'; relying on the gazetteer's completeness and accuracy but it is more efficient and guarantees that 'basic' objects (which are fundamental to the WordNet hierarchy) will always be found.

It is predicted that there will be unwanted words in the object gazetteer. In anticipation of this, a small function that checks if the 'object' is a noun or not has been implemented. All of the objects we wish to find the size of are nouns. Therefore, this feature will hopefully reduce the chances of incorrectly labelled objects (such as 'depend') from having a relationship to a size, causing a lower number of incorrect relationships (false positives).

To maintain simplicity and a low runtime, the gazetteer will be loaded in as a set from a text file. We will ignore the extraction of object synonyms (synsets) to keep the gazetteer to a minimal size, at the potential cost of missing a small number of objects. In order to search through the text looking for objects we will use a window. This window will be the same number of words as the maximum number of words in an object from the gazetteer. The window size will gradually decrement by one until it is looking at a single word or an object is found (the window of words matches an entry in the object gazetteer). This greedy method of object detection was chosen over the efficient finite state machine because of its simplicity to set up, providing more time to develop the relation extraction classifier. An example of the greedy recogniser if below:

```
Sentence: The apple is 6 cm in height
'The apple is 6 cm' in height - Not an object
'The apple is 6' cm in height - Not an object
'The apple is' 6 cm in height - Not an object
'The apple' is 6 cm in height - Not an object
'The' apple is 6 cm in height - Not an object
The 'apple is 6 cm in'| height - Not an object
The 'apple is 6 cm' in height - Not an object
The 'apple is 6' cm in height - Not an object
The 'apple is' 6 cm in height - Not an object
The 'apple' is 6 cm in height - Object found, add to chunks of entities
```

Figure 4.3: Iterations showing how the object window changes in order to identify an object in the sentence. Selection window indictaed by ''

4.5 Relation Extractor

4.5.1 Overview

The relation extractor links objects to sizes in a sentence. A baseline relation extractor will be created and a more sophisticated Naive Bayes Classifier will be created. Using Naive Bayes as a supervised-learning classification method is straight-forward to implement but is still very effective when the features it learns from are distinguished and descriptive.

The relation extractor will look at just one sentence at a time; taking as an input the original sentence in addition to the output of the NER (labelled sentence and chunks of entities). It will first ensure that there is both an object and quantity in the sentence by observing the chunks and from there it will determine the relationships between the entities. The final output of the relation extractor will be a list of each combination of objects and quantities, tupled with their relation ('SIZE-OF' or "NO-RELATION"). A simplified diagram of this can be seen below.

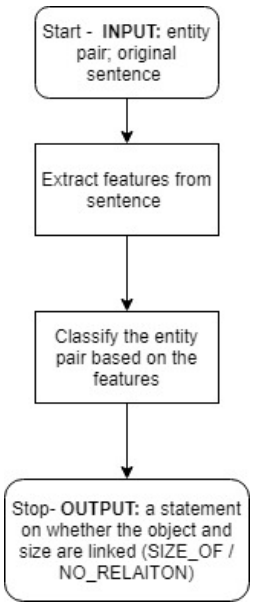

Figure 4.4: Simplified diagram of what the Relation Extractor does with each pair of entities

4.5.2 Baseline Extractor

Given a sentence, the baseline extractor will take every object and simply assume it is related to any other quantity in the sentence. Although basic, we predict for simple sentences this could prove effective. We will certainly observe its limitations in sentences describing multiple objects with multiple measurements, therefore it is important to use test data that contains a varied mixture of sentences to observe the usefulness of this method in comparison to the Naive Bayes Classifier.

4.5.3 Naive Bayes Classifier

This classifier is going to be created with the objective of outclassing the baseline. After implementation and extensive testing, the strength of this supervised-learning method can be confirmed but at a minimum it is expected that this classifier can reduce the pairs of entities that clearly have no relation. NLTK's library [23] will be used to train, test and present the best features of the classifier.

This classifier will require training data to learn from and a set of features to make use of to distinguish a pair of entities that are related, from a pair of entities that are not. We will craft our own training data to pass into the classifier. The training data will consist of a list of sentences taken from Wikipedia and will be varied to cover a

range of sentence structures. The sentences will be broken down into chunks of object and quantity entities to make a new list in addition to another list containing the related pairs of entities. These three lists will be used by the classifier to learn patterns in the text. We aim as a minimum to pass in one hundred relationships to the classifier, ensuring it is well-trained to distinguish relationships from non-relationships.

After training is complete, the classifier will take in a list of test sentences to determine the relationships in the test data. The text will be broken down into the individual features, and the features will be parsed into the classifier (in the same way it is trained). Beneath the list below is an explanation for each, but features that we are going to investigate include:

1. The distance from the object to the size in the sentence.
2. Whether there exists an object between the entities.
3. Whether there exists a size between the entities.
4. Whether the object comes first in the sentence.
5. Whether the object is a noun.

4.3.3.1. Distance from Object to Size. *Returns: integer.*
The distance between the pair in the original sentence appears to be important because a larger distance seems to indicate a smaller chance of the object and size being related. Thus, this will be a function that, given the sentence; object and size; will return the number of characters between the pair. The number of characters was chosen because there will be a larger variation in space between the pair, compared to simply 'three words' compared to 'four words'.

4.3.3.2. Existing Object Between the Pair. *Returns: Boolean*
Typically, when there is an object between the pair of entities we are investigating, the size tends to represent the object between them and not the object in our pair. This leads us to believe that this will be a strong feature to add because it will allow the classifier to eliminate potential relationships, reducing the false-positives the system might otherwise retrieve.

4.3.3.3. Existing Size Between the Entities. *Returns: Boolean*
This feature will detect whether there is another size between the pair undergoing classification. In foresight it might not be as useful as observing the presence of an object because there can be two measurements in a sentence for one object; typically expressing the same size with different unit of measurement. Nevertheless it is worth implementing and experimenting with to identify its usefulness.

4.3.3.4. Whether the Object Comes First in the Sentence. *Returns: Boolean*
The sentences we observed from preliminary investigations very commonly matches the structure of: *'The [OBJECT] is [SIZE] long.'* From this we can infer that the order of the entities in the sentence will hold valuable information when determining the relationship between them.

4.6 User Experience

The objective of this project is to determine the feasibility of obtaining sizes of objects from Wikipedia in order to store them in a database and search for them in the future. As stated previously this system would likely have a user simply looking for a magnitude and unit; meaning that the user experience relies on effectively showing a size and not on a aesthetically pleasing user interface. Thus, the sizes will be accessed using a command line interface where the user searches for an object and the system will look up for an entry of that object in the database. If one exists it simply outputs it, along with the Wikipedia page the data was extracted from. Below is a clear example of the user interface to be used:

```
 Type in the object you wish to find the size for: lizard
The object:  lizard
 -  Has a size of:  4 feet . Found from: https://en.wikipedia.org/wiki teiid_lizard
 -  Has a size of:  20 cm . Found from: https://en.wikipedia.org/wiki earless_lizard
```

Figure 4.5: An example search from the user interface

4.7 Evaluation

When evaluating the system, the results will be closely examined to see if there is any way to improve the classifiers. If a new fix is found, it will be implemented and tested showing the evolution of the system through thorough experimentation.

As discussed in previous chapters, the most honest and balanced way to measure the effectiveness of classifiers is by obtaining the F-measure from their output. The system will be evaluated in this way against test data we have created. This way the experiment can be controlled, unlike the web crawl through hundreds of Wikipedia pages. The final evaluation will be on the database created. This will be performed by examining the entries of a group of objects in the database.

The test data, akin to the training data, will be manually created by obtaining sentences from Wikipedia; chunking the object and size entities; and recording the relationships between them. There will be at least 100 test relationships to determine, and they will be randomly selected from the total relationships and sentences we prepare.

The NER will be assessed on its effectiveness to withdraw the correct objects and quantities from a sentence. Although numeric evaluation scores will be useful, me must more importantly observe the raw data coming out of the classifier because that allows us to see the extent to which it is going wrong. For example, obtaining '10mm' from '5-10mm' would not count as a direct match to a gold standard, but it is still a relevant and useful extraction from the data.

