University of Sheffield

Size Matters: Acquiring Vague Spatial Size Information from Textual Sources

Alexander White

Supervisor: Dr Robert Gaizauskas

A report submitted in partial fulfilment of the requirements
for the degree of Computer Science with a Year in Industry BSc in Computer Science

in the

Department of Computer Science

May 2, 2020

Declaration

All sentences or passages quoted in this document 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.

Name: Alexander White

Signature:

Date:

Abstract

Estimating the size of an object is a task that humans can perform easily but struggle to explain the justification of. It is common sense to be able to estimate the size of an object. This dissertation project aims to create a database of objects with their commonly found sizes to assist software in being able to make these same estimates. The project will achieve this by using information extraction techniques to find written examples of objects sizes. The final system will be using named entity recognition and relationship extraction to extract this information from a text source.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding an object's size can help machine learning models in image and video recognition by allowing estimation of an unknown object's size by comparing it with other objects in the image. For example, if you had to classify an object in an image as either a dog or a horse knowing that it was standing next to a person, and that on average a person is larger than a dog but smaller than a horse, this could inform your classification.

This dissertation takes on the problem of determining the general sizes of different objects. Humans are talented at estimating sizes of objects based on common sense or memory. As mentioned above this can help us make estimations about new objects, can help us to judge the distance of an object, or can help us to design accurate landscapes.

Named entity recognition techniques have been shown to produce accurate results, as have relationship extraction models. Combining this with gazetteers for objects or units of size is likely to give good results. The limitation to these models will be the quality and quantity of the training data. This is a trade-off, increasing the quality of our data takes more time and therefore reduces the quantity we can gather. Therefore, this project will go with a semi-supervised learning technique that tries to strike a good balance between the two.

1.1 Aims and Objectives

The aim of this dissertation is to make progress towards creating a database containing information about objects and their usual sizes. This can be broken down into three stages. Stage one is fulfilling our requirement of training data containing various objects and sizes. The aim is to train machine learning models to be able to identify objects and sizes within a sentence and determine if they are related. To collect enough data to adequately train these models we will use semi-supervised learning, which means that this stage will run simultaneously alongside stage two.

Stage two is building the named entity recognition and relationship extraction models for both identifying objects and sizes in text and determining if they are related. To help with the semi-supervised learning we can build some very basic models to start collecting data. Using regular expressions to determine sizes, and part-of-speech tagging to determine nouns, we can build models with poor accuracy but that will help in collecting data that can be refined into the final training set. This final training set will be used to train the models and

from there we can refine them to get the most accurate results.

The final stage of the project will be to collect all the results into a database. The accuracy of the collected data can be improved if objects have been found multiple times. We can look at previously found sizes of the object to determine if this new measurement is accurate. If we introduce an object hierarchy to determine if objects are related, then we can also use similar objects to estimate realistic sizes.

1.2 Overview of the Report

This report will begin with a literature survey, explaining the different options and previously tried techniques for various problems this project will face. It will give an overview of the techniques, its advantages, and disadvantages.

This will be followed by an in-depth investigation into the requirements of the project and an analysis of the problem and how the project will be tackling it. This will be similar to the literature survey except it will only discuss why a technique has been chosen over another with regards to the nature of the project.

This will be followed by an update on progress made so far and finally, there will be a conclusion and detailed project plan. This plan will include a gantt chart that accurately shows deadlines for different aspects of the project.

Chapter 2

Literature Survey

There has not been much previous work that has been done on this topic, but each of the individual aspects of the project have been researched in depth by other parties. The majority of the project is an information extraction task which has been widely researched with many techniques to perform well on various forms of data. Other considerations to be made will be what tools/programming languages to use, and how to store both the training data and the results.

2.1 Information Extraction

Something about active learning

The information extraction task can be broken down into two parts, named entity recognition and relationship extraction. This section will overview the problem, different techniques for the tasks, their advantages, and their disadvantages.

