Temporal Ordering of Historical Events using Contextual Data

James Robert Friel

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2017

Abstract

Temporal ordering of events has typically been based on related information. In this project we investigate the effectiveness of external sources as an aid to temporal ordering of historical events. We discuss the ability to extract meaningful content from these external data sources and the relevancy of retrieval. The external source used is Wikipedia. The use of edge classification is explored as a means of deriving a temporal ordering of events. By optimising learning using a multilayer perceptron, we are able to see the impact of edge classification. The the use of global inference shows the usefulness of external data in temporal ordering against other baselines.

Acknowledgements

Firstly I would like to thank my supervisor Shay Cohen for his guidance throughout the project.

I would also like to thank the “School of Memeformatics” group for assisting as a brain trust throughout the project.

Thank you to Greggs Bakery for keeping me caffeinated and well fed.

I wish to also thank my family for their unyielding support throughout my life.

Lastly, thank you to Emily Wilson for her tireless proofreading and encouragement in times of need.

Table of Contents

Chapter 1

Introduction

1.1 The Problem

In this project we investigate the effectiveness of content from external data sources as aids to the temporal ordering of events. Temporal ordering is defined as the arrangement of events in time. The problem of ordering distinct events is not a new problem. There have been numerous papers published in recent years on automatic ordering of events. These papers have typically worked on a single topic domain such as cooking recipes or news articles. These related papers are discussed further in Chapter 2. We intended to build on this work by investigating if the reading of external data sources could aid in the temporal ordering of short historical news article headlines.

Our event data comes from the "Today in History" data set from Mydatamaster [1]. Each of these data points consists of an event and a date such as

$$[\text{“Alaska becomes 49th State”}, \text{“1959-01-01”}]$$

We tackled this problem through the use of summarising related articles and data of particular events and determining their feasibility as aids to temporal ordering.

1.2 Aims & Objectives

The aim of this project was to experiment with machine reading on external data sources to investigate the feasibility of extracting contextual data, from these sources, to aid in the temporal ordering of historical events.

We aimed to do this by experimenting with machine reading techniques to extract the most relevant contextual information from our data sources.

With this information, we intend to experiment with classification methods to optimise estimations of single edge ordering before generating a complete digraph from the resulting data.

Using the graph generated from our classifier, we attempted to find the optimum maximum spanning path through every node by experimenting with various path finding techniques. This path is the estimated ordering of history.

Using an evaluation metric, we discuss how the use of external data sources affected the system's ability to temporally order events. Our methodology for the experiments conducted are discussed in detail in Chapter 3.

1.3 Testing & Evaluation

With our data set of 6250 entries, it can easily be split into training, development and testing with no need for overlap. The data is split 80% training, 10% development and 10% for testing.

Evaluation of the system was completed using the Kendall rank correlation coefficient as it is a statistic used to measure the ordinal association between two measured quantities. This was applied to our results by comparing the system's estimated ordering of events with the label associated with the event from the data. The results of these experiments and discussion of their meaning can be found in Chapters 5 and 6 respectively.

1.4 Implementation

The experiments undertaken in this project consisted of Python and Bash scripts. The following Python packages were used:

- Scikit-learn [41] - Machine Learning tools for Python
- Numpy [40] - package for scientific computation
- Scipy [32] - Python library for scientific tools
- Wikipedia [24] - Python wrapper for integrating with Wikipedia's API
- NLTK [9] - NLP toolkit for Python
- Pool [39] - Multiprocessing utility for Python
- Pattern.en [15] - A web mining module
- Networkx [28] - Graph utility for Python
- Matplotlib [31] - Visualisation tools
- GenSim [42] - Topic modelling for Python

All classification and path-finding experiments were conducted using Python. A combination of functions were built into the aforementioned packages, along with algorithms and function designed and implemented specifically for the project.

Bash was used for automation of data sanitation and separating of the various data splits.

Stanford's suite of NLP tools were used to perform information extraction on our data to provide the object and relations of each event in order to gather article data from Wikipedia.

1.5 Contributions

While there has been a recent interest into temporal ordering of events, most papers discussing the topic focus on ordering by using data already attributed to each event. Within this project we investigated the usefulness of external data sources in aiding the ordering of short event texts. This was done through implementation of article retrieval and relevancy, optimising classification of edge categorisation along with construction and optimisation of various pathing algorithms based on known theories and existing work.

Chapter 2

Background & Related Work

In this chapter we will discuss the current state of the research field relating to our project. We will discuss in depth the topics of relation extraction and machine reading within a natural language understanding context. Lastly, we will evaluate the current research and studies that relate to the work undertaken in this project.

2.1 Relation Extraction

Dependency parsing is the core of relation extraction. By focusing on the dependencies between words, we can easily discover the relationships embedded within the text. While there are numerous types of dependency representations, relation extraction is focused around semantic dependencies [37]. Semantic dependencies are understood in terms of predicates and arguments.

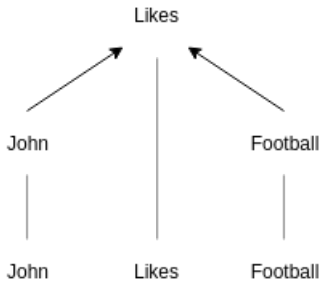

Figure 2.1: Example Semantic Dependency

In the example semantic dependency in Figure 2.1 the two arguments *John* and *Football* are dependent on the predicate *likes*. This semantic dependency is the basis for relation extraction.

The information extraction (IE) techniques used to gather arguments and predicates for relation extraction use a combination of well known natural language processing

(NLP) techniques. Given a raw source text, IE typically follows a standard pipeline for entity extraction [10], as seen in Figure 2.2. From the information gathered through IE, dependencies are build between the entities seen in the text

Language Identification	Match source text language to known language footprints
	⇓
Tokenization	Break input into a stream of word, phrases or other meaningful elements, called token
	⇓
Sentence Splitting	Split tokens into discrete sentences
	⇓
Part-Of-Speech (POS) Tagging	Assigns word tags, such as noun or verb, to each token
	⇓
Named Entity Recognition (NER)	Either through handwritten rules or machine-learned NER recognising objects, such as a name or an organisation from the construction of word tags

Figure 2.2: The Entity Extraction Pipeline

Historically, dependency parsing has consisted of only highly local models that extract each event and argument independently. Modern techniques are building upon this to build systems that can handle more complex arguments. These systems use a tree of event-argument relations represented in a re-ranking dependency [37]. This system captures both flat relations, such as in Figure 2.1, but also nested relations. An example nested relation can be seen in Figure 2.3. These nested relations allow for more context to be drawn from the source text.

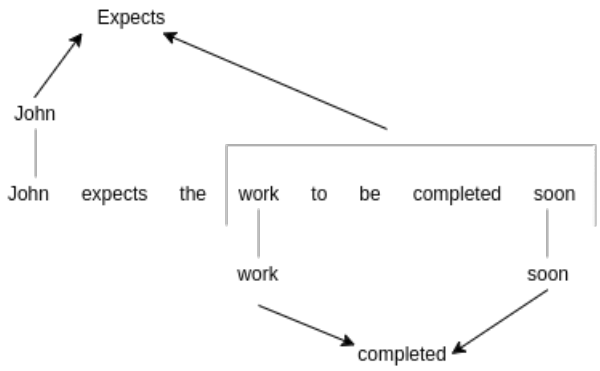

Figure 2.3: Example Nested Relation

2.2 Machine Reading

The core aim of this project was to investigate the usefulness of Wikipedia article content as an aid to temporal ordering of events. In order to perform this, we used machine reading on said articles. Machine reading is the core problem of natural language understanding (NLU). NLU deals with machine reading comprehension, the problem of dissecting natural language inputs into usable features, and is considered an AI-hard problem [54].

Reading comprehension is the ability to read, process and understand text. There are two levels of reading comprehension, shallow (low-level) and deep (high-level). Shallow comprehension involves structural and phonemic recognition along with the processing of sentence and word structure. Deep comprehension involves semantic processing, which happens when we encode the meaning of a word and relate it to similar words, this is fully discussed in Section 3.4.3. This levels system of comprehension was first proposed by Fergus I. M. Craik and Robert S. Lockhart [50]. The ability to read and process text for all level of comprehension is built upon the IE pipeline discussed in Section 2.1.

One of the earliest known attempts at NLU by a computer was the STUDENT system in 1964 [44]. Since then, NLU has become an umbrella term for a diverse range of problems and applications. These applications range from simple tasks, such as issuing simple commands to robots, to highly complex tasks such as full comprehension of poetry passages. Most problems within NLU fit between these extremes and will implement one of the types of comprehension depending on the task [11].