The Relation Extraction's effectiveness will be directly affected by the output of the NER. This means that the Relation Extraction classifier must be tested against a gold standard input as well as the input coming from our NER, to observe the extent of its usefulness if the NER worked perfectly. In addition to calculating the F-measure we must also observe the precision it scores. This is important to the system because it

allows us to determine the usefulness when using this system to retrieve data from the internet; a large number of false positives would mean the database is disproportionately full of incorrect data (noise).

Evaluation of the user interface will be carried out by objectively examining the ease of use of retrieving a size from the database.

Chapter 5

Implementation and Testing

5.1 System Overview

A flowchart of the system processes is below. Everything is in the same file because the code is not extraordinarily long. Therefore, with a good structure and use of several functions, the code is still highly maintainable.

MAKE FLOWCHART AND PUT IT HERE

After splitting the dataset randomly into train and test lists, the Naive Bayes classifier can be trained and the system can be tested. The test sentences must first be preprocessed; removing stop words and forcing all test to be lowercase. These sentences are then passed into the Named Entity Recogniser which detects sizes using Spacy and objects using the object gazetteer.

5.2 Dataset

5.2.1 Creation

There are 121 positive relationships in the manually created dataset, in addition to 50 negative relationships, making for 171 relationships in total. The dataset was created manually, due to difficulties using BRAT. This took up a significant amount of time, affecting the total number of relationships that could be created within the time constraints of the project.

5.2.2 Train Test Split

There is a randomly generated 60:40 split in the training:test relationships. This ratio is against the advice from the literature survey, which shows that typically there is an 80:20 split or even 90:10. However, When it came to testing the overall system (NER and RE together), only 21 correct relationships between chunks *could* be found out of 49 due to the faults of the NER (using a 60:40 split). Therefore, any lower than a proportion of 40 for the testing data meant the number of positive chunks that could be found using the NER and RE were even lower. In turn this meant that missing just one positive relationship would decrease the recall by over 5%. This would have an unpredictably large effect on results, but the training data still must have a majority of the total dataset

meaning that 60:40 was the most balanced.

The training data consists of 71 positive relationships between sizes and objects. This was enough to show the Naive Bayes' classifiers superiority (discussed further in the next chapter), meaning that the 60:40 split was justified. Due to the relationships being written out manually, it was necessary to test the chunks to ensure that they all exactly match the test. One way to do this was to use the baseline classifier (that is guaranteed to find every correct positive relationship in a group of sentences) and run it on both the training and test data. When the output was a precision of 1.00, it was confirmed that the training and test data was accurately created, without mistakes.

5.3 Unit Testing

5.3.1 Overview

This section tests some key parts of the coding to ensure the outputs are correct. They are carried out independently meaning they aren't influenced by the outputs of any other processes in the system (when they would be if this system was running normally).

5.3.2 Preprocessing

It was ensured that the preprocessing function was thoroughly tested in order to ensure no important information was accidentally removed from the sentences. Unfortunately, because the structure of the sentence changed when the stop word removal process was done automatically using NLTK or Spacy, the code had to be written such that the words were removed manually. For example, '5mm' in the sentence, when broken down and put back together again, would change to '5 mm'. Below are the results. An unusual finding was that the removal of a stop word would somehow be detrimental to results if it was the first word in the system. This meant the occasional stop word was left in, causing a negligible change in run time with improved results. This may have been caused by a logic error but the output appears flawless.

- **Input Sentence:** "Female bears measure from 130 to 190 centimeters (50 to 75 inches) and weigh 40 to 80 kilograms (90 to 175 pounds)."
- **Expected Output:** "female bears measure 130 to 190 centimeters (50 to 75 inches) weigh 40 to 80 kilograms (90 to 175 pounds)."
- **Actual Output:** "female bears measure 130 to 190 centimeters (50 to 75 inches) weigh 40 to 80 kilograms (90 to 175 pounds)." - **CORRECT**
- **Evidence:** Figure A.2.

As seen in the table, preprocessing works to perfection.

5.3.3 Object Tagging

Below is a table proving the object recogniser works. Spacy automatically tags quantities, so it is already assumed that is functional.

- **Input Sentence:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **Expected Output:** FOUND: ('apple trees'), FOUND: ('crown')
- **Actual Output:** FOUND: ('apple trees'), FOUND: ('crown') - **CORRECT**
- **Evidence:** Figure A.3.

5.3.4 Classifier Features

The features must be tested to ensure they output the correct observations, before the Naive Bayes classifier can be used on test data. The tests below show that the features work as expected.

'objToQuant': 'medium' 'isObjBetween': False 'quantsBetween': 0 'isObjFirst': 'Yes'

isObjFirst

- **Input Sentence:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **Input pair:** apple trees, 13-39 ft
- **Expected Output:** 'isObjFirst': 'Yes'
- **Actual Output:** 'isObjFirst': 'Yes' - **CORRECT**
- **Evidence:** Figure A.4.

isObjBetween

- **Input Sentence:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **Input pair:** apple trees, 13-39 ft
- **Expected Output:** 'isObjBetween': False
- **Actual Output:** 'isObjBetween': False - **CORRECT**
- **Evidence:** Figure A.4.

quantsBetween

- **Input Sentence:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **Input pair:** apple trees, 13-39 ft
- **Expected Output:** 'quantsBetween': 1; (4-12 m is between them)
- **Actual Output:** 'quantsBetween': 1 - **CORRECT**
- **Evidence:** Figure A.4.

objToQuant

- **Input Sentence:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **Input pair:** apple trees, 13-39 ft
- **Expected Output:** 'objToQuant': 'medium'; 18 characters between them therefore medium
- **Actual Output:** 'objToQuant': 'medium' - **CORRECT**
- **Evidence:** Figure A.4.

While testing the objToQuant feature, we found that it was highly beneficial to approximate a distance to return ('close'; 'medium'; 'far') rather than an absolute distance of characters (which was the initial design). This is due to the fact the Naive Bayes classifier would be able to generalise much more easily on a smaller training set rather than overfitting the training data.

An example of this is as follows; there could be a distance of 5 characters (which would count as 'close') between the object and quantity where there is not a positive relationship. This would be considered anomalous but because there are no other instances in the training data that have a distance of 5 characters, the classifier can only train from this one example meaning that it cannot generalise on a large corpus such as Wikipedia. Thankfully, unit testing spotted this design flaw and fixing it, boosting the performance of the classifier when running against the test data.

A 'close' distance is considered to have two words between the entities; 'medium' is between two and four and 'far' is over four. The average word length in the English language is typically just over five words [43] but we have assumed it to be 4.5 due to stemming words in the sentences. Therefore a distance of two words is 11 characters (including whitespaces); 22 maximum for 'medium' distance and over 22 for a 'far' distance.

5.3.5 Precision, Recall and F-Measure

Unfortunately, there was no success evaluating the classifiers using the readily-made module [44]. This meant that the evaluation process had to be calculated manually, therefore this process must be thoroughly in order to ensure that the final evaluation of the system is fair and accurate. The code snippets can be seen in the figures below:

```
if classifier.classify(featureResults) == 'SIZE_OF': #if classification is positive

    if((o,q) in testRelationships): #if gold standard says true positive
        relRet += 1                 #True positives increase by 1
        relNotRet -= 1              #False negatives decrease by 1 (Starting value = length of relationship list)
        testData.append((featureResults, 'SIZE_OF'))
    else:

        irrelRet += 1               #False positive increments
        testData.append((featureResults, 'NO_RELATION'))
```

Figure 5.1: Where decisions are made on whether a result is a true/false postive or true/false negative

```
# GET RESULTS ----------------------------------------------------
def getPRF(relRet, relNotRet, irrelRet):
    precision = relRet/(relRet + irrelRet)
    recall = relRet /(relRet + relNotRet)
    fmeasure = (2 * precision * recall) / (precision + recall)

    print('--------- Results ---------------')
    print('Precision: {:4.2f}'.format(precision))
    print('Recall: {:4.2f}'.format(recall))
    print('F-Measure: {:4.2f}'.format(fmeasure))
```

Figure 5.2: Where calculations are made to find Precision; Recall and F-1 score

The code was run on a simplified test set specified below:

Given the set of relationships {a,b,c,d,e}:

- **Chunks returned by classifier:** {a,b,c}
- **Chunks left out:** {d,e}
- **Expected Chunks (gold-standard; correct positives):** {a,d}

Therefore, one can observe that the following scores should be found:

	TP	FP	TN	FN	Precision	Recall	F1-Score
Expected	1	1	1	2	0.33	0.50	0.40
Actual	1	1	1	2	0.33	0.50	0.40

Table 5.1: Expected calculation vs Actual Calculation (TP = True Positive; TN = True Negative; FP = False Positive; FN = False Negative

Below is an explanation for the values:

- TP = 1 because 'a' was successfully found.