2.1.1 Named Entity Recognition

Named entity recognition is the process of identifying entities in text. This task can be broken down into finding the entities and classifying them into classes. The most common classes used for named entities are person names, organisations, and locations but you can build models to identify any class of entity. There are many challenges that come along with this, such as coreference, which is when an entity is referred to in the text but not by name. For example: "Henry Ford was born in 1863. He is known for founding the Ford Motor Company.". In the second sentence the entity "Henry Ford" is referred to as "He". This is called coreference and needs to be resolved before entities can be identified if you want to capture all occurrences of the entities.

Three primary approaches to named entity recognition are knowledge-engineering, supervised learning, and semi-supervised learning (Barrault 2019*a*):

Knowledge-Engineering

Knowledge-engineering approaches use gazetteers and human written rules to determine if a token is an entity. Its strengths are its high performance (on its specific domain) and its transparency. However, its weaknesses are that it requires domain experts to write the rules,

changing to another domain requires possibly rewriting all the rules, and you need domain specific gazetteers.

Supervised Learning

Supervised learning is a technique that attempts to fix the generalisation problem in knowledge-engineering approaches. Supervised learning systems learn from labelled examples, moving to a new domain only requires a new set of labelled examples. These labelled examples include features which inform the model as to the context in which the token was found. Features for a named entity recognition model would usually include information about the token, previous and future tokens, their part of speech tag, and any other information that might be useful in identifying a specific class of entity. There are a variety of different models that can be built using supervised learning such as decision trees, support vector machines, and neural networks. The advantages of this approach are that the model is easier to generalise towards different domains, depending on your problem the level of expertise to label the data is usually less than would be required to write rules, and you don't need any domain specific additional information such as gazetteers (Although these can help improve accuracy). Issues with this approach are usually that you need a large amount of annotated data for accurate results. Manually labelling that amount of data can take many hours, and in domains where the labelling might be subjective you would need multiple labellers to ensure high accuracy in your training data.

Semi-Supervised Learning

Semi-supervised learning is a similar approach to supervised, but the advantage is that you don't need as many labelled examples in your training data. You have a small amount of labelled data as part of a larger unlabelled training data set. Using the labelled examples and by looking at the structure of the unlabelled data you attempt to form a model, you are relying on assumptions that either points close to one another share a class, points within a cluster share the same class, or the classes can be inferred by patterns hidden on a lower dimension than the input space (Olivier Chapelle 2006).

2.1.2 Relationship Extraction

Relationship extraction is the process of determining if two or more entities in a text are related. Due to needing to know which tokens are entities this process can only be performed after named entity recognition or on a manually labelled set of text. An example of this relevant to the project would be the sentence: "The average apple is 7cm in diameter". The relationship extraction task is to determine if the two entities, in this case the object "apple" and the size "7cm", are related. There are techniques to tackle this problem with good results but there are some key challenges. Relationships over multiple sentences are much harder to detect and usually systems will ignore these relationships and only attempt to extract ones within a sentence when using a semi-supervised or bootstrapping approach. Semantic drift is another challenge the model will face. It describes the issue that arises when you use a small set of labelled training data for your model. If the model attempts to learn its own rules to generate more training data, an incorrect rule will generate more incorrect examples which will then generate more incorrect rules and repeat exponentially. This causes an exponential

drift away from the initial accuracy of the labelled examples. Techniques to combat this include human intervention. In the first few iterations where there are still relatively few examples, a human can manually check the results to see if they are accurate. However, this is not a perfect solution and semantic drift can be a significant factor in decreasing your model's accuracy.