Reading comprehension within these tasks is important. Applications, such as these, need not only understand the literal content being passed to them, but also the contextual meaning within to better understand the nuances of the content. Machine reading is able to provide contextual information to provide better task performance when involving unstructured data.

Unstructured data is data that has no predefined data model or manner. This type of data is commonly seen in NLU with text mining and natural language rarely conforming to any cohesive model [19]. From the data sources typically used with NLU (news articles or free-form speech and text) the unstructured format of the data is beneficial as structured data typically has little contextual data [19].

In recent years, there has been a commercial interest in NLU, mostly focusing on news-gathering, text categorising and content-analysis. These systems vary vastly in terms of difficulty and use of comprehension. Some of the more advanced techniques used within these systems include applying logical inference to the framework. This is often done by mapping derived meaning into predicate logic, upon which logical deductions arrive at a conclusion.

2.3 Related Work

In recent years there has been an increasing interest in temporal ordering. A number of different papers have been published on machine learning approaches to temporal ordering of events. Although there have been various different approaches to this problem, the most common and successful approaches typically use unsupervised learning techniques. Typically, works often use pattern-based approaches and manually crafted rules [13].

Chambers and Jurafsky [12] addressed unsupervised learning of event relations. Their evaluation scenarios dealt with binary classification related to event ordering and aim to distinguish ordered sets of events from random permutations. Along with this, their work on event structure and semantic roles will prove useful to our work as external data sources can be riddled with synonyms and objects with similar attributes. Their research into argument representation and semantic roles were a beneficial base knowledge for our research into event extraction. This argumentation representation consists of robust, frame-specific roles about arguments. While frames are commonplace in NLP, the authors aimed to learn information about the world without predefined frames, roles and corpora. This unsupervised learning of roles and frames could be useful to our problem. Due to the nature of our data, structure and roles are not known prior to IE. This representation also has the potential to assist in a variety of NLP applications [12], which we intended to build upon.

Abend et al. [5] addressed the use of edge-factor models in temporal ordering. The classification task we aimed to undertake is similar to that of this paper but with a significantly larger set of events to order. Abend et al. [5] use a variation of the Hamiltonian path to complete an integer linear programming ordering of their events. Along with this, their baseline is that of a greedy inference algorithm. We looked at these two techniques as options for pathway estimation, but used a random assignment ordering as a baseline. We chose to look at a random baseline as opposed to using greedy inference as we wished to investigate the usefulness of greedy path-finding in comparison to an ILP approach.

Mani et al. [36] discussed the use of both hand written and lexical rules for event tagging. Their domain-independent approach to temporally anchoring and ordering events in the news is very accurate for both temporal anchoring and partial ordering of events. Anchor events are time expressions which are normalised via context, such as *Last Tuesday*. Although we did not build upon this paper's work with anchor events, it provides a good insight into the effectiveness of partial orderings. We performed a total ordering without the use of anchored events, as we perform unsupervised ordering of events.

While the methods discussed by Mani et al. [36] did not take into account dependencies between pairs of events, we see an alternative that does encompass these dependencies in [45]. Schapire et al. discussed methods using greedy algorithms to approximate optimum rankings of objects. It also discusses the use of ranking algorithms in the ordering of objects. The ranking methods discusses a weakly-connected digraph and greedy ranking of said graph. This relates to our problem, except we

deal with strongly-connected digraph and investigated both greedy and ILP methods of finding optimal solutions.

From the various paper discussed, temporal ordering of objects in any domain is of interest. The use of argument and relation extraction from unstructured data has been discussed in numerous papers, however none have yet succeeded in the domain of unstructured, non-framed data. There has been a significant improvement in recent years on the temporal ordering of events through the use of directed graphs. We can see this from the use of ranking algorithms on weakly-linked digraphs by Schapire et al. [45], compared to Abend et al.'s [5] use of edge-factor models in ordering predictions.

Chapter 3

Methodology

In order to evaluate the usefulness of an external data source, such as Wikipedia, we constructed a methodology for our experiments. We will discuss our choice of data set, extraction methods along with the reasoning behind the classifiers and pathing methods chosen.

3.1 Today In History Data set

The Today In History data set we worked with consisted of 6225 entries and is from Mydatamaster [1]. Each entry consisted of a brief event title and the date in the form "YYY-MM-DD". An example entry is

Alcatraz officially becomes a Federal Prison, 1934-01-01

The data is a collection of events and has an even distribution of events across the year (average 16.96 of events per day).

This data set was chosen as the events within are all objective historical events. This is beneficial as it minimised any ambiguity when selecting Wikipedia articles to process. This, coupled with date labels already associated with each event, made the data set favourable.

3.2 Choice of External Data Source

Wikipedia was chosen as the external data source for this project. This is due to the nature of the events in our data set, historical points of note, needing a source with a wide and deep range of articles. As Wikipedia has over 5 million English language articles [2] and our original data is taken from a Wikipedia outlet, we can assume that most events are contained within its pages.

Wikipedia is also written in an reasonably unbiased, procedural manner which allows more effective comprehension of events than that of raw, unstructured data. It is the key sentences within articles that we uses to estimate a timeline.

3.3 Text Retrieval

Using Wikipedia's article retrieval Application Programming Interface (API) [24], the task of gathering article content from keywords in its title was trivial.

However, choosing what information is contextually relevant to the subject was not so straightforward. Along with looking at how to extract the key entities and actions from our headlines, we also experimented with methods of extracting contextually relevant information from retrieved articles related to the event headline.

3.3.1 Information Extraction

2.1 Stanford's NLP suite, a natural language analysis toolkit, was selected to aid in extraction of features from the data set. Using this toolkit, the Open Information Extraction (OpenIE) from the University of Washington was used to extract subjects, objects and their relations from our event titles [46].

OpenIE implements a number of extraction techniques, such as noun-mediated extraction and N-ary extraction. Noun-mediated extraction is an extraction technique where the predicate between the subject and object is a noun [38]. This is in contrast to N-ary extraction which extracts N length relations using a combination of the simpler extraction techniques, including noun-mediated [6]. We have seen through experimentation with the various extraction techniques that no method provided usable relations for all inputs. Using OpenIE provided us with the ability to use multiple techniques and use the best fitting results.

This system provided better relationship extraction than other systems available. Systems such as ReVerb [17] or WOE [53] suffer from a few key weaknesses. These systems only provide verb mediated relation extraction [46]. This was problematic as sentences such as "US President Donald Trump" would not have the relation between *US* and *Donald Trump* extracted due to the structure of the relation lacking a verb. These systems are also not built to use context for relation extraction and thus extract non-factual relations [46]. It is for these reasons that ReVerb and WOE were not chosen as our information extraction technique.

Along with OpenIE's ability to seamlessly use multiple extraction techniques to find the best available relations for each event, it has numerous other properties that were beneficial to our project. OpenIE was also useful as it does not require a schema specified in advance. This is useful as we did not know the structure of the content that was being looked at. This allowed the system to extract more complex relationships, such as those seen in Figure 3.1, if such a relationship could be found.

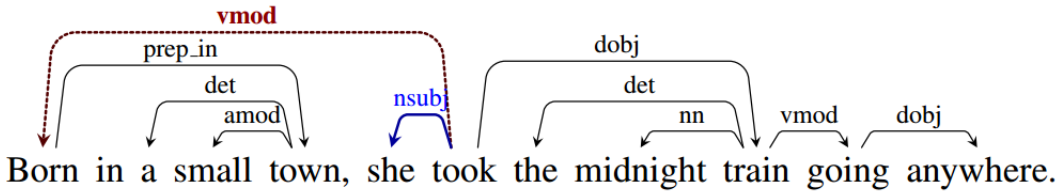

Figure 3.1: Example of a compound relationship [26]

OpenIE can also deal more complex relations than the simple relations. It can look at positive or negative assertion within these relations. This extra information may be beneficial to our solution as particular lenses are often applied to historical events. It was these properties that made OpenIE the optimal system to extract the key relations from our events.

3.3.2 Article Retrieval

Using the objects and subjects generated by OpenIE we retrieved the relevant Wikipedia articles. To do this we used fuzzy searching of Wikipedia, through their API, to get the articles relevant for these objects and subjects.

Several different methods of extracting key sentences were experimented with. These included:

1. Extraction of only sentences that had a date within them
2. Extraction of only sentences that had the other party in the relation within them
3. Extraction of only sentences that had a date and the other party within them
4. Extraction of paragraphs that referenced the other party
5. Extraction of sentences that contained the object or the subject
6. Extraction of sentences that contained the object or the subject referenced the action between the two

As shown in Table 3.1, experimentation yielded mixed results. The results showed a large range of average number of sentences retrieved and varying degrees of relevancy. Relevancy was measured manually using Cohen's Kappa (see Section 3.6.1) [52].