- TN = 1 because ‘e’ was successfully left out.
- FP = 2 because ‘b’ and ‘c’ were accidentally found when they aren’t in the gold-standard.
- FN = 1 because ‘d’ was left out when it should not have been.
- Precision = TP/(FP+TP) = 0.33
- Recall = TP/(TP+FN) = 0.5
- F-measure = 2 * ((P * R)/(P + R)) = 0.40

The F-measure calculator works exactly the same as every other one in existence but the difference is this one works with our results.

5.4 Wikipedia Web Crawler

5.4.1 Scraping the Web Pages

The initial idea was to download a Wikipedia dump and run the code through it. In practise, the dump was sized at over 60GB, meaning that even on a powerful system, the file still could not be loaded into memory to process. Instead of breaking the file down, the decision was made to repeatedly search Wikipedia real-time for each word or phrase in the object gazetteer. If a search result was found, that object’s page would have its paragraphs extracted to run the system on. The system would be looking for an instance of any object across the pages it looked through, instead of looking just for an object corresponding to the title of the Wikipedia page.

5.4.2 User Interface

The user interface to interact with the database is simply a command line. When the object has been entered, the database is scanned and the stored sizes are returned. This works exactly as expected.

The tests carried out below checks if the object and size appears in the source specified by the database. They also briefly discuss whether the relationships stored are relevant.

```
Type in the object you wish to find the size for: box
The object:  box
 -  Has a size of:  2.5 cm . Found from: https://en.wikipedia.org/wiki/cat_box
 -  Has a size of:  36 inches . Found from: https://en.wikipedia.org/wiki/racquetball
```

Figure 5.3: Searching through the database to find size of ‘box’

The figure above sets out the following test cases:

Test case 1

The words ‘box’ and ‘2.5 cm’ appear in the same sentence at the url: https://en.wikipedia.org/wiki/cat_box

Evidence:

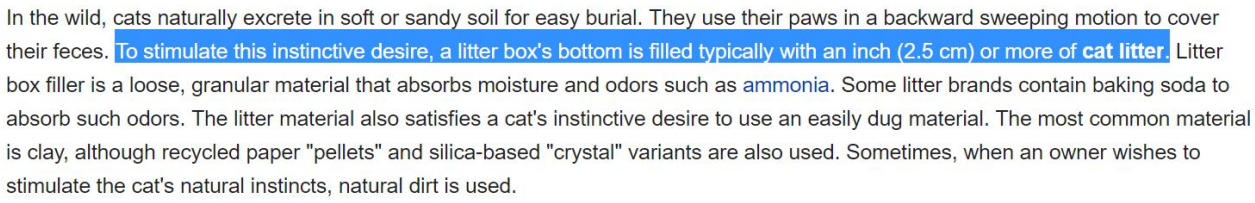

Figure 5.4: Going to https://en.wikipedia.org/wiki/cat_box to test if the entities exist

The above paragraph contains both entities and thus, the highlighted sentence was where the relationship was extracted. An interesting point to note is that Wikipedia redirects that URL to https://en.wikipedia.org/wiki/Litter_box, but this does not matter as they mean the same thing. Unfortunately, this pair is not a true relationship because the size refers to the depth of the cat litter, not the size of the box.

Test case 2

The words ‘box’ and ‘36 inches’ appear in the same sentence at the url: https://en.wikipedia.org/wiki/racquetball

Evidence:

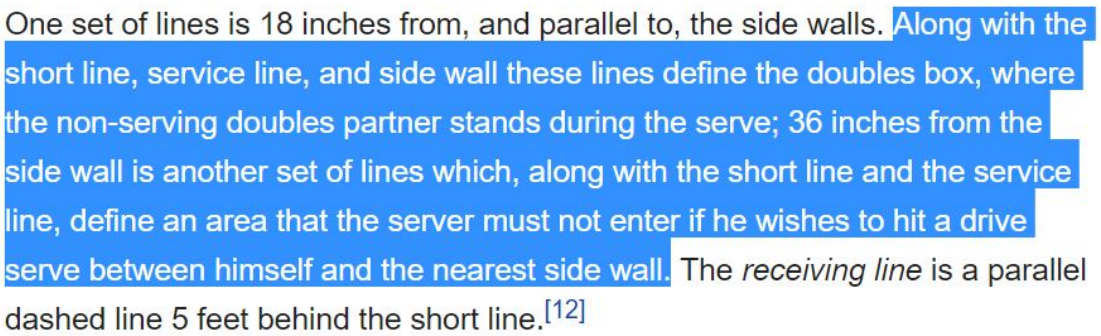

Figure 5.5: Going to https://en.wikipedia.org/wiki/racquetball to test if the entities exist

It can be seen in the highlighted sentence that the words ‘box’ and ‘36 inches’ appear. Once again this means that the relationships are correctly stored in the database with their Wikipedia sources, but it is disappointing to observe that neither size was related to the object when the context of the sentence is understood. In this case, the ‘box’ is not referring to a physical entity but the layout of a squash court.

Chapter 6

Results and Discussion

6.1 Overview

In this chapter there will be several experimentations documented. They will follow the format of explaining the set up and aim of the test; the results and then an analysis of what the results show. For each part of the system, the future possibilities are explained if this project was taken in more depth. The training and testing data is set up according to 5.2.2.

6.2 Pre-Processing

The aim of this experiment is to determine the effectiveness of preprocessing on the system. The system runs two separate times; once using preprocessing (stop word removal; lemmatization; stemming and converting text to lower case) and once without any kind of preprocessing. The table below shows the results.

Experiment	Precision	Recall	F1	Baseline F1	Naive Bayes F1
Without Preprocessing	0.47	0.55	0.51	0.27	0.31
With Preprocessing	0.52	0.59	0.55	0.32	0.38

Table 6.1: Effects of Preprocessing on the NER and Overall System

Preprocessing was beneficial to the system because more words now match entries in the gazetteer. It can be observed that before preprocessing, words such as 'doors'; 'leaves' and 'Apple' are now detected as objects and rightfully so; more correct object chunks allow for more correct relationships to possibly be found.

6.3 NER

6.3.1 Object Recognition

In the implementation, two different WordNet lists were extracted. One list had 26,000 entries while the other had 24,000. As stated previously, the smaller list is the larger list after the removal of any 'object' that is not a noun. This experiment determines whether

the removal of any non-noun 'objects' is a correct decision. The results are shown in the table below.

	Precision	Recall	F1	Baseline F1	Naive Bayes F1
Gazetteer with Non-Nouns	0.44	0.60	0.51	0.27	0.32
Gazetteer without Non-Nouns	0.52	0.59	0.55	0.32	0.38

Table 6.2: Effects of Lemmatizing words being checked against the object gazetteer

The results show the trade off between a smaller (therefore more efficient) gazetteer, and a larger gazetteer that contains more noise but also some more relevant objects. The larger gazetteer finds 1 more correct chunk, but the precision suffers for this advantage. Judging from the results it can be seen that the larger gazetteer contains incorrect object words such as 'with' and 'depend'. As some of these incorrect object words are very common (appearing in most sentences), the smaller gazetteer appears to be much more useful for use across a large corpus such as Wikipedia as there will be much less noise.

A drawback to both lists is the lack of some specific objects. Examples from our test data show the gazetteer does not contain 'squid giant axon' and 'glacial ice', even though the gaztteer contains 'axon' and other types of ice existing in the list. This could likely be fixed by expanding the search through WordNet, but it would also introduce more noise from incorrect objects.

Another potential improvement to the object recogniser would be the use of supervised learning to label objects in the text. This is because a supervised-learning classifier can look at the context of a word in a sentence. In theory this would allow the system to distinguish words that are used both as a verb and noun to see if they are referring to an object in that sentence. For example, the sentence 'Jon goes fishing' does not have an object, but when it is lemmatized and compared to the gazetteer, the current rigid system would find 'fish'. A supervised-learning classifier could be able to detect that the word is not referring to an object but instead a verb.