Four primary approaches to relationship extraction are knowledge-engineering, supervised learning, bootstrapping and distant supervision (Barrault 2019*b*):

Knowledge-Engineering

Knowledge-Engineering approaches follow rules for identifying relationships. They can be split into two different categories: shallow, and deep. Shallow systems use pattern-action rules to determine if a relationship exists. For example:

> Pattern: "$Person, $Position of $Organisation"
> Action: add-relation(is-employed-by($Person, $Organisation))

Then the sentence "Mr. Wright, executive vice president of Merrill Lynch Canada" would match the pattern and the system would determine that "Mr. Wright" has a relationship to "Merrill Lynch Canada" of type "is-employed-by".

Deep systems use language rules to determine relationships. The means looking at examples and determining the grammatical relationships between the entities. For example, a subject, person, and an object, organisation, separated by a verb such as "works for" would indicate an is-employed-by relationship.

Advantages of this approach are that it has high precision and is transparent, i.e. a human can read the rules to understand why a relationship has been classified in the way that it was. However, the disadvantages usually outweigh this as it is impossible to write all rules to capture all instances and the approach would need new rules to be written for every different domain.

Supervised Learning

Supervised learning for relationship extraction is similar to that for named entity recognition, covered in section 2.1.1. The differences come when choosing the features for the training data. Due to already having the entities identified the features would include these classifications. They would also include the tokens between the two entities as this could be informative in classifying their relationship.

To reiterate, the strengths of this approach are its adaptability to new domains and the no requirement for writing complex rule sets. Disadvantages are that the model performance greatly relies on the quality and quantity of training data and the degree of which this data represents the real problem.

Bootstrapping

Bootstrapping can be thought of as a self-teaching model that starts by seeding it with a few examples. An example of this method is illustrated in Brin (1999). Given either some pattern examples or some relationship examples the model parses the training text to find

relationships that fit these patterns or patterns that fit the relationships given. From here it will build new patterns from relationships found or build new relationships from patterns found. It can do this iteratively until the rule set is deemed large enough. In the Brin (1999) example pairs of authors and their books are given as examples. These were then used to find patterns relating the authors to their books from 24 million web pages. These patterns were then used to generate new pairs of authors and books, and so on, until they had 15,257 unique books, all from starting with only 5.

The strengths of this approach are that it requires no labelled data, just a few examples, which it can very quickly generate more from. The disadvantage to this is semantic drift, if the model incorrectly matches two entities to a relationship it will generate a rule from this pairing. This rule will then allow the model to discover more incorrect relationships. Every iteration the number of incorrect rules will exponentially increase decreasing the accuracy of the model.

Distant Supervision

The aim of this approach is to reduce the need for labelled examples. You can think of this approach of being a mixture between bootstrapping and supervised learning. You perform one iteration of bootstrapping on a large set of training text and use this to build your supervised learning model.

The advantages of this is the reduced need for labelled training data and the speed at which the model learns new rules. The accuracy of these models is worse than that of supervised models (or very narrow domain knowledge-engineering models) but it requires much less data labelling.

2.1.3 Feature Selection

Machine learning models use features to inform themselves about the context in which the information has been found. For example, if the model was trying to classify an entity as a person then information such as if the word is in a dictionary, if the word is capitalised, the preceding and succeeding words, could all be informative in making the classification. Without features the model would have to see every example of a person entity in text to be able to classify them, essentially making it a fancy look up table. Features allow the model to identify new contexts as simlar to previous examples it has encountered and make a classification based on that similarity.

For the sake of computational efficiency and reducing the complexity of the model, the number of features should be kept low. This generally also increases the accuracy of the model by removing features that aren't informative or are even misleading (Hira & Gillies 2015).

2.2 Programming Languages

Python has recently become the go to language for data science and machine learning. Due to this there are a lot of libraries that have been built to assist programmers in these two domains. There are libraries for data manipulation and storage (numpy and pandas), libraries for natural language processing (NLTK), and libraries for building machine learning models

(sci-kit learn). Python also has the advantage of being easy to write and understand due to its simplified syntax, this in turn makes programming faster and more efficient.