Given that we wished to categorise our retrieved sentences into either related to the event, or not related, we only had two possible categories to assign each of our N sentences to.

Method	Average # Sentences Retrieved	Relevancy
1	16	-0.875
2	2	-1
3	0.04	0.5
4	20	-0.4
5	27	-0.1$\dot{1}$
6	32	0.44$\dot{4}$

Table 3.1: Retrieval Methods and their results

From these results, method 6 was chosen as our context extraction method. While it did not return the most relevant results, extraction method 3 did not provide a sufficient number of sentences on average to build features.

Despite extraction method 6 not generating a perfect relevancy, under the time constraints of the project, it was decided to move forward with the project rather than to incorporate more complex solutions to improve relevancy.

3.3.3 Potential Issues With The Data Retrieved

While we have seen positive attributes of Wikipedia, there are certain issues that, if the content domain was to be altered, may cause an issue.

As Wikipedia is community managed, anybody can edit any page. This has been known to cause "edit wars" with users uploading biased information about subjects [29]. While in our current domain of long-term history, the facts and timelines are suitably coherent, if we were to apply this project to a much more current or hotly contested subject - such as the O.J. Simpson trial, or 2016 presidential election - then Wikipedia may not be the most appropriate source of information.

3.4 Feature Extraction

Having extracted a series of sentences for each event in our data, we required a concise method for representing these sentences. It is standard in NLP and machine learning domains to represent features as vectors. Several different techniques were discussed, but ultimately due to time constraints only two methods were experimented with: bag-of-words and word embedding.

3.4.1 Bag-Of-Words

The bag-of-words model is a simplifying representation of text. In this model, we take a sentence, or multiple sentences, and represent them as a bag (multiset) of its words. A multiset is similar to a set, but can allow multiple of the same item within. For

example, $\{\alpha, \beta\}$ and $\{\alpha, \beta, \alpha\}$ are the same set, but individual multisets. This technique disregards grammar and order, but keeps multiplicity. It is a common technique in computer vision and other learning situations [47].

For our project, we implemented bag-of-words by using the relevant sentences for both of our events as a feature. We chose to represent our multisets as counts over the vocabulary, so as to build numerical features to easily use the data with classifiers. An example of the bag-of-words model can be seen in Figure 3.2

Given the sentences

"John likes to watch movies. Mary likes movies too."
"John also likes to watch football games."

Based on these two text documents, a vocabulary list is constructed as follows:

["John","likes","to","watch","movies","also","football","games","Mary","too"]

From this list a vector is constructed of the occurrences of each word across the sentences

Result: [2,3,2,2,2,1,1,1,1,1]

Figure 3.2: An Example Bag-of-Words Usage

As bag-of-words only looks at a single word at time and the contextual data may be lost through this method, we looked at alternative methods to improve on this.

3.4.2 N-Grams

When experimenting with N-Grams, we tried to capture some of the meanings of phrases that appeared in the gathered sentences. This was achieved by looking for phrases and sequences of words that recurred throughout the sentences, under the assumption that what reoccurs must be important. Due to the wide variety in the structure of the data we were seeing, simple 2- or 3-Grams provided little to no use, returning junk phrases.

3.4.3 Word Embedding

Word Embedding was considered as this feature learning technique maps words or phrases to vectors of real numbers. Advances in this method from Google in the past

few years, such as the use of neural network architectures, have become popular as methods to perform natural language processing (NLP) such as syntax parsing [48].

Using Gensim's Doc2Vec models [42], we used this system to build thought vectors of our sentences. Thought vectors are vectors made of whole sentences or documents, typically used in machine translation [49]. It was decided to use thought vectors as they correlate words with labels rather than word with words, which is done by bag-of-words. This provided better representation of the words relating to ordering label, providing better results.

From our experimentation, we discovered that thought vectors produced an accuracy of 54% less than that when using bag-of-words. While this did not conform our hypothesis of using this technique, we believe the main reason that word embedding did not perform as we expected was due to imperfect sentence retrieval causing noise in the features. For this reason we did not move forward with word embedding beyond initial testing.

3.5 Classifiers

3.5.1 Decision Trees

Decision trees were considered as they are a non-parametric, supervised learning method that are commonly used for classification and regression. They are based on the idea of asking simple yes/no questions. The algorithm is set up to be a binary tree with each node being a question and each leaf a decision.

By inferring decision rules from data features, the system creates a model that predicts the value of a target variable.

These attributes were promising for our experiments as they fit with our data and generated classifications of test data along with probabilities of these estimations.

3.5.2 Support Vector Machines

Similarly to decision trees, support vector machines (SVMs) provide supervised learning models based on associated learning algorithms for classification and regression.

Given a set of labelled training data, an SVM will map this to a 3-D vector space. When an unlabelled data point is inputted into the system, the SVM fits the new data into the vector space and assigns it to the closest vectors class.

3.5.3 Logistic Regression

Logistic regression was also experimented with as it utilises a binary dependent variable - that being where it can only take one or two values, "1" or "0" in our case to

represent correct ordering.

The logistic regression model estimates the probability of a binary response based on one or more independent features. This is beneficial as along with providing classification estimates, the model can produce probabilities for its decision. These probabilities allowed us to use weighting techniques to improve our path building.

3.5.4 Perceptron

The perceptron is a common algorithm used for large-scale learning. It is a type of linear classifier that bases predictions on a linear predictor function that combines a set of optimised weights and the input feature vector. It is primarily used for supervised learning of binary classification. This fit well with our data as the perceptron decides whether an input vector corresponds to a specific class [20].

The perceptron model has several attributes that aided in improving the accuracy of our results. These include:

- It does not require a learning rate.
- It is not regularised (penalised).
- It updates its model only on mistakes

Perceptrons allow retraining of the model when newer data is added, this allows the perceptron to iteratively improve. An example of this improvement can be seen in Figure 3.3. It is these attributes that made the perceptron a suitable model to experiment with.

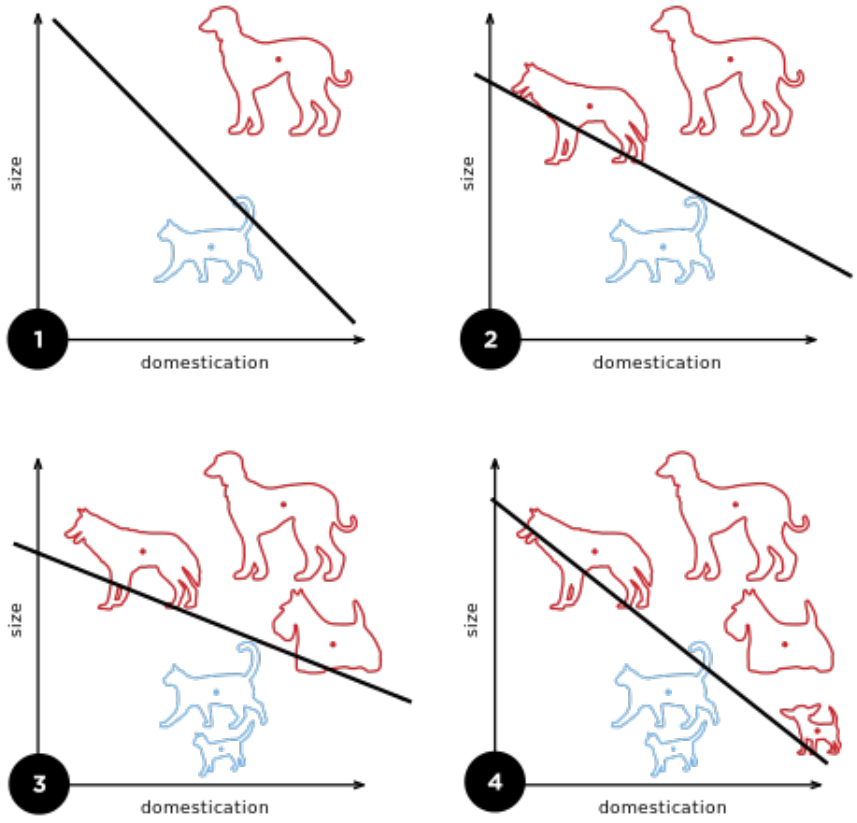

Figure 3.3: A diagram showing a perceptron updating its linear boundary as more training examples are added. [25]

3.5.5 Multilayer Perceptron

Having discussed the use of perceptrons, the next logical method to experiment with was multilayer perceptrons (MLP). A multilayer perceptron is an feed-forward artificial neural network that maps the set of inputs onto a set of appropriate outputs. In Figure 3.4 we see an example layer of an MLP. An MLP consists of layers of nodes in the form of a directed graph, with each layer connected to the next. Each node, except the input node, is a neuron (processing element) with an activation function. An activation function is a function that maps the weighted inputs of each processing element to an output. MLPs are trained using backpropagation.

