Event Ordering Of News Articles
Interim Report

James Friel
S1332298

January 27, 2017

Contents

Chapter 1

Introduction

1.1 The Problem

Nominal data is descriptive in nature, making is difficult to assign an canonical ordering to.

The problem tackled in this dissertation is the ordering of news article headlines to generate a most-probable traversal of a weighted directed graph of these events.

The problem originally proposed for this dissertation was the ordering of news article headlines to generate a most-probable traversal of a weighted directed graph of these events. The project has been modified from this original proposal to use fetched article data to attempt to derive context to improve ordering estimation.

This problem is inspired by the paper "Lexical Event Ordering with an Edge-Factored Model" [3] and some techniques discussed and used henceforth are based off of this paper.

Our data comes from the Wikipedia "Today in History" data set and the article data is retrieved from Wikipedia articles.

1.2 Aims & Objectives

The aim of this project is to construct a system that predicts the most probable path through an edge-weighted directional graph of events. We aim to construct this graph by extracting data from Wikipedia for each event and build a date estimate from Wikipedia. From this data we will conduct several experiments to find the most likely spanning path across the generated graphs.

The core aims of the project were:

- Investigate the usefulness of Wikipedia text in feature generation
- Generate probable dates For each event
- Order these events linearly

The further goals of the project were to:

- Generate edge-weighted directional graphs of the data
- Construct probabilities of maximum spanning walks through the graph

Aspects of the project still to be looked at include:

- The use of word embedding
- Improved data extraction from article data

1.3 Testing & Evaluation

With our data set being so large, 6226 entries, it can easily be split into training, development and testing with no need for overlap. For this the data is split 10% training, 10% development and 80% for testing.

Evaluation of the system was done using the Kendall rank correlation coefficient (Kendall Tau) as is a statistic used to measure the ordinal association between two measured quantities. This was easy be applied to our results by comparing the systems estimated date for each event with the label associated with the event from the data.

Chapter 2

Wikipedia As a Data Source

2.1 Subject Extraction

Stanfords natural language processing (NLP) suite, a natural language analysis toolkit, was selected to aid in extraction of features from the data set.

Using this toolkit, the Open Domain Information Extraction (OpenIE) system based on research from Washington University [4].

Open IE refers to the extraction of relation tuples, with the advantage of not requiring a specified schema in advance. Relation names are typically just the text linking two arguments.

As an example of OpenIE's extraction, the sentence:

"The U.S. president Barack Obama gave his speech on Tuesday to thousands of people."

Will produce these potential binary relations

"president(Barack Obama, the U.S.)"
"gave(Barack Obama,his speech)"
"gave-speech(Barack Obama, on Tuesday)"
"gave-speech(Barack Obama, to thousands of people)"

OpenIE was chosen as it allows us to easily model the relationship within the

article headline and ealisy identify subjects and objects for article retrieval about.

2.2 Text Retrieval

Using Wikipedia's article retrieval API, gathering whole articles from the site is trivial. Choosing what to extract is, however, not so straightforward

While it was simple to retrieve the data, there was simply too much noise to be of use. So it was required to minimise the noise.

For this we look at each article and only retain information that is self referencing, contains a date or references the other article subject. This method allowed us to cut down the text retained from each article from 600 words [2] to 300 words, on average. We now have much smaller article data to allow us to generate features.

2.3 Feature Extraction

Choosing which type of feature extraction method to use for our data provided several challenges.

- How to model the data as feature vectors
- Which type of feature would be most appropriate

Given that each event in out data set is of the form $(t_i, d_i)$for $i\epsilon[M]$, where t is the title, d is the associated date and M is the original data set, we constructed a new data set $\{(t_i, s_i, d_j)\}$ for $i, j[M]$ containing the sentences retrieved from Wikipedia. This will form the basis of our data to generate features.

From this we built a new data set $\{(t_i, s_i, t_j, s_j, b_{ij})\}$ for $i, j\epsilon[M]$ where $b_{ij} = [y_i > y_j]$ indicating which event came first.

With this new data set, we explored feature extraction techniques to train a classifier on the provided sentences.

2.3.1 Bag-Of-Words

Bag-Of-Words features looked promising from the start of the project as they allow easy representation of a large volume of words.

This type of feature modelling works by generating features based on word frequency over a set of all possible words.

A toy example of the bog-of-words model is as follows.
If we have two example documents

(1) "John likes to watch movies. Mary likes movies too."
(2) "John also likes to watch football games"

We can convert them into a set of words
['John', 'likes', 'to', 'watch', 'movies', 'also', 'football', 'games', 'Mary', 'too']

With this new set of all words, we can build our bag-of-words features in terms of word frequency

(1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

For our features, we built a set of all words found in all sentences in our data set. We also take each entry into out data set, $\{t_i, s_i, t_j, s_j, b_{ij}\}$, and build a feature vector from s_i and s_j.

While using a bag-of-words model is simple to implement and computationally inexpensive it does have some flaws. These include very large, sparse feature vectors and the issue that for testing, if an article has a word the model has not seen it will be ignored.

Chapter 3

Machine Learning

3.1 Approach

As our data set allows for a binary class label for each entry, $y_i > y_j$, it is apparent we should be looking for a binary classifier that we can train that will give us classification probabilities. These classification probabilities will be needed in order to weight a path through the graph that will be generated.

3.2 Local Learning

Initially we decided to look at local learning, as opposed to global learning, to try and classify our data. This was chosen so as to provide a simple baseline for us to compare against. This baseline is randomly assigned classes, but as our data conforms to a binary classification, random resolves to 50% accuracy.

3.2.1 Decision Trees

Decision trees were looked at as they are a non-parametric, supervised learning method that are commonly used for classification and regression. They learn simple decision rules inferred from data features to create a model that predicts the value of a target variable.

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

It was also decided to run the experiment using three events, rather that two to see how these effected the results.

Some issues found with decision trees were that the probabilities generated were too narrowly distributed to infer any confidence in the paths generated. Due to this, it was decided to explore other methods of classification.

3.2.2 Support Vector Machines

Similarly to Decision trees, Support Vector Machines (SVMs) provide supervised learning models based on associated learning algorithms for classification and regression.

SVMs were experimented with as given data with a binary classification build the points in space, making a non-probabilistic binary linear classifier.

This proved beneficial as the results generated were better than the decision trees 3.2.3. SVMs also provided much better classification probability, which is favourable for path generation.

3.2.3 Evaluation

For evaluation of our classifier, 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 is easily applied to our problem of correct ordering classification of event tuples.

The Tau coefficient is defined as $\tau = \frac{(number of concordant pairs) - (number of discordant pairs)}{n(n-1)/2}$ [1]

Given that the project is yet to be completed, we have not ran our experiments using the test data, but have been using the development data to act as "test" data. The results from these experiments can be seen in table 3.2.3

Accuracy	Decision Trees (tuple)	Decision Tree (triple)	SVM	Logistic Regression
With Articles	53	48	56	TBC
With Titles	43	36	TBC	TBC

Table 3.1: Local Learning Results

Chapter 4

Graphing

The next, and last, stage of the project is to build a weighted, directed graph of the results of our classified testing data. From this we wish to construct a solution for the travelling salesman problem for our graph.

4.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is a well known algorithm problem, often expressed as a graph describing the locations of a set of nodes.

The problem states a salesman who must travel between N cities. The order does not matter, but they must visit each city only once.

4.2 Solution

Our problem is similar to TSP with the added stipulation that all edges between nodes are weighted and are unidirectional.

In order to solve this problem, we built a path finding algorithm based on the minimum spanning tree algorithm.

4.3 Results