However, due to Python being a relatively new language there are some advantages to using languages that have been around for longer. Another language that has some natural language processing roots is Java. One of the best natural language processing toolkits is StandfordNLP, a library for Java. Many academic projects investigating natural language processing techniques have been written with the assistance of this library and it has been proven to be a very powerful tool. Another advantage Java has over Python is that Java is a compiled language whereas Python is interpreted. This means that the run time of the Java system will be significantly faster once it has been compiled.

2.3 Datasets

One of the requirements for this project will be a large textual dataset that contains mentions of objects and their sizes. Other datasets that would be useful to the project would be any labelled sets that contain information on either objects, sizes, or both.

2.4 Databases

Once information about the objects has been identified it will need to be stored in a database. There are two main different types of databases, relational databases (SQL) and non-relational (NoSQL). SQL databases are used in situations where one item's relationship to another is important. They follow strict standards to ensure low data redundancy and high reliability. However, if the data you need to store does not require relationships between items then a NoSQL database is often faster and more adaptable.

2.5 Evaluation

A standard evaluation technique in machine learning projects is to use an F1 score.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

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

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

Precision is the measure of how many classifications made by the model are correct and recall is a measure of how many correct classifications are made by the model. F1 scores use a harmonic mean between precision and recall, meaning that if there is a large difference between the two, the score will be significantly skewed towards the lower. This prevents models from scoring too highly if their approach is play safe and label almost everything as the most common class.

Chapter 3

Requirements and analysis

The aims and objectives of this project (in chronological order) are as follows:

1. Build a data labelling tool
2. Find a suitable text data source
3. NER model v1
   (a) Build NER model v1 using POS, Wordnet, regular expressions to identify candidate sentences which can then be manually labelled
   (b) Label data found with NER model v1
   (c) Build NER model v2 to identify objects and sizes in text using machine learning and data found from NER model v1
4. NER Model v2
   (a) Build RE model v1 to determine if an object and size in text are related using machine learning
   (b) Build a feedback loop tool to relabel incorrect classifications
   (c) Correct NER model v2 and RE model v1 classifications and feedback into training data
5. Create database to store data
6. Use previous found instances of same objects to inform decision
7. Use object hierarchy to inform decision

Objective 4 marks the end goal of the project from a definition standpoint. Reaching this objective would mean that the project will have implemented the minimum requirements. The models might not be accurate, and the data might be poor, but a database of objects and their relative sizes will have been built using machine learning. Objectives 6 and 7 are additional steps that could help improve the accuracy of the model. These go alongside objectives 3.3 and 4.1, a model can quickly be built as a proof of concept initially, however

improving the features used and the quality and quantity of the data can drastically change the outcome of the project.

Objective 3.3 has a caveat, building a machine learning model to identify sizes might not be time efficient. Depending on the results of the regular expressions in objective 3.1 it might be more effective to stick with this approach and focus on the more difficult task of identifying objects. This is due to the simple nature of sizes, all sizes contain a unit, of which there are a limited number, and all sizes all contain a numeric value. Both these things are easily picked up by a regular expression.

There is also an extension to this project that would be to include sizes of objects relative to others. For example, "That tree is as tall as a house.", if you know the size of a house from previous text then you can infer that some trees are equal to this height.

The evaluation of the project's success will be difficult to measure. Testing the machine learning models using the labelled training data is an easy enough task. However, how the training data is gathered might skew our models. This means that although it might perform well on the test set, we need to ensure that the test set is an accurate representation of the real text we're trying to extract information from. We can limit the gap by trying to include examples of all classifications. Sentences with just objects, just sizes, both objects and sizes, sentences where the object and size are related and sentences where they are not. The means that the model will have an accurate representation of all different possibilities.

3.1 Machine Learning Models

Semi-supervised learning is the technique that this project will employ in order to train models. This is due to the limited time in which the project must be completed, as well as a lack of readily available datasets that would be helpful for this task. It is a good compromise between supervised and unsupervised learning.