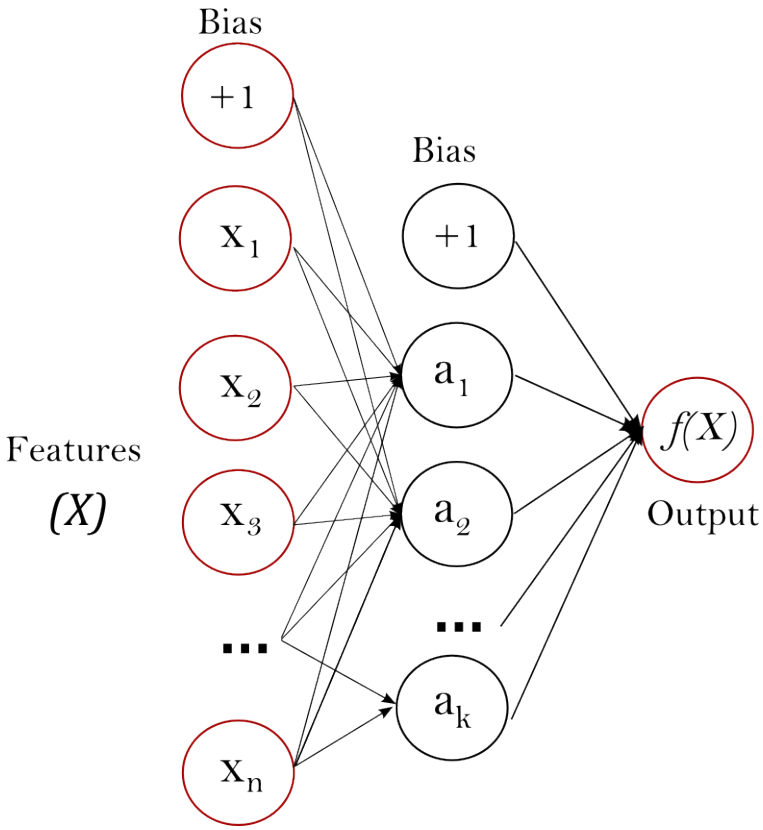

Figure 3.4: An Example one hidden layer MLP [41]

3.6 Evaluation

Throughout our experiments, there were several aspects that needed to be evaluated. The aspects that required measurement were:

- Retrieved Sentence Relevancy
- Accuracy of Classifier
- Correlation of Predicted and True Path

There are multiple different statistics that were considered for each of these problems.

3.6.1 Choice of Sentence Relevancy Metric

In order to select a method to extract relevant sentences from Wikipedia, a statistic that measured the agreement of relevancy was required. There are two main statistics for measuring inter-rater reliability: Cohen's Kappa and Fleiss' Kappa [27]. These two statistics measure the agreement between reviewers. The main difference between Cohen's and Fleiss' Kappa is that Cohen's Kappa only works for two reviewers, where as Fleiss' Kappa works for any number of fixed reviewers. Given the short time frame

of this project and availability of reviewers, it was decided to use Cohen's Kappa to find an agreement between two reviewers selection of relevant sentences gathered from Wikipedia.

Cohen's Kappa measures the inter-rater agreement between two raters who each classify N items into C mutually exclusive categories.

Cohen's Kappa is defined as:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

Where p_0 is the relative observed agreement between raters and p_e is the hypothetical probability of a chance agreement using the observed data to calculate the probabilities of each observer randomly assigning each category.

If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters, other than that expected by p_e, $\kappa \leq 0$.

Both statistics are thought to be a more robust measure than a simple percentage agreement. This is because it takes into account the possibility of agreement occurring by chance.

3.6.2 Choice Of Correlation Coefficient

There are various, different correlation coefficients that could have been used to evaluate our generated pathways. Some popular coefficients include:

- Pearsons Correlation Coefficient (Pearson's r) - a measure of the strength and direction of the linear relationship between two variables
- Intraclass Correlation - describes how strongly units in the same group resemble each other
- Kendall Rank Correlation Coefficient - measure of the portion of ranks that match between two data sets

For evaluation of our ordering, we decided to use the Kendall rank correlation coefficient (Tau coefficient). This statistic was chosen as it can be used to measure the ordinal association between two measured quantities. This allowed us to evaluate what proportion of our ordering was correct, giving a truer representation of accuracy compared to Pearson's r or Intraclass correlation [14]. Pearson's r was not chosen as it measures the linear correlation between two variables. This would not produce an accurate representation of the system as a single misaligned event would reduce the correlation drastically. Kendall's Tau was also preferred over Intraclass correlation as Intraclass correlation only quantifies the degree to which two variables are related. This was of no value to the correlation of our estimated path as this metric groups related items together.

The Tau coefficient is defined as

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{n(n-1)/2}$$

[4]

The Tau coefficient was used as it measures results in terms of agreement in the range $-1 \leq \tau \geq 1$. This allowed us to show the agreement of our orderings, with 1 being a perfect agreement of the ordering, -1 being a perfect disagreement of the ordering and 0 claiming total independence.

3.7 Summary

Throughout this section we have discussed the reasoning and methodology behind the project. The use of our chosen data set, Today In History, was due to its large spread of events across each day of the year. This should minimise bias to specific dates. The data also came with date labels and consists of events outwith recent history, which aids in external data retrieval.

Discussing the benefits and drawbacks of using Wikipedia as an external data source found that for our particular data set, historical events, bias in Wikipedia should not be of concern. However, it was decided that if the project was to move into more recent history then this bias would have to be discussed further.

Various, different article retrieval methods were discussed and tested. Scoring each method on retrieved sentence relevancy, the methodology chosen provided acceptable relevancy and a large enough corpus of sentences to be useful.

Also discussed were the advantages and drawbacks of the different feature extraction techniques and classification techniques used within the project.

The different evaluation techniques used throughout the project were discussed and evaluated. For the task of sentence relevant, Cohen's Kappa [52] was chosen as it measures the inter-rater agreement of reviewers. Kendall's Tau [4] was chosen as the metric for measuring the correlation of our estimated ordering against the true ordering of event. It was chosen as it measures the ordinal association between two measured quantities rather than the linear comparison.

Chapter 4

Experiments

For the methods discussed in Chapter 3 we required experimentation to optimise the classifiers. In this Chapter, we will discuss how and why certain choices were made in regards to classifiers. We will also discuss the pathfinding methods used to find the optimal path through the graph generated by the classifiers.

4.1 Classification

In order to investigate if retrieved sentences aided in the ordering of linear events we experimented with various machine learning techniques to learn how to order events by sentences. We aimed to determine if features generated from these important sentences provided a suitable training model to improve orderings against a random assignment baseline. We then compared these features to the use of titles alone to see if the use of retrieved sentences aids single edge ordering.

4.1.1 Approach

A labelling technique was established to encode our data and train the classifier.

Given that each event in our data set is of the form:

$$(t_i, d_i) \text{ for } i \in [M]$$

where t is the title, d is the associated date and M is the original data set,

Using the Wikipedia extraction techniques discussed in Section 3.3 we constructed a new data set:

$$\{(t_i, s_i, d_j)\} \text{ for } i, j \in [M]$$

Where s_i is the sentences retrieved from Wikipedia for title t_i. This formed the basis of our data to generate features.

Using these sentences, we built a new data set

$$\{(t_i, s_i, t_j, s_j, b_{ij})\} \text{ for } i, j \in [M]$$

where $b_{ij} = [y_i > y_j]$ indicates which event came first.
We used this value as the label for our training data and to evaluate our results against. A similar data set was constructed using only the article headlines.

From each of the entries in this new data set, we used the feature representation techniques discussed in Section 3.4 to build numerical features to train our classifier with.

Along with building a tuple event data set, we experimented with triple event entries.

$$\{(t_i, s_i, t_j, s_j, t_k, s_k, b_{ijk})\} \text{ for } i, j, k \in [M]$$

Where $b_{ijk} = [y_i > y_j > y_k]$ indicating if the ordering was correct for these events.

While we did not anticipate that increasing the dependency length of each entry would improve accuracy, it was investigated. We anticipated that using more events per feature would diminish the accuracy of the system as with a greater dependency tree, the availability of path transitions lessens [22].

As we expected, increasing the number of events associated with each training point greatly decreased the accuracy of results in all cases. Due to this, we will not focus on the use of triple sets of events, but instead focus on the use of the tuple data set.

We split our data set into 80% for training, 10% for development and 10% for testing. This breaks our 6226 entry data set into 4982, 622, 622 respectively. The allotment of the entries into each category was done randomly.