6.3.2 Size Recognition

Spacy Models

1st Iteration Spacy's supervised-learning classifier can be trained on 3 different models. Each one varies in size and have been trained using different corpora. This experiment has the objective of determining the most effective model for size recognition.

Spacy Model	Size (MB)	Runtime (s)	Precision	Recall	F1	Baseline F1	Bayes F1
SM	10	08.5	0.46	0.56	0.51	0.24	0.29
MD	91	20.1	0.44	0.60	0.51	0.25	0.26
LG	788	19.0	0.44	0.54	0.48	0.24	0.24

Table 6.3: How the Spacy model used affects performance

What was most instantly most noticeable when running the program was the change in run-time when using each model. Due to their difference in size, loading in the larger models took more time, but the processing time on sentences were the same. Strangely, the medium-sized model had a large average run time than the largest model, in spite of the significant difference in size. However, an interesting finding is that the largest model proves to be the least effective; finding the fewest relevant measurements thus producing the worst F-measure for the overall system (displayed in the table below).

MD produces the most correct sizes (seen through the highest recall) but also detects a significant amount of noise with it. SM, running the quickest by some distance, provides a balance, producing little noise but still getting a satisfying amount of sizes. The SM model will therefore be used in the information extraction task performed on Wikipedia.

The size classifier is certainly the most difficult part of this system to implement; as we are not making our own recogniser we attempted to improve Spacy's somewhat inconsistent one. Having observed from testing all models that chunks that occasionally sizes such as '5 to 10 m' would be processed as two different chunks: '5 to'; '10 m'. Due to the design of the gold standard, the system must always detect a range of measurements in this fashion as one single chunk. A solution was implemented to form a second iteration of experimentation.

2nd Iteration A quick solution to this problem was to connect two measurements that come directly one after another, forming one single chunk. This experiment determines whether this quick fix proves to be an improvement to the NER. The results are specified below. The model used in this experiment was SM (the best found from the first iteration).

	Precision	Recall	F1	Baseline F1	Naive Bayes F1
Without Fixes	0.45	0.56	0.50	0.24	0.32
With Fixes	0.52	0.59	0.55	0.32	0.38

Table 6.4: How improving the Quantity Recognition changed performance

The results proved that this hotfix to Spacy's model improved the performance of the system, so it was necessary to perform the experiment on the Spacy models again to see how they are affected by the new changes.

Spacy Model	Size (MB)	Runtime (s)	Precision	Recall	F1	Baseline F1	Bayes F1
SM	10	08.5	0.52	0.59	0.55	0.32	0.38
MD	91	20.1	0.47	0.60	0.53	0.28	0.30
LG	788	19.0	0.47	0.54	0.50	0.27	0.27

Table 6.5: How the Spacy model used affects performance

Clearly all of the models benefit from this improvement, but SM remains the most useful model for extracting information from Wikipedia. The precision improves across all models because two incorrect chunks now become one correct chunk for every instance of this problem. On average, the run time of each model did not change.

Regardless of the minor fixes made to the Spacy output, the chunks were still noticeably inconsistent. It struggled particularly when detecting measurements using the units 'in' (short for inches). This was commonly used across the Wikipedia pages but was never picked up by the NER, as observed by seeing the quantity output chunks. This is likely due to the vast utility and frequency of the word; using it as a unit of measurement is a low probability given how often it appears in any English sentence. Furthermore, the NER was observed to occasionally include words like 'roughly' in the quantity chunk. This in theory could be useful for a more sophisticated system that could make use of the extra information, but in our system it was just seen as noise that prevented a match to a gold standard entry. Although this recogniser does a satisfactory job at recognising sizes, it would be stronger if 'in' could be detected as 'inches' and it could be consistent in the chunks it detects.

Scope for Improvement

An interesting experiment that could be made would be to use a rule-based system and compare its effectiveness the classifier trained by Spacy. Spacy boasts an 87% F-Measure using SM, but this accounts for all kinds of entities and could be significantly worse when detecting specifically sizes. The structure of sentences containing relationships appears to be repetitive and given more time in this experiment we would have implemented another rule-based system to test against that judges a phrase based solely on its grammatical structure. The findings from chapter 2 suggest this would have been a useful approach when there could be so few rules to implement. This was a low priority however, given Spacy's NER system is well-developed and it was unlikely we would create a classifier to top it. Additionally we have already seen the limitations of a rule-based classifier first-hand, when evaluating the effectiveness of the object gazetteer.

A significant improvement that could be made (out of the scope of this project) would be to judge the relative sizing of an object by taking an object and its measurement in addition to the structure of a sentence. This addition would prove beneficial when using the program to scrape information from Wikipedia because the size of some objects are not always explicitly mentioned, meaning that more objects will be measured with this feature implemented. An example of this feature would be seen in the sentence 'Jupiter has more than 11 times the diameter of Earth'; the system could infer that Jupiter is

larger than Earth, meaning that the minimum size of Jupiter is Earth's recorded size in the system.

6.3.3 Results Sampling

The overall score for the NER classifier is below. This has been seen across all other tables but to mkae it clear, these were the overall best results the system could achieve. While it is not bad, it is arguably not good enough to use in a corpus as large as Wikipedia because of the noise it will pick up. It is inferred that there would be the same number of correct relationships as there are incorrect relationships retrieved (as the precision is 0.52).

Sentence 1

Spacy Model	Runtime (s)	Precision	Recall	F1-Score
SM	08.5	0.52	0.59	0.55

Table 6.6: Final Results of NER

Three sentences below have been picked out from the NER results. We will discuss where the classifier performs or falls short. Note that the inputs below have been preprocessed, hence they do not make grammatical sense. Text in green indicates a correctly identified chunk whereas red text is an incorrect chunk.

Sentence 2

- **INPUT:** the 13 otter species range in adult size 0.6 to 1.8 m (2.0 to 5.9 ft) in length 1 to 45 kg (2.2 to 99.2 lb) in weight.
  - **OBJECT CHUNKS FOUND:** ['otter', 'range', 'adult', 'weight']
  - **SIZE CHUNKS FOUND:** ['0.6 to 1.8 m', '2.0 to 5.9 ft']

This sentence shows the NER working well; all of the correct objects and sizes have been found. Unfortunately this recall comes with some noise. The word 'weight' does not have the meaning of an object in the sentence; it is describing a feature (i.e the mass) of the otter species. This is also the case for 'adult'; a perfect NER would be able to identify that these two words in this case are not objects due to the context of the sentence.

Sentence 3

- **INPUT:** the leaves 3–10 cm (1.2–3.9 in) long, alternate, simple, serrated margin.
  - **OBJECT CHUNKS FOUND:** ['leaves', 'alternate', 'simple']
  - **SIZE CHUNKS FOUND:** ['3–10 cm']

Once again some of the correct objects have been identified but there is still noise. Only one of the two sizes were identified here. As stated earlier in this chapter, Spacy's size recognition cannot detect any measurement that has the unit 'in' (for inches). The output shows this problem when '1.2–3.9 in' is not detected. The object gazetteer has two more objects in it than it should have, further proving the inconsistencies with the object gazetteer.

- **INPUT:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
  - **OBJECT CHUNKS FOUND:** ['apple trees','crown']
  - **SIZE CHUNKS FOUND:** ['13–39 ft']

In this case the size recogniser is missing a simple size, '4-12 m', and the object recogniser accidentally recognised 'crown'. The word 'crown' refers to a part of the apple tree, but the system identifies it to be an independent object due to the rigidity of the look-up function in the object recogniser.

6.4 Relation Extraction

6.4.1 Overview

The performance of the relation extractor in this project is entirely dependant on the output of the NER classifier. Taking this into account, the relation extractor is evaluated when using a manually created gold-standard output of the test data, and comparing that to the effectiveness when using it as part of our system. As a further matter, we compare the Naive Bayes classifier's performance to the baseline, in order to conclude which relation extraction classifier will be used to extract data from Wikipedia.

6.4.2 Feature Selection

Most Informative Features

This experiment judges the importance of each feature by looking at the 'most informative feature' function from the NLTK library. Table 6 below examines the usefulness of every feature after the classifier is tested.