Other approaches for information extraction such as knowledge-engineering could be a good fit for certain aspects of this project such as recognising sizes. Knowledge-engineering approaches approach the problem by using domain experts to write rules to capture the information. Rules for objects such as sizes could be as follows: The token must contain a unit of size (metres, m, ft, inches, etc), and the token must contain a numeric value (a scale of the size).

Choosing the features for information extraction to inform the model is a difficult task. Tools can be used to analyse training data to determine which features are providing the most information. Features with a high correlation can usually be reduced which helps to improve model size, run time, and accuracy. A starting set of features for named entity recognition can be found in Figure 3.1 below. The gazetteer in this feature set will contain different units of size to help identify size entities and also words classified as objects in WordNet. To adapt this feature set for relationship extraction we willl include the number of words between the two entities, the identity of both entities, and the order in which they appear in the text i.e. object then size or size then object. From this base we can use evaluate the models to see what features are most helpful in informing our classification.

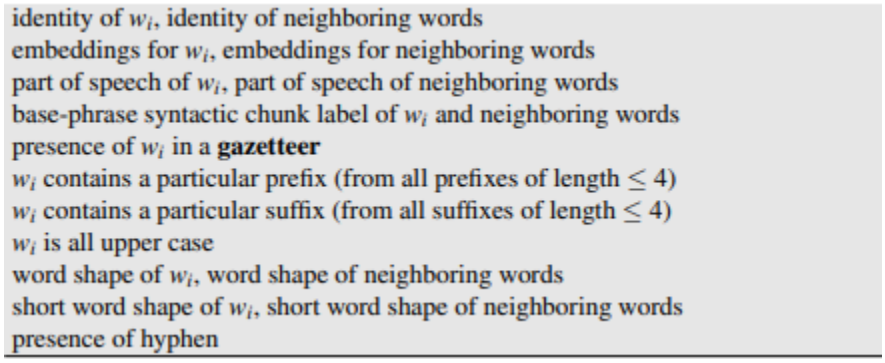

Figure 3.1: *Set of features found in Speech and Language Processing by Jurafsky & Martin (2019) pp. 4. for named entity recognition*

3.2 Programming Languages

Despite the faster run time and existence of the StanfordNLP library, for machine learning and data science Python has the advantage. There are countless libraries that will allow the implementation of this project to be much more efficient. Also despite Python running more slowly, once the models are trained the processing time of classifying a token or relationship is small enough that it will not be an issue.

3.3 Datasets

The most readily available dataset that is likely to contain information on object size would be Wikipedia. English Wikipedia can be downloaded into an xml file of just over 60GB. Due to the nature of Wikipedia and the type of information it contains it is a good candidate for our primary text dataset. As it is used for educational purposes, sentences describing objects should be more common than in other text datasets.

Another dataset that could be used to improve the performance of named entity recognition is Wordnet. Wordnet is a large lexical database of English and could be used as a gazetteer to identify entities that are objects. Wordnet will also allow the identification of multiple references to the same object as it lists alternative words for the same noun object.

3.4 Databases

The database for this project will be relatively simple and can be created using SQL. Database design guidelines explain the importance of normalising the tables to ensure that information is not stored multiple times. In the case of this project all we need to store is a reference to an object,a list of found sizes, and a link back to where each relationship was found, which is why we prefer this method over a NoSQL approach.

This is the database to store the results of our information extraction. We must also use a database to store the training data for the machine learning models. These will not need a relational database structure as all information can be contained in one row but due to the

results being stored in an SQL database it makes sense for all data to be stored in the same place.

3.5 Evaluation

Dividing the labelled dataset intended for training the models into a training and testing dataset will allow us to test our models on a dataset in which we know the correct classifications. Dividing up the dataset can be done randomly and repeated a number of times to generate and average performance for each model, allowing us to get a better idea of the general performance. The testing will test different types of model as well as how effective different features are in informing the models classifcations.