Experimentation was done using the development set to optimise the parameters to give the most optimal results, when finally tested on the testing data. For each classifier, we fed in the training features and their associated label. These features were either the features generated from titles alone or with the sentences retrieved from Wikipedia.

Once training of the model was completed, we fed in a new data point without a class label. The system returned a predicted class label for the new data and a probability score for its categorisation.

Each of these data points was similar to that of the training data

$$\{(t_i, s_i, t_j, s_j)\} \text{ for } i, j \, \varepsilon \, [M]$$

The classification returned for each data point was of the same style as that of the training data. Each of the classification methods discussed in Section 3.5 required experimentation to optimise the parameters. Once optimisation of the parameters had occurred, we saw the best results from our classifiers.

4.1.2 Decision Tree

As Decision Trees infer decision rules from the training data provided to it, there were few parameters to experiment with. With our system being a binary classification system, there was no need to offset the class weightings. From this we did not require any parameters to be used when training the classification tree.

4.1.3 Support Vector Model

SVMs have the option of various parameters to be tuned to optimise the results. In our implementation of an SVM we had the following parameters to experiment with:

- Penalty parameter of the error term
- The kernel type to be used in the algorithm

From experimentation with our development data, we found the optimal penalty parameter to be 1.0. This parameter indicates to the model how much you want to avoid classification of each example. Setting our penalty to 1.0 severely penalised the model for classification and lead to the model producing better results.

SVMs use a kernel to turn a linear model into a non-linear model. This is beneficial as it allowed the system to operate in high-dimensionality, implicit feature space without having to compute the coordinates of the feature in space [30]. From our experimentation the radial based function (RBF) kernel was chosen as it helps SVMs scale to large numbers of training samples which they do not typically perform well with.

The RBF kernel for two same vectors, x and $\tilde{x}$, is defined as:

$$K_{RBF}(x,\tilde{x}) = \exp[\frac{||x = \tilde{x}||^2}{2\sigma^2}]$$

[30] This kernel can project into infinite dimensions and therefore allows for easy computation of new coordinates, as we have a high number of training samples.

4.1.4 Logistic Regression

When using the logistic regression model, there were only a few parameters that were of interest:

- Type of penalty
- Type of solver

There are two main types of penalties that can be applied to logistic regression models, L1 and L2 loss functions. L1, also known as the least absolute deviations, is a loss function that minimises the sum of of the absolute differences (S) between the target

(Y_i) and the estimated values ($f(x_i)$) [35].

$$S = \sum_{i=1}^{N} |Y_i - f(x_i)|$$

L2 is also known as least square error. It minimises the sum of the square of differences (S) between the target (Y_i) and the estimated values ($f(x_i)$) [35].

$$S = \sum_{i=1}^{N} (Y_i - f(x_i))^2$$

L2 was chosen as it always provides a single, stable solution whereas L1 can return multiple solutions which is of no use for our system.

There are several different solvers that can be effectively used with a Logistic Regression model. For our system, we chose to use the LIBLINEAR solver as it supports L2, provides probabilities for Logistic Regression and is a linear classifier for large data sets [18].

4.1.5 Perceptron

During our experimentation with perceptron parameters, we discovered that giving the system nine epochs (passes over the training data) and shuffling the training data on each iteration provided results that maximised the estimations. For the same reasons stated in Section 4.1.4, we used an L2 penalty with our perceptron model.

4.1.6 Multilayer Perceptron

In order to effectively use an MLP for our data we experimented with various parameters.

- Number of hidden layers - the number of neurons per layer
- Type of activation function
- Type of solver
- L2 penalty
- Maximum number of iterations

After running experiments on small subsets of our data, we found the optimal number of hidden layers to be 100. This aids the model in transforming the inputs into appropriate information for the output layer.

Having looked at logistic and identity activation functions, the rectifier activation function was chosen. The rectifier activation function is the most popular activation function [34] and is defined as

$$f(x) = \max(0, x)$$

Where x is the output of a neuron.

The rectifier function was chosen as it is more effective than its available counterparts [23].

In order to solve the weight optimisation, we used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. This algorithm was chosen as it is an iterative model for non-linear optimisation. An L2 penalty of 0.00005 was chosen to aid our model beyond the training data. We chose the value of 0.00005 so as to prevent division by zero.

The last parameter we experimented with was the number of epochs. 100 layers were chosen as during experimentation with subsets of the data improvements were insignificant.

4.2 Path Finding

From our classifier, we constructed a directional ordering between every two events. Each of our events was represented by a node and placing each individual ordering on an edge, we constructed a digraph of the data. A digraph is an ordered pair G = (A,V) where [8]

- V is the set of nodes
- A is a set of ordered pairs of vertices

An example of such a graph can be seen in Figure 4.1.

It was with this new graph that we began to experiment with methods to construct the most probable path through every node in the graph.

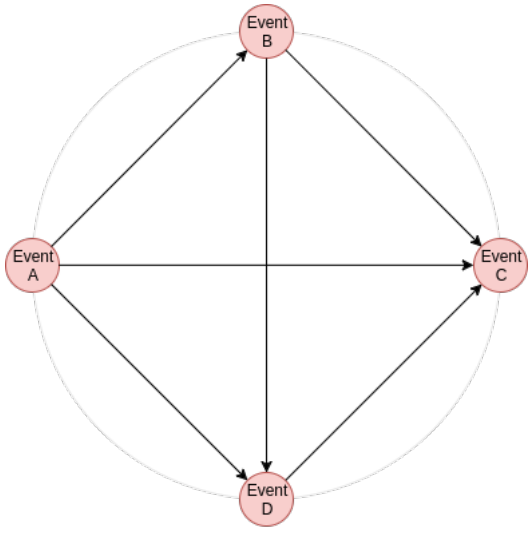

Figure 4.1: A small example of our results placed in a digraph

4.2.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is a NP-hard algorithm problem [33].It is defined as

"A travelling salesman wants to visit each of a set of towns exactly once, starting from and returning to his home town. One of his problems is to find the shortest such trip" [33]

While the TSP algorithm is still unsolved, we fortunately only had to solve a modified version of the problem.

"Given a graph with directed edges, find the optimal path from some starting node through all other nodes in the graph."

These restrictions on the graph made finding an optimal path through the graph simpler.

4.2.2 Pathing Algorithms

There have been numerous algorithms developed to find the shortest and longest paths through various types of graphs. As we required a total traversal of our graph, several of these algorithms were eliminated.

In order to accommodate many shortest-path algorithms, a new graph was constructed with edge weights between nodes being inverted, $\frac{1}{W}$ where W is the edge weight. This allowed paths that look for the cheapest path to be usable as they will traverse the true most expensive path, which encompass all nodes.

There are two main types of solutions to optimal pathing: greedy and integer linear programming. We will discuss the options available in each type of solution and the benefits therein.

4.2.3 Greedy Solution

A greedy algorithm is an algorithm that follows the problem solving heuristic of making locally optimal choices at each stage with the aim of finding a global optimum [16]. While a greedy strategy does not in general produce a global optimum, it may produce locally optimum solutions that approximate a globally optimum solution.

The greedy strategy for our variation of the travelling salesman problem was the following heuristic: "At each stage, visit an unvisited city nearest to the current city." Where nearest is defined as the lowest edge-weight.

The greedy algorithm used was beam search. It was chosen as it follows the greedy heuristic and also stores partial orderings, allowing us to prune solutions that do not cover all nodes. An example of this pruning ability can be seen in Figure 4.2.