Feature Name	Feature Output	Worded Ratio	Ratio
isObjFirst	'No'	NO_REL : SIZE_O	5.7 : 1.0
objToQuant	'close'	SIZE_O : NO_REL	3.7 : 1.0
isObjBetween	True	NO_REL : SIZE_O	3.1 : 1.0
quantsBetween	2	NO_REL : SIZE_O	2.4 : 1.0
objToQuant	'medium'	SIZE_O : NO_REL	1.9 : 1.0
quantsBetween	1	SIZE_O : NO_REL	1.6 : 1.0
quantsBetween	3	NO_REL : SIZE_O	1.6 : 1.0
quantsBetween	0	SIZE_O : NO_REL	1.6 : 1.0
isObjBetween	False	SIZE_O : NO_REL	1.5 : 1.0
isObjFirst	'Yes'	SIZE_O : NO_REL	1.4 : 1.0
objToQuant	'far'	NO_REL : SIZE_O	1.2 : 1.0

Table 6.7: Most Informative Features

From these results we can observe that if the object is not the first entity (of the pair) to appear in the sentence, there is a high likelihood that the object and size in question are not related. A close distance between the entities indicates that they are related, whereas an object or several quantities being between the entities in question prove useful features in negating their relationship, as one would expect. Interestingly, when there is one quantity between the entities the classifier infers that the entities in question are in fact related. This is because most sizes of objects typically appear twice in a sentence in imperial and metric units.

Using Different Inputs for the Classifier

In this experiment, it is observed how the system handles different inputs for the relation extractor. The gold-standard inputs have no noise and are perfect; whereas the NER input has some noise and inaccuracies (described earlier in this chapter). Table 7 and 8 show the results when running the system using the gold-standard chunks and NER chunks, respectively. As the most useful feature is clearly *isObjFirst* (seen from the previous experiment), this experiment also investigates how the removal of the other features affect the performance of the system. The features are described in chapter 4.3.

Features Removed	Precision	Recall	F1-Score
None	0.81	0.94	0.87
objToQuant	0.81	0.94	0.87
quantsBetween	0.92	0.98	0.95
isObjBetween	0.85	0.94	0.89
objToQuant; quantsBetween	0.86	0.98	0.91
objToQuant; quantsBetween; isObjBetween	0.86	1.00	0.92

Table 6.8: Comparing the combination of features using gold-standard chunks as the input

When it comes to using gold-standard chunks, the classifier runs admirably. The best results with the Naive Bayes come from the removal of the feature, *quantsBetween*. This provides the best F-measure (0.95) and highest precision (0.92), even if it does not have maximum recall like the baseline has. Although this score is very impressive, when using the output of the NER as the chunks to classify, the scores drop significantly, due to the inaccuracies of the NER described earlier in this chapter.

Features Removed	**Precision**	**Recall**	**F1-Score**
None	0.33	0.37	0.35
objToQuant	0.36	0.37	0.36
quantsBetween	0.46	0.33	0.38
isObjBetween	0.31	0.35	0.33
objToQuant; quantsBetween	0.52	0.31	0.38
objToQuant; quantsBetween; isObjBetween	0.38	0.41	0.39

Table 6.9: Comparing the combination of features using NER's output as the input

It can be seen that when the NER chunks are used as the input, removing several features prove to be beneficial. The removal of *objToQuant*; *quantsBetween* and *isObjBetween* (leaving only *isObjFirst*) gives the highest F-measure and recall. However, the highest precision is found by removing only *objToQuant* and *quantsBetween*. This precision score is significantly higher than the rest; making the classifier that only uses the other two features (*isObjFirst* and *isObjBetween*) the most effective classifier.

It must be noted that the 0.1 decrease in recall between the bottom two combinations in table 6.9 appears more significant than it actually is; the number of true positives (correctly identified relationships) only decreases by 6. This absolute change is rather small compared to the number of false positives decreasing from 33 to 14 (causing the high precision).

6.4.3 Comparison to Baseline

This experiment compares the baseline relation extractor to the Naive Bayes classifier that makes use of different features. The results for this experiment are in table 6.10 when gold-standard chunks are used, and 6.11 when the NER's output is used (as the input for the RE).

Type of RE	**Precision**	**Recall**	**F1-Score**
Baseline	0.78	1.00	0.88
Naive Bayes	0.92	0.98	0.95

Table 6.10: Comparing Baseline to Naive Bayes using gold-standard input chunks

It can be observed that both the Naive Bayes and baseline Relation Extractor performs highly. This is because there aren't typically many objects and sizes in a single sentence, meaning that there a low number of false positives (noise). The recall of the

baseline extractor does not overshadow the higher precision and F-measure of the Naive Bayes. It can therefore be concluded that the Naive Bayes beats the baseline extractor because of its ability to detect and ignore incorrect object-size relationships.

Type of RE	Precision	Recall	F1-Score
Baseline	0.20	0.43	0.27
Naive Bayes	0.52	0.31	0.38

Table 6.11: Comparing Baseline to Naive Bayes using NER's output as input chunks

The difference in performance is more noticeable when the input chunks to the RE are the output of the NER. This is due to the additional noise being passed in. The Naive Bayes classifier does well to ignore a lot of incorrect object-size relationships but the baseline classifier falls short by simply linking together every possible object-size pair. This is the cause of the disparity in precision and F-measure.

Although the baseline is obviously not an intricate rule-based system (having only one basic rule), the comparison still shows the Naive Bayes classifier's superiority, confirming the suggestions made from the literature analysis that a more sophisticated machine learning would be a superior approach for this classifier.

6.4.4 Results Sampling

Here is a table presenting some of the sentences that undergo classification from the program. The predicted output is shown alongside the actual output. Once again it is important to look at both the relation extraction using the gold-standard chunks and using the NER chunks, because the structure of the sentence in terms of the chunks change notably, causing the Naive Bayes classifier to act differently. It is observed that the classifier does a good job at classifying the basic sentences and a reasonable job with the more complicated sentences.

Three sentences below have been picked out from the RE results when using the NER chunks as the input. Each result follows the pattern of showing the input sentence; the input chunks (output of NER); the expected outputs of the RE lastly the output relationships after the RE processes the inputs. Green relationships indicate a true positive relationship while red relationships indicate a false positive. Any false negatives will be discussed beneath it.

Sentence 1

- **INPUT SENTENCE:** the 13 otter species range in adult size 0.6 to 1.8 m (2.0 to 5.9 ft) in length 1 to 45 kg (2.2 to 99.2 lb) in weight.
- **INPUT CHUNKS:**
  - Objects: ['otter', 'range', 'adult', 'weight']
  - Sizes: ['0.6 to 1.8 m', '2.0 to 5.9 ft']

- **EXPECTED RELATIONSHIPS:**
  - otter SIZE_OF 0.6 to 1.8 m
  - otter SIZE_OF 2.0 to 5.9 ft
- **ACTUAL RELATIONSHIPS FOUND:**
  - adult SIZE_OF 0.6 to 1.8 m
  - adult SIZE_OF 2.0 to 5.9 ft

No correct relationships were identified in this case. This is because the object 'adult' is between the object 'otter' and its sizes, causing the classifier to determine that 'adult' is more likely to be connected to the size.

Sentence 2

- **INPUT SENTENCE:** the leaves 3–10 cm (1.2–3.9 in) long, alternate, simple, serrated margin.
- **INPUT CHUNKS:**
  - Objects: ['leaves', 'alternate', 'simple']
  - Sizes: ['3–10 cm']
- **EXPECTED RELATIONSHIPS:**
  - leaves SIZE_OF 3–10 cm
  - leaves SIZE_OF 1.2–3.9 in
- **ACTUAL RELATIONSHIPS FOUND:**
  - leaves SIZE_OF 3–10 cm

The RE correctly identifies one of the two relationships in the sentence. It could not connect 'leaves' to '1.2-3.9 in' because that size was not identified in the NER process. The RE performs as well as it can here because it does not connect the noise objects ('alternate' and 'simple') to any sizes, thus minimising the noise in the final results.

Sentence 3

- **INPUT SENTENCE:** apple trees typically 4–12 m (13–39 ft) tall maturity, dense, twiggy crown.
- **INPUT CHUNKS:**
  - Objects: ['apple trees','crown']
  - Sizes: ['13–39 ft']
- **EXPECTED RELATIONSHIPS:**
  - apple trees SIZE_OF 13–39 ft

- apple trees SIZE_OF 4-12 m

- **ACTUAL RELATIONSHIPS FOUND:**
  - apple trees SIZE_OF 13–39 ft

Once again the RE does not classify the noise object 'crown' to a size, meaning there are no incorrect outputs. Due to the NER not identifying '4-12 m' as a size (shown by its absence in the input sizes), 'apple trees' could not be connected to it, meaning the RE could not link the two entities together. The RE works to perfection, given the inputs.