Chapter 4

Methodology

4.1 Design

Although the primary objective of the project is to generate a machine learning model that is able to detect related objects and sizes in text, there are steps that must be taken in order to get stage where we can generate and test such a model. Along with this the project also includes steps on improving the models performance by feeding corrected low confidence classifications back into the training data.

4.1.1 Model Design

The design of the models comes down to the features being used to inform the classifier and the classifier itself. The features inform the classifer as to the context of word/relationship that it has been asked to classify.

Features

Models

The models are implementations of models from the Python library scikit-learn (Pedregosa et al. 2011). Sci-kit learn provides a variety of models and by using the library we were able to implement different models and compare how they each performed on the test dataset. The models implemented are:

- Multinomial Naive Bayes
  Naive Bayes model for discrete features. Has been used in text classification and has been shown to have results comparing to those of SVMs (Rennie et al. 2003).
- Bernoulli Naive Bayes
  Naive Bayes model for discrete features very similar to multinomial naive bayes, however it differs in that it is designed for binary features. This model is being included as some of the features will be binary and this model could help inform the combined voting model.
- Logistic Regression
  Implements regularized logistic regression.

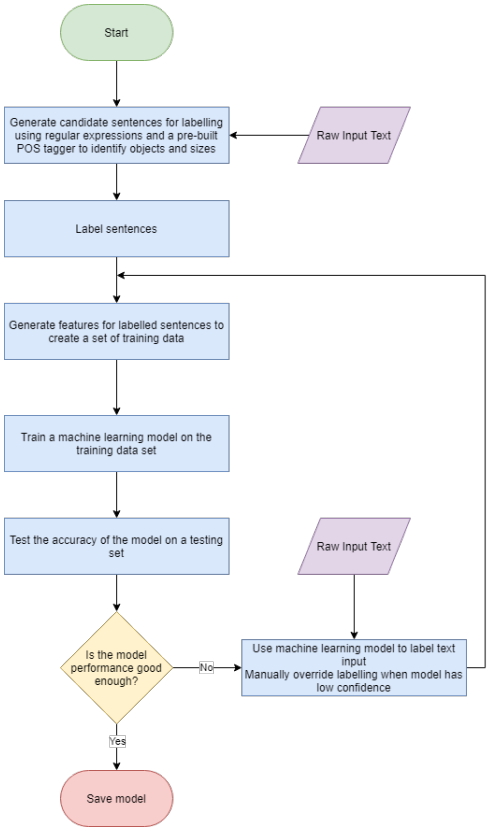

Figure 4.1: *Flow diagram to show the process of generating and iteratively improving the model*

- Stochastic Gradient Descent
  A standard stochastic gradient descent model, implements a regularised linear model with stochastic gradient descent.

- Support Vector
  A linear support vector that implements its linear classifier from libsvm (Chang & Lin 2011).

- Linear Support Vector
  A linear support vector that implements its linear classifier from liblinear (Fan et al. 2008) meaning that it is more suited for large datasets than the support vector model.

- Nu-Support Vector
  A variation on the support vector model that allows control over the number of support vectors. Will be used to test to see the effects that the number of support vectors have on the performance of the support vector models.

- Combined Voting
  The combined voting model will use a set of the above models to each classify the input, it will classify the input by majority vote. This aims to reduce the variance in the performance of the models. If there is an anomoly result that one of the models would have missclassified then the other models with override the vote and will correct the classification.

4.2 Implementation

4.2.1 Generating the training data

As outlined earlier there are a number of steps between the raw input data and the labelled features. One of the steps that could potentially be very time consuming when implementating a project of this size is labelling training data. To reduce the amount of time spent on labelling the data for this project I developed a labelling tool that allowed me to more quickly label entities and relationships. It works by displaying each candidate sentence one at a time, an input of "q" means that the sentence doesn't contain an entity or relationship, else you would either label the entities using the BIO scheme, or label the sentence as having a relationship, depending on what you are labelling the data for.