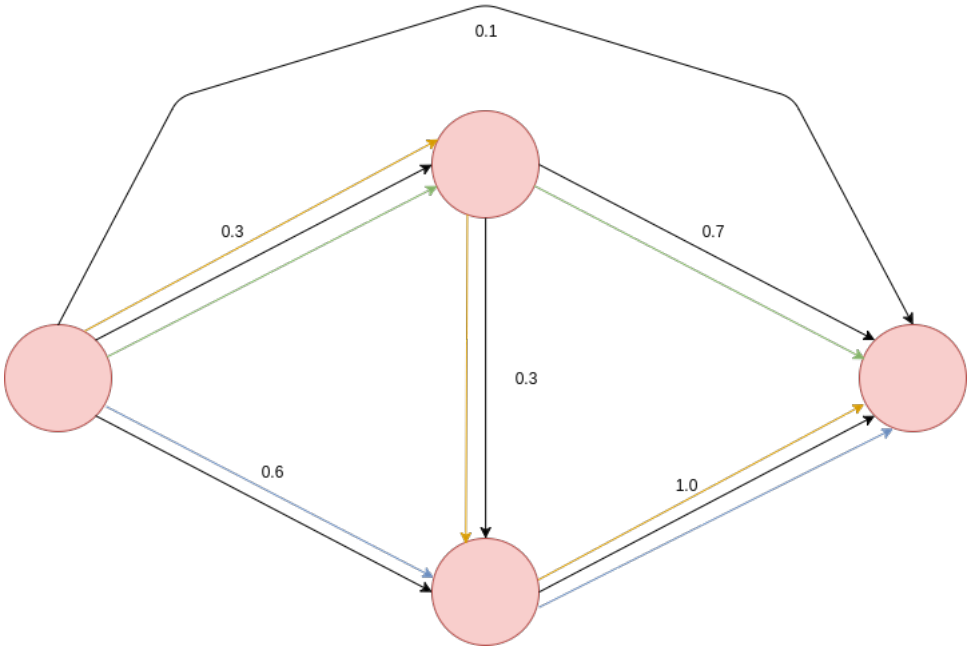

Figure 4.2: An Example of beam search pruning
Iteration 1: Blue Iteration 2: Green, Iteration 3: Yellow

As we did not know the optimal starting node, we executed our greedy solution from each of the event nodes. Due to every edge of graph being directional, it was possible that for any given starting node, that a solution for traversal of every node is impossible. To counteract this, and allow every node to generate a full path, a penalty system was implemented. This penalty system worked by allowing a path to deviate from any edges classified direction given that all other path options were exhausted. This deviation caused a large penalty to be applied to the path and typically rendered the paths start node unfeasible as a potential true starting node.

With this we generated the optimal greedy path through our graph. The results of which can be seen in Section 5.2.

4.2.4 Integer Linear Programming Solution

Integer linear programming (ILP) is a mathematical optimisation where the objective function and the constraints are linear. ILP is common in the field of optimisation and is ideal for our needs. ILP aims to find the globally most optimal solution using global inference. Global Inference produces the best global solution by inferring predictions from each input [43].When applied to path-finding ILP should generate the best global path through all nodes, rather than finding the next most optimal node like in a greedy solution.

The A$*$ search algorithm is a special case of the generalisation of branch and bound [7] and is derived from the primal-dual algorithm for ILP [55]. We chose this algorithm as it follows the ILP methodology. The algorithm works in a similar way to best-

first search, but takes into account the path cost already travelled. This was beneficial as it will give us the globally optimum path by finding the most probable total path, rather than the most probable next step when using a greedy algorithm. We can see an example of the complete digraph and the estimated path through it in Figure 4.3 and 4.4. Similar to the greedy solution, as we do not know the optimal start and end node, we had to run the algorithm with every possible start and end node combination.

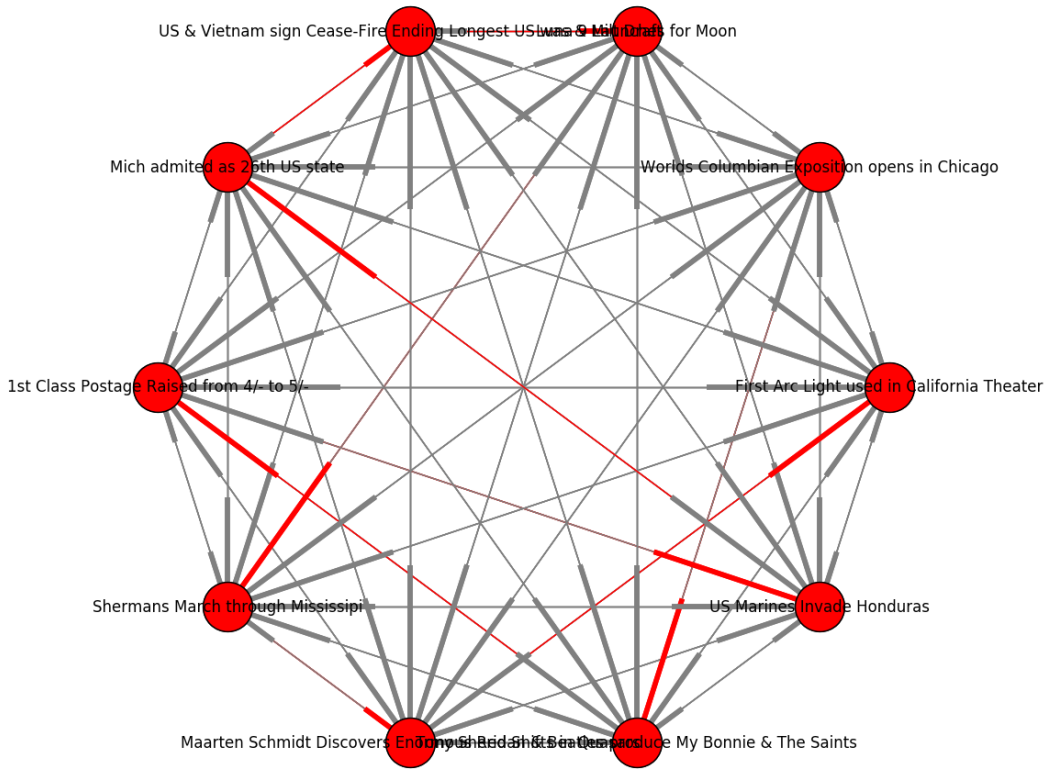

Figure 4.3: An example graph generated using 10 events, where each edge direction is indicated by a block head. Red edges indicate estimated path

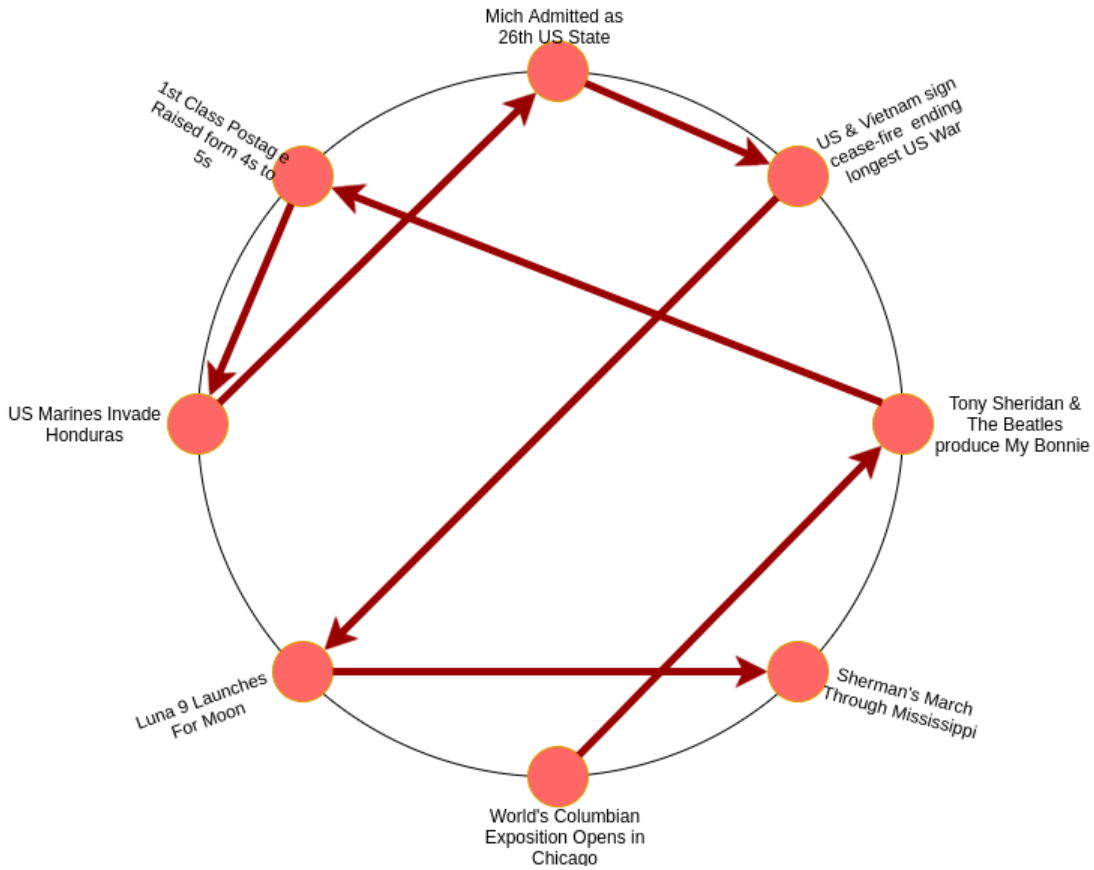

Figure 4.4: The example graph from Figure 4.3 with only the estimated path shown

4.3 Summary

In this chapter we have discussed the experiments carried out within this project. We also discussed the approach taken to adapting our data to allow classification of event pairs. This involved building a new data set in which each entry consisted of two events. This was done for each entry with every other entry, from which a complete graph can be constructed.