6.4.5 Scope for improvement

Judging from the scores achieved using the gold-standard input chunks, the best improvement that could be made to the relation classifier would be making improvements to the NER. This classifier appears to work much better when the inputs match exactly what's expected. Perhaps one could experiment with the classifier's training data to adjust to the NER noise, but this classifier would then become obsolete if improvements were made to the NER.

Particularly with species of fish or mountainous creatures, the distance at which they live above or below sea level is noted in many Wikipedia articles. This was unexpected from the design and the only way this classifier could combat this would be to use a feature that looked for key words or phrases such as 'live'; 'deep' and 'sea level'. This would require a large set of training sentences to learn from and then more in the test sentences to ensure its effectiveness. Generally, more sophisticated classifiers investigate the surrounding text in the sentence but this project did not have enough training data to find patterns of words in a sentence. This meant that it was not feasible to create a feature that looked at the text in the sentence, but the Naive Bayes classifier does a reasonable job regardless.

One could experiment with using alternative supervised-learning classifiers but it is arguably a waste of time when the Naive Bayes implemented here can achieve an F-measure as high as 0.95. Implementing co-reference resolution solutions would provide the ability to extract more data from web sources but it would be out of the scope of this project and would require a completely different classifier to the one that has been created to account for entities outside of the current sentence.

While this is certainly as much of an NER problem as it is a relation extraction problem, we observed from certain Wikipedia pages that some physical description information of objects are held in tables when there are several variants of that object. Due to the structure of HTML source code, each row is segmented within its own parentheses, meaning that once the NER locates the object and size within the code the RE could assume they are related. The strenuous part of this task would come when this is intertwined with the classifier; at the moment the classifier simply takes chunks and sentences as inputs but as HTML source code gets evaluated in this we would have to find a new method of passing this data in effectively.

6.4.6 Preferred Classifier for Web Scraping

When the program is web scraping from Wikipedia, there will be a focus on having the minimum false positives possible. Judging from our results we can conclude that the Naive Bayes classifier is a higher performer; retrieving only a couple fewer relevant relationships but eliminating the noise (false positives) by around half. This caused the decision to be made that the Naive Bayes classifier will be used in conjunction with the NER classifier to obtain information from Wikipedia with the maximum efficiency available.

Additionally, the Naive Bayes classifier will use the features *isObjFirst* and *isObjbetween*. This combination gave the highest precision from the experimentations (seen in Table 6.9).

6.5 Web Scraping from Wikipedia

6.5.1 Results

In total, around 129,000 results were obtained from under 20,000 web pages. Due to the database design, only two measurements were taken per object so only a fraction of these were stored. The extraction process took 28.5 hours.

It can be seen from the sqLite3 database that there is an overwhelming amount of noise and inaccuracy across the saved data. This is partly to do with the NER detecting incorrect names for objects; such as 'excavation'; 'service' and 'ranging'. In turn, this noise and misinformation probably would provide very little use to the end reader because, in a commonsense robot which would rely on a system like this, there is no guarantee the information they are looking for is correct and it could incorrectly change their decision making process.

Although it was arguably poor design to simply have two stores of the sizes for each object without checking them, the storage was not a primary focus of this task. The fundamental idea was that a perfect group of classifiers would only find the correct information meaning that the storage of every found size would only amount to one or two sizes per object.

Perhaps a useful scope for improvement with this part of the system would be to check against other entries for the same object to ensure that the current one being added fits the general estimation. This would eliminate noise from the database and we could generalise the storage format into an object entity having the properties 'mean size' and 'size range'.

6.5.2 Usability of User Interface

This experiment looks at the user interface and how easy or clear the database information is displayed.

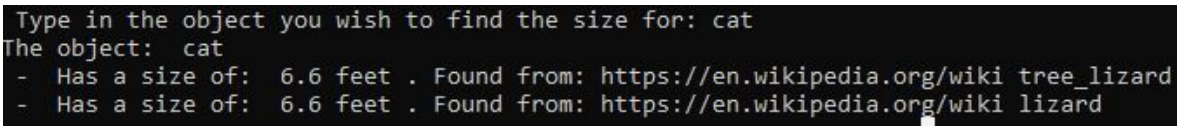

Figure 6.1: An example search from the user interface

The user interface simply consists of a command line. Searching for the object is easy as long as it matches exactly what is written in the gazetteer (there is no 'did you mean Y?' options), and what is returned is a list of measurements that were found for that word or phrase. There is an option to search for another object after displaying the results of the current search.

6.5.3 Results Sampling

The results sampling consists of two evaluations. Firstly, 25 random objects are picked out of the database and the results are examined to see if they are correct. This is done by going to the Wikipedia pages stored and observing if the object and size really are connected. The results are in the table below:

Total Relationships from Sample	Correct Relationships	Incorrect Relationships	% Object Sizes Correctly Stored
45	21	24	47

Table 6.12: A table showing how many database relationships were correct from 25 object entries

It can be seen that the system does a reasonable job on obtaining size measurements from the Wikipedia corpus. There are 45 out of a possible 50 entries to the database and this is because two sizes were not found on some of the rarer objects. The precision of these results is slightly worse than the test data evaluation (0.47 compared to 0.52) and even still the precision must be higher in order to allow any other software to rely on this database. It would be useful to sample a much larger size to see how it performs on a wide range of objects but unfortunately this is heavily time consuming because the data must be manually checked against the Wikipedia pages. However, from the sample made it would be fair to assume that this could not be used in any other real-world software yet due to its inaccuracies.

The next evaluation takes 5 of the objects from the 25 just tested, and looks at them in more detail to see exactly where the system falls short.

```
 Type in the object you wish to find the size for: apple tree
The object:  apple tree
 -  Has a size of:  8 to 15 metres . Found from: https://en.wikipedia.org/wiki/osage_orange
 -  Has a size of:  30–50 ft . Found from: https://en.wikipedia.org/wiki/osage_orange

 Type in the object you wish to find the size for: bird
The object:  bird
 -  Has a size of:  22 m . Found from: https://en.wikipedia.org/wiki/procellariiform_seabird
 -  Has a size of:  130 meter . Found from: https://en.wikipedia.org/wiki/casuist

 Type in the object you wish to find the size for: shark
The object:  shark
 -  Has a size of:  1.2–1.3 m . Found from: https://en.wikipedia.org/wiki/smalleye_hammerhead
 -  Has a size of:  5.5 m . Found from: https://en.wikipedia.org/wiki/requiem_shark

 Type in the object you wish to find the size for: cap
The object:  cap
 -  Has a size of:  35 cm . Found from: https://en.wikipedia.org/wiki/boletus_edulis
 -  Has a size of:  2.5 cm . Found from: https://en.wikipedia.org/wiki/boletus_mirabilis

 Type in the object you wish to find the size for: fish
The object:  fish
 -  Has a size of:  3 m . Found from: https://en.wikipedia.org/wiki/oarfish
 -  Has a size of:  600 m . Found from: https://en.wikipedia.org/wiki/albacore
```

Figure 6.2: Five search results from the database

From these results, inaccuracies can be identified when running the system across Wikipedia. The most interesting observations can be seen when comparing vague terms to more specific objects. Vague terms (such as 'bird' or 'fish') appear to have more inaccurate entries in the database because many more relationships are found for them across Wikipedia. This system doesn't make any data decisions to determine the most useful relationships for an object; it just stores any two that are found per object. Therefore, more vague terms are much more likely to have random, incorrect sizes.

When looking at 'fish' in figure 6.2, one of the sizes are accurate ('3 m', referring to a type of fish called the 'oarfish') while the other size ('300 m') is referring to how deep in the ocean they live. This contrasts to the more accurate results for 'apple tree' where both sizes are the height of the tree and are found from an apple tree page on Wikipedia. However, there is still room for improvement with this object. A more sophisticated system would recognise the size of the apple tree being referred to is the 'height' and store the type of size in the database to give more context to the end user.