Having to process and label a large number of sentences is still very time consuming, so again to speed this up I created a basic "version 1" named entity recognition model. This model identified what I call "candidate sentences". These are sentences that should have a higher likelihood of containing entities. The version 1 model uses regular expressions to identify sentences that contain a unit of size in them. It also uses nltk's built in part-of-speech tagger to identify nouns along with Wordnet, although performance is practically identical whether you also use Wordnet or not.

4.2.2 Training the models

The labelled features are loaded in using pickle and then are split into a training and testing set. When testing the average performance of the models over different training data the

labelled features dataset will be randomly sorted before splitting. The split of the data is 90% for training and 10% for testing. The models are then successively trained on the training dataset and then combined into a voting model, before they are all tested.

4.2.3 Creating the database

The database consists of two tables: one for storing the entities and their average size from all references, and another for storing all references found in the text to the entities and their sizes. The second table is used to calculate the average size of each object from all the various references to them found in the text. The table to store the references also stores the sentence in which the relationship was found and this is helpful when later trying to analyse the performance of the models.

4.2.4 Extracting objects and sizes from text

Once the models have been generated they are saved as .pickle files. They can then be loaded into another Python script and used to classify incoming text inputs. The implentation of this part is very similar to the implementation of the version 1 model which was used to identify candidate sentences for labelling. The input text is broken down into sentences and run through the named entity recognition model, if both an object and size entity exist in the sentence then it is run through the relationship extraction model. If the entities are classified as being related the the database is searched to see if the object has been found before, if not the a new record is added to the objects table, else a new reference will be added in the references table and the average size updated.

4.3 Testing

Table 4.1: *Models performance on feature set 1 for named entity recognition*

Model	**Precision**	**Recall**	**F1**	**F1 Variance**
Multinomial Naive Bayes				
Bernoulli Naive Bayes				
Logistic Regression				
Stochastic Gradient Descent				
Support Vector				
Linear Support Vector				
Nu-Support Vector (0.3)				
Nu-Support Vector (0.5)				
Nu-Support Vector (0.7)				
Combined Voting				

Chapter 5

Results

Chapter 6

Discussion

Chapter 7

Conclusion

Bibliography

Barrault, L. (2019*a*), 'Information extraction: Named entity recognition', pp. 1–47.
URL: *https://vle.shef.ac.uk/ultra/courses/_77898_1/cl/outline*

Barrault, L. (2019*b*), 'Information extraction: Relation extraction', pp. 1–49.
URL: *https://vle.shef.ac.uk/ultra/courses/_77898_1/cl/outline*

Brin, S. (1999), 'Extracting patterns and relations from the world wide web', pp. 1–12.
URL: *http://ilpubs.stanford.edu:8090/421/1/1999-65.pdf*

Chang, C.-C. & Lin, C.-J. (2011), 'LIBSVM: A library for support vector machines', *ACM Transactions on Intelligent Systems and Technology* **2**, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. (2008), 'LIBLINEAR: A library for large linear classification', *Journal of Machine Learning Research* **9**, 1871–1874.

Hira, Z. M. & Gillies, D. F. (2015), 'A review of feature selection and feature extraction methods applied on microarray data'.
URL: *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4480804/*

Jurafsky, D. & Martin, J. H. (2019), 'Speech and language processing', pp. 2–18.
URL: *https://web.stanford.edu/ jurafsky/slp3/18.pdf*

Olivier Chapelle, Bernhard Schölkop, A. Z. (2006), 'Semi-supervised learning'.
URL: *http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf*

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

Rennie, J., Shih, L., Teevan, J. & Karger, D. (2003), 'Tackling the poor assumptions of naive bayes text classifers'.
URL: *http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf*

Appendices

Appendix A

An Appendix of Some Kind

Appendix B

Another Appendix