Also discussed was the experimentation with parameters for each classifier to maximise the results. This was done through testing of penalties, solvers and kernel type among others.

The various different path-finding methods for the graph generated from the data was also discussed. We looked at both ILP and greedy solutions to the problem.

Chapter 5

Results

In this chapter we report the results gathered from our experiments discussed previously. We look at both classification and path-finding results along with measurements about the data gathered. From our experiments, results were generated from both our classification and path-finding methods. Given the spread of results from our classification of tuples and triples of events, we used both of our path finding techniques in each of the resulting graphs. Our results include accuracies and correlations from experiments using both the sentences extracted from Wikipedia and just the event titles alone.

5.1 Classification

The results of edge classification between both two and three events can be seen in Tables 5.1 and 5.2 respectively. The results of these tables is measured by the proportion of true results, both positive and negative, among the total number of results.

The baseline used for classification is a random assignment. As each classification made is a binary value along a single edge between events, each classification has a 50% chance of correct classification because we do not apply class weights. Given that each classification of triple sets requires two edges (A $\rightarrow$ B $\rightarrow$ C) the probability of a correct classification is 0.25%. By applying this across the whole graph, the random assignment of edge classification yields an accuracy of

$$\frac{1}{4^N}$$

Where N is the number of nodes in the graph. Given that our results are based on 622 events, the baseline accuracy for triple set events is $\frac{1}{4^{622}}$.

The results generated using triple sets of events per classification report such poor accuracy with (mean = 19.4, standard deviation = 6.22896) or without articles (mean = 20.6, standard deviation = 8.61972). These poor results, regardless of use of article data, conformedf with our original hypothesis in Section 4.1.1.

The baseline for tuple of events is similar to that of the triples, but as each classification only requires one edge the random assignment accuracy is

$$\frac{1}{2^N}$$

Where N is the number of events. Given our 622 test events, the accuracy is $\frac{1}{2^{622}}$.

From our results generated using tuples of events, we can see that classification using only the event titles have a small spread of results (mean = 47, standard deviation = 15.09). Our experiments that used the retrieved sentences out-perform the results using only titles by large margins. These experiments are centred around 66 and have a much lower spread that with the use of titles alone (mean = $65.\dot{3}$, standard deviation = 12.1)

Accuracy	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	53%	66%	76%	66%	83%	$\frac{1}{2^{622}}\%$
With Titles	43%	51%	52%	46%	54%	$\frac{1}{2^{622}}\%$

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.1: Classification Results for Tuples

Accuracy	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	13%	13%	23%	27%	21%	$\frac{1}{4^{622}}\%$
With Titles	21%	13%	24%	33%	14%	$\frac{1}{4^{622}}\%$

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.2: Classification Results for Triples

5.2 Path Finding

As discussed in Section 3.6.2, our path finding results were measured using Kendall's Tau.

Our baseline for the generated paths is zero when using Kendall's Tau. This is due to when randomly ordering a set of data it will, on average, score a zero against the true ordering when using Kendall's Tau.

From the use of ILP path-finding method, we gathered the results for tuples of events in Table 5.3. As we can see, the use of articles greatly increases the correlation of estimated path with the true path. These results have a much smaller spread (mean = 0.50466 , standard deviation = 0.1756) than the ILP pathing of titles alone (mean = -0.1, standard deviation = 0.219).

A∗ Search	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	0.695	0.419	0.3	0.376	0.7333	0
With Titles	0.06	-0.09	0.048	-0.52	-0.28	0

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.3: ILP Pathing Results for Tuples

Using the greedy solution to path-finding, we found our results to have a much lower spread than the ILP results (mean = 0.46, standard deviation = 0.05) when using article data. Our results in Table 5.4 indicate that once again, using article data aids in the ordering correlations of estimated paths rather than when just using event tiles. The results from just event tiles shows no significant difference to that of the ILP results (mean = 0.0348, standard deviation= 0.3357).

Beam Search	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	0.42	0.5151	0.3	0.44	0.5454	0
With Titles	0.214	-0.62	0.17	0.09	0.32	0

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.4: Greedy Pathing Results for Tuples

With the use of triples of events, pathing in either greedy or ILP prove fruitless. The results seen in Table 5.6 confirm our hypothesis that longer event dependencies are detrimental to the accuracy of the system. The use of Articles (mean= -0.642, standard deviation= 0.16131) shows no significant difference from the use with just titles (mean= -0.674, standard deviation= 0.11459) with both results falling far below that of the tuples of events.

A∗ Search	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	-0.64	-0.54	-0.92	-0.53	-0.58	0
With Titles	-0.62	-0.68	-0.87	-0.61	-0.59	0

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.5: ILP Pathing Results for Triples

Beam Search	DT	SVM	LR	Perceptron	MLP	Baseline
With Articles	-0.81	-0.73	-0.96	-0.73	-0.69	0
With Titles	-0.81	-0.79	-0.9	-0.74	-0.68	0

DT = Decision Tree

LR = Logistic Regression

MLP = Multilayer Perceptron

Table 5.6: Greedy Pathing Results for Triples

Chapter 6

Discussion & Further Work

In this chapter we will discuss the results gathered and what this means in terms of our project goals. We will also discuss potential improvements that could be made to the system as a whole under further work.

6.1 Discussion

As we can see from Tables 5.2, 5.6 and 5.5 our original hypothesis, outlined in Section 4.1.1 involving the use of triples of events holds true. These results confirm that performing edge classification on three events, per edge, greatly reduces the accuracy of classification and subsequent path-finding. This is due to the dependency of each edge increasing with each node added and as the dependency increases the likelihood of total success lessens [22].

These results are eclipsed by the those seen in Tables 5.1, 5.3 and 5.4 with the use of tuples of events. Due to these results being unequivocally better than those using tuples of events, we will focus our discussion from here on the results generated by tuples off events.

As we see in Tables 5.3 and 5.4, the use of event titles alone provides a poor classification, centred around roughly 50% accuracy. This may be due to a number of reasons, but the key factor on these results is the lack of strong features to learn from. With each event title having an average of 8.24 words, the features generated are typically sparse. We believe this is the main reason that ordering using only title information is so poor.

The varying results of Table 5.1 show a wide spread of accuracies between the classifiers. Both with and without article data, the classifiers follow the same trends of accuracy

$$\text{MLP} \Rightarrow \text{Logistic Regression} \Rightarrow \text{SVM} \Rightarrow \text{Perceptron} \Rightarrow \text{Decision Tree}$$

With the orderings going from highest to lowest accuracy.

The results are interesting for several reasons. It is unusual for SVMs to perform better than Decision Tree Classifiers, in our situation this may be due to the chosen kernel for the SVM performing linear decision boundaries very well. Decision trees, while simple, may misidentify the important aspect of a feature, in retrospect it may have been useful to look at additive trees, such as MART [21], in order to improve upon these results. We also note that the use of a single percepetron is vastly worse than the use of a MLP, this is as expected as the properties of an MLP typically provide better results than that of a single perceptron. These results for the various classifiers used are as expected, with Logistic Regression and MLP performing the best at edge classification. Logistic Regression performed well due to its ability to deal with linear categorical data. While Logistic Regression performed well at the task, MLP overshadowed these results with it's significantly higher accuracy.

Overall, the ILP pathing solution outperformed the greedy solution. This was to be expected as greedy path finding can approximate an ILP under favourable conditions [45]. We can see from Table 5.3 that the correlation accuracies do not align perfectly with the classifier accuracies, notably with Logistic Regression moving down the rankings. We believe this reordering of most accurate classifiers is due to the pathing algorithms misaligning sections of the true ordering. An example of this being:

Generated Order : 1 2 3 6 7 8 4 5 9
True Order : 1 2 3 4 5 6 7 8 9
Where each number is an event

As we can see from this example, the generated order will place a small number of ordered events (6,7,8) earlier or later in the path. This is due to an incorrect edge direction with a high probability and relates back to the inaccuracies in our classifiers. It is this these sort of issue that cause the reordering of preferred methods when considering path optimisation.

The most accurate results come from the decision tree classifer and the multilayer perceptron classifier. Both of these methods, using an ILP solution, prove very strong result correlations. This shows that, despite there being room for improvement within the experiments, the use of external data sources does aid in the temporal ordering of historical events.

6.2 Further Work

While the work done in this project does confirm the validity of aiding temporal ordering through the use of contextual data, there are several aspects that could be improved upon and various methodologies that could be explored, given further time.