Chapter 7

Conclusions

7.1 Key Findings

7.1.1 Named Entity Recogniser

The object gazetteer in the NER covers an extremely wide range of objects (at the size of 24,000 words or phrases), but this proved to be a notable source of noise as several words were mis-read as objects. Inaccuracies are caused by the NER taking words out of context. Additionally, some objects were not included at all.

Spacy has a mediocre natural language processor that is capable of tagging a wide range of sizes, but cannot detect anything measured using 'in' (for inches) as the unit. Additionally, it was inconsistent in the phrases that were chunked; sometimes including 'roughly' and other words, making the chunks difficult to evaluate just by looking at F-measure. It was not tested whether a rule-based size recogniser would improve the system's performance but it would be an insightful experiment to try in the future.

7.1.2 Relation Extractor

The performance of the Named Entity Recogniser directly effects the performance of the Relation Extractor. A simple relation extractor would only work if the chunks passed into it were perfect; even then there would be a significant amount of noise. Through using a Naive Bayes classifier as the relation extractor, the noise can be reduced significantly while maintaining a high precision, but ultimately the best immediate improvement to this system would be refining the NER.

7.1.3 System vs System Baseline

The Naive Bayes classifier consistently outperformed the rule-based baseline classifier, obtaining a higher F-measure with a significantly improved precision. Therefore, the Naive Bayes classifier should be favoured when using the system to determine the relationships between objects and their sizes. Perhaps a more intelligent rule-based classifier could improve the scores further than this baseline could, but it seems to be that machine learning would be the best approach due to the variety of writing styles across

Wikipedia's several thousand editors, in spite of the repetitive sentence structure of the sentences containing sizes.

7.1.4 Use of System on Wikipedia

Looking through a corpus as large as Wikipedia will find a significant amount of sizes for a wide range of objects, but some noise will be unavoidable. Sentences such as, "They are bottom-dwelling fish, living down to 200 m (660 ft)", will almost always be connecting an object to a size using a competent Relation Extractor. It is certain that the problems with the NER is the main cause of noise after looking at the sources of the relationships with the linked Wikipedia page.

7.1.5 Implications for Commonsense Reasoning

Although it is clear that the system would be extremely effective using using perfect chunks to pass into the relation extractor (obtaining an F-measure of 0.95), due to the lack of ability demonstrated by the NER it is unfortunate to conclude that this system cannot yet be used in the artificial intelligence industry. Several improvements must be made to the NER's approach to recognising objects and consistently finding sizes, in addition to looking at a better way of storing the correct sizes in a database. If our sole objective in this project was to only obtain correct sizes as easily as possible we would have used a web crawler that looks at the text provided by Google in the search results (figure A.1.). However, this project's aim was to investigate how effectively sizes can be obtained from a large corpus such as Wikipedia, meaning that this decision has certainly cost the system's reliability for the end user but it has provided some worthwhile findings with regards to information extraction using Wikipedia.

Unfortunately the database set up was not designed to deal with the overwhelming amount of sizes for each object. In addition to correcting the NER problems to reduce the number of relationships extracted, the system would further add to a commonsense machine by having the ability to differentiate between different measurements of the same object (for example, distinguishing the diameter of an object and the height).

7.2 Further Work

This is a summary of work that would not necessarily change the software we have created, but it would make a more refined and capable system overall.

7.2.1 Relative Sizing

A useful feature to add to this system would be the ability to detect the relative sizes of objects. For example, given the sentence 'Jupiter has more than 11 times the diameter of Earth', the system could infer that Jupiter is larger than Earth and specifically the extent to which it is larger, if it already knows the Earth's size. This would likely require another classifier to fully implement that would be able recognise comparison words such as 'larger' or 'smaller', in addition to the location of the objects relative to them. As

stated in chapter 3, this could provide an estimation to the sizes of objects in addition to a validity check on relationships found in the corpus.

7.2.2 Creating an Object and Size Corpus

In order to better train and test the relation extraction classifiers, a corpus could be made containing hundreds of labelled sentences. This would arguably provide a more in-depth and fairer overview of the relation extractors we compared, as we only looked at around 150 positive relationships.

7.2.3 Data Decisions

Storage of the measurements could be improved through the use of data decisions specified in chapter 3. Duplicate measurements found across the corpus and anomalous findings could be ignored, meaning that the final database for the end user is more reliable and accurate.

7.3 Final Thoughts

The system was able to determine the relationship between objects and sizes in a given sentence. Although the Relation Extractor was a highly effective classifier, the Named Entity Recogniser left a lot to be desired and ultimately let down the system when it came to extracting sizes of objects from Wikipedia. The primary faults with the NER included the inept object gazetteer that missed out some objects, in addition to the inconsistencies in size detection using Spacy's quantity recogniser. As such, we can can conclude that a strong foundation for this system is in place, but there is still some work to be done in this specific area of commonsense reasoning before it can be used by other artificial intelligence software.

Appendix A

Additional Figures

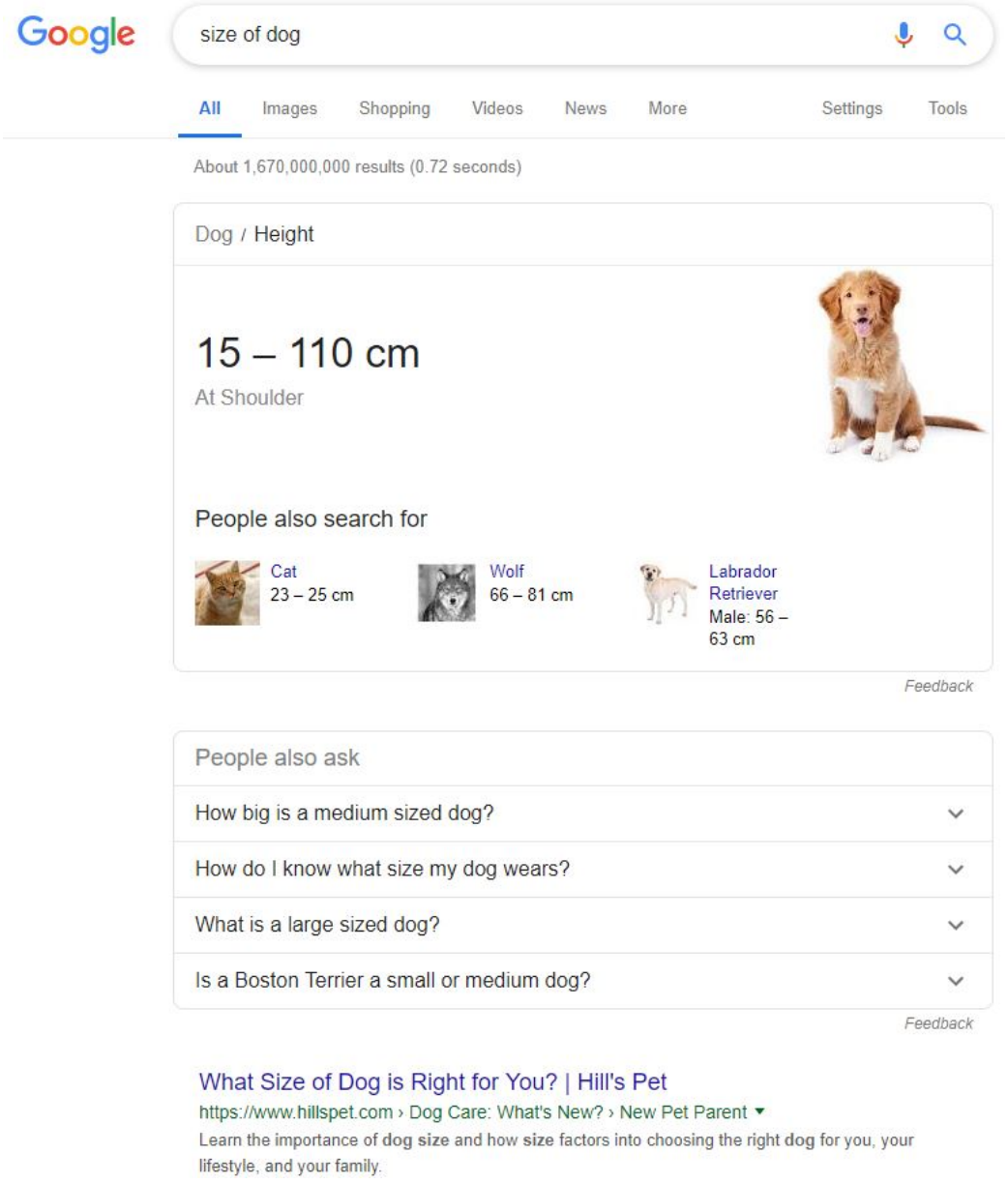

Figure A.1: An example of Google extracting the size automatically to show as the first result of a search

```
205 def preProcess(corpus):
206     ppS = []
207     stopWordsToRemove = []
208     for s in corpus:
209 
210         s = s.lower()
211         # print('LOWER CASE SENTENCE: \n', s)       #UNIT TEST: prints lowercased sentence
212 
213         nlpS = nlp(s)
214         for word in nlpS:
215             if (word.is_stop) :
216                 if word.text != 'in' and word.text != 'to':
217 
218                     newS = re.sub(r'\W%s\W' % ps.stem(word.text), ' ',s)
219                     s = newS
220         # print('STOPLISTED SENTENCE: \n', s)       #UNIT TEST: prints sentence with stop words removed
221         ppS.append(s)
222     return ppS
```

Figure A.2: Line 220 demonstrates the result of preprocessing

```
372             #check if window of words matches an object in gazetteer
373             foundObject = ObjectFinder(sentence, startSearchIndex, totalWords)
374             # If object is found:
375             if (foundObject != None    ):
376                 print('FOUND: ', foundObject, ' IN: ', sentence) #DEBUGGING: shows when an object is found
377                 object, skipIndexes = foundObject
378                 # print(object, skipIndexes)
379                 nlpSentence = TagObject(listOfWords, startSearchIndex, skipIndexes, object)
```

Figure A.3: Line 376 demonstrates the result of object chunking

```
693             # print('objToQuant output: given ',o, ' and ', q,'in: \n', s, '\n return : ', (objToQuant(''.join(s), o, q)), '\n')       #UNIT TEST: find
694             featureResults.update(objToQuant(''.join(s), o, q))
695 
696             # print('isObjBetween output: given ',o, ' and ', q,'in: \n', s, '\n return : ', (isObjBetween(''.join(s), o, q, set(objects))), '\n')
697             featureResults.update(isObjBetween(''.join(s), o, q, set(objects)))
698 
699             # print('quantsBetween output: given ',o, ' and ', q, 'in: \n', s,'\n return: ', (quantsBetween(''.join(s), o, q, set(quantities))), '\n')
700             featureResults.update(quantsBetween(''.join(s), o, q, set(quantities)))
701 
702             # print('isObjFirst output: given ',o, ' and ', q, 'in: \n', s,'\n return: ', (isObjFirst(''.join(s), o, q, objects)), '\n')       #UNIT TES
703             featureResults.update(isObjFirst(''.join(s), o, q, objects))
704             # print(o, classifier.classify(featureResults), q)
```

Figure A.4: Lines 693; 696; 699; 702 demonstrate the outputs of the each feature extraction process

Bibliography

[1] J. McCarthy, "Programs with common sense," in *Semantic Information Processing*, pp. 403–418, MIT Press, 1968.

[2] E. Davis and G. Marcus, "Commonsense reasoning and commonsense knowledge in artificial intelligence," *Commun. ACM*, vol. 58, pp. 92–103, Aug. 2015.

[3] A. K. H. Tung, *Rule-based Classification*, pp. 2459–2462. Boston, MA: Springer US, 2009.

[4] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine learning*, vol. 3, no. 4, pp. 261–283, 1989.

[5] J. Alty and G. Guida, "The use of rule-based system technology for the design of man-machine systems," *IFAC Proceedings Volumes*, vol. 18, no. 10, pp. 21–41, 1985.

[6] Wikipedia contributors, "Plagiarism — Wikipedia, the free encyclopedia," 2004. [Online; accessed 22-July-2004].

[7] F. Zheng and G. I. Webb, "A comparative study of semi-naive bayes methods in classification learning," *AUSDM05*, 2005.

[8] H. Shi and Y. Liu, "Naïve bayes vs. support vector machine: resilience to missing data," in *International Conference on Artificial Intelligence and Computational Intelligence*, pp. 680–687, Springer, 2011.

[9] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd ed)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

[10] "Wikipedia: Statistics," Nov 2018.

[11] R. Bunescu and M. Pasca, "Using encyclopedic knowledge for named entity disambiguation," in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, pp. 9–16, April 2006.

[12] P. Denning, J. Horning, D. Parnas, and L. Weinstein, "Wikipedia risks," *Commun. ACM*, vol. 48, pp. 152–152, Dec. 2005.

[13] U. C. Bureau, "Usa house median and average square feet."

[14] D. W. Homes, "Average house sizes in uk, https://www.dwh.co.uk/library/average-uk-house-sizes/," Jan 2018.

[15] J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, pp. 900–901, Dec. 2005.

[16] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "Brat: A web-based tool for nlp-assisted text annotation," in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, (Stroudsburg, PA, USA), pp. 102–107, Association for Computational Linguistics, 2012.

[17] P. M. Domingos, "A few useful things to know about machine learning.," *Commun. acm*, vol. 55, no. 10, pp. 78–87, 2012.

[18] K. K. Dobbin and R. M. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC medical genomics*, vol. 4, no. 1, p. 31, 2011.

[19] L. A. Ramshaw and M. P. Marcus, *Text Chunking Using Transformation-Based Learning*, pp. 157–176. Dordrecht: Springer Netherlands, 1999.

[20] J. M. J. Pustejovsky and M. Verhagen, "Iso-space: The annotation of spatial information in language," in *Proceedings of the Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, 2011.

[21] Wikipedia contributors, "Unit of length — Wikipedia, the free encyclopedia," 2019. [Online; accessed 23-April-2019].

[22] "Princeton University "About WordNet." WordNet. Princeton University.," 2010.

[23] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[24] E. Alfonseca and S. Manandhar, "Extending a lexical ontology by a combination of distributional semantics signatures," in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, (Berlin, Heidelberg), pp. 1–7, Springer-Verlag, 2002.

[25] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, 2017.

[26] F. Pan, R. Mulkar, and J. R. Hobbs, "Modeling and learning vague event durations for temporal reasoning," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pp. 1659–1662, 2007.

[27] B. L. Webber, "Description formation and discourse model synthesis," in *Proceedings of the 1978 Workshop on Theoretical Issues in Natural Language Processing*, TINLAP '78, (Stroudsburg, PA, USA), pp. 42–50, Association for Computational Linguistics, 1978.

[28] G. Hirst, *Anaphora in Natural Language Understanding: A Survey.* Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1981.

[29] "Wikipedia module, https://pypi.org/project/wikipedia/," 2019.

[30] "Ship sizes, http://maritime-connector.com/wiki/ship-sizes/," 2019.

[31] "Door sizes, https://www.cim-mappano.it/supplier/366.html," 2019.

[32] "Bear sizes, https://www.bearbiology.org/bear-species/american-black-bear/," 2019.

[33] "Calories info, https://listcaloriesinfo.blogspot.com/2018/08/apple$_6$.*html*," 2019.

[34] "Nature in greece — mountains, lakes, forests, fauna and flora, https://naturallyzagori.gr/crabapples-mountain-fruit-greece/," 2019.

[35] "Zappbug, https://www.zappbug.com/bed-bug-pictures/," 2019.

[36] "Seafriends.org.nz, http://www.seafriends.org.nz/enviro/fish/goatfish.htm," 2019.

[37] "Prepressure.com, https://www.prepressure.com/library/paper-size/din-a4," 2019.

[38] "Dogtime, https://dogtime.com/dog-breeds/greyhound," 2019.

[39] "Vetstreet, http://www.vetstreet.com/dogs/beagle," 2019.

[40] "Howlingpixel, https://howlingpixel.com/i-en/ice$_c$*ap*," 2019.

[41] "Universe today, https://www.universetoday.com/25756/surface-area-of-the-earth/," 2019.

[42] "Worldpopulationreview, http://worldpopulationreview.com/countries/countries-in-world-by-area/," 2019.

[43] V. Bochkarev, A. Shevlyakova, and V. Solovyev, "Average word length dynamics as indicator of cultural changes in society," *Social Evolution and History*, vol. 14, pp. 153–175, 08 2012.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.