6.2.1 Improvement Of Relevant Sentence Retrieval

In Section 3.3 we discussed various possible sentence extraction techniques. While the relevant and number of sentences extracted was suitable for our purpose, there was certainly room for improvement.

In order to improve the relevancy of sentences extracted, we would suggest exploring the following avenues:

- Implementing Semantic Sentence Trimming, a method discussed by Zajic et al. [56] that uses linguistic-motivated heuristics to remove low content syntactic constituents
- The use of inter-sentence relevance instead of similarity of content may provide a potential improvement of overall sentence retrieval relevancy. We would suggest that this method be based upon the work of Zhong et al [57]

6.2.2 Potential Extension of Data Retrieval

As we discussed in Section 3.3.3, while Wikipedia has a large catalogue of articles to extract from, it does have several points that may be an issue if this project was to be taken further.

These issues include potential bias in information due to Wikipedia's open editing rules [51]. The current lack of cross-referencing is also a potential issue as cross-validation of timelines would further reinforce the knowledge acquired. It is with this in mind that we would suggest, if this project was to be changed to deal with more developing stories, that multiple sources of information be investigated with cross-referencing between these sources to build a stronger picture of the events.

6.2.3 Use Of Global Learning Methods

In this project we have explored methods involving single edge classification. Given more time, this project could be advanced through the use of techniques discussed by Abend et al. [5]. In this paper they discuss the application of global learning on temporal ordering problems. Global learning fits a distribution over the data. This methodology learns the parameters from observations of the data [3] and has the potential to make vast improvements on this work thus increasing the validity of external contextual data on temporal ordering. It is with this in mind that using global learning techniques on the problem would be strongly suggested in the future.

6.2.4 Comparison of Path-finding Techniques

In this project, we considered both ILP and greedy solutions to optimal path-finding through the generated graphs. The implementations of algorithms were not optimal.

Looking further into both ILP and greedy solutions would be recommended as while ILP typically outperforms Greedy, the runtime of our ILP solution is more computationally expensive. Greedy algorithms have been shown to provide approximations that come close to ILP solutions under the right conditions, with much lower cost, as shown in Schapire's paper [45]. With this in mind, such a solution should be investigated for our problem. If this method can be applied to the problem of temporally ordering events, then the scale of events considered could be greatly increased with an improved runtime.

Chapter 7

Conclusion

As we have seen throughout our experiments, the usefulness of external data in aiding the temporal ordering of historical events is valid. We have seen that the use of this external content can improve accuracies by up to 30% over the use of event titles alone. Article retrieval and relevant sentence extraction proved to substantially aid in the temporal ordering of events. The methods discussed, in combination with experimentation with classification models, performed substantially better than using data from event titles only. The various classification methods used show this trend with varying success. Using a multilayer perceptron provided an improved accuracy of 30% with article usage over titles alone, boasting 83% accuracy compared to 54% when using titles alone.

These results are not the ceiling for this subject area, methods and approaches discussed in this paper have scope to be improved upon. The use of Wikipedia as an external source of content has been proven as a valid source for long term historical information, but may pose issues if applied to recent history. Overall, we have shown that the use of external data sources, such as Wikipedia, do aid in the temporal ordering of historical events and provide a promising base for future research into this domain.

Bibliography

[1] Today in history dataset. http://mydatamaster.com/. Accessed: 13-09-2016.

[2] Wikipedia: Size comparison. https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons. Accessed: 27-02-2017.

[3] *Global Learning vs. Local Learning*, pages 13–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] Herve Abdi. The kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, pages 508–510, 2007.

[5] Omri Abend, Shay B Cohen, and Mark Steedman. Lexical event ordering with an edge-factored model. In *Proceedings of NAACL*, 2015.

[6] Alan Akbik and Alexander Löser. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. Association for Computational Linguistics, 2012.

[7] Egon Balas and Paolo Toth. Branch and bound methods for the traveling salesman problem. Technical report, DTIC Document, 1983.

[8] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[9] Edward Loper Bird, Steven and Ewan Klein. Natural language processing with python, 209.

[10] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. Twitie: An open-source information extraction pipeline for microblog text. In *RANLP*, pages 83–90, 2013.

[11] Lawrence A Bookman. *Trajectories through knowledge space: A dynamic framework for machine comprehension*, volume 286. Springer Science & Business Media, 2012.

[12] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 602–610. Association for Computational Linguistics, 2009.

[13] T Chklovski and Pantel PVerbocean. mining the web for fine-grained semantic verb relations. In *Conference on empirical methods in natural language processing*, 2004.

[14] Nian Shong Chok. *Pearson's versus Spearman's and Kendall's correlation coefficients for continuous data*. PhD thesis, University of Pittsburgh, 2010.

[15] W. De Smedt, T. & Daelemans. Pattern for python.

[16] Ronald A DeVore and Vladimir N Temlyakov. Some remarks on greedy algorithms. *Advances in computational Mathematics*, 5(1):173–187, 1996.

[17] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[18] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[19] Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

[20] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[21] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[22] Kim Gerdes, Eva Hajičová, and Leo Wanner. *Computational Dependency Theory*, volume 258. IOS Press, 2013.

[23] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[24] John Goldsmith. Python wikipedia api wrapper. `https://github.com/goldsmith/Wikipedia`. Accessed: 25-03-2017.

[25] Elizabeth Goodspeed. *A diagram showing a perceptron updating its linear boundary as more training examples are added.* May 2015.

[26] Stanford NLP Group. Stanford openie documentation. `https://nlp.stanford.edu/software/openie.html`. Accessed: 26-03-2017.

[27] Kilem L Gwet. *Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters*. Advanced Analytics, LLC, 2014.

[28] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[29] Brent Hecht and Darren Gergle. Measuring self-focus bias in community-maintained knowledge repositories. In *Proceedings of the fourth international conference on Communities and technologies*, pages 11–20. ACM, 2009.

[30] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[31] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[32] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. `http://www.scipy.org/`, 2001–. [Online; accessed ¡today¿].

[33] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.

[34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[35] log0. Differences between l2 and l2 as loss functions and regularization. `http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/`. Accessed: 26-03-2017.

[36] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 753–760. Association for Computational Linguistics, 2006.

[37] David McClosky, Mihai Surdeanu, and Christopher D Manning. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1626–1635. Association for Computational Linguistics, 2011.

[38] Seyed Iman Mirrezaei, Bruno Martins, and Isabel F Cruz. The triplex approach for recognizing semantic relations from noun phrases, appositions, and adjectives. In *European Semantic Web Conference*, pages 230–243. Springer, 2015.

[39] T. Sullivan A. Fang M.A.G. Aivazis M.M. McKerns, L. Strand. Building a framework for predictive science.

[40] Travis Oliphant. Python for scientific computing, 2007.

[41] 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*, 12:2825–2830, 2011.

[42] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges*

for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[43] Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document, 2004.

[44] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:19, 1995.

[45] William W Cohen Robert E Schapire and Yoram Singer. Learning to order things. *Advances in Neural Information Processing Systems*, 10(451):24, 1998.

[46] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics, 2012.

[47] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):591–606, 2009.

[48] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465, 2013.

[49] Deeplearning4j Development Team. Thought vectors, deep learning & the future of ai. https://deeplearning4j.org/thoughtvectors#thought-vectors-deep-learning--the-future-of-ai. Accessed: 16-03-2017.

[50] Richard K Wagner, Christopher Schatschneider, and Caroline Phythian-Sence. *Beyond decoding: The behavioral and biological foundations of reading comprehension*. Guilford Press, 2009.

[51] Matthew Wall. Wikipedia editing rules in a nutshell. http://www.bbc.co.uk/news/technology-32412121. Accessed: 30-03-2017.

[52] James M Wood. Understanding and computing cohens kappa: A tutorial. *WebPsychEmpiricist. Web Journal at http://wpe. info/*, 2007.

[53] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics, 2010.

[54] Roman V Yampolskiy. Ai-complete, ai-hard, or ai-easy-classification of problems in ai. In *MAICS*, pages 94–101, 2012.

[55] Xugang Ye, Shih-Ping Han, and Anhua Lin. a note on the connection between the primal-dual and the a* algorithm. In *Innovations in Information Systems for Business Functionality and Operations Management*, pages 170–181. IGI Global, 2012.

[56] David Zajic, Bonnie Dorr, Richard Schwartz, Christof Monz, and J Lin. A sentence-trimming approach to multi-document summarization. In *Proceedings of HLT/EMNLP 2005 Workshop on Text Summarization (HLT/EMNLP05)*, pages 151–158, 2005.

[57] Maosheng Zhong, Yi Hu, Lei Liu, and Ruzhan Lu. A practical approach for relevance measure of inter-sentence. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 4, pages 140–144. IEEE, 2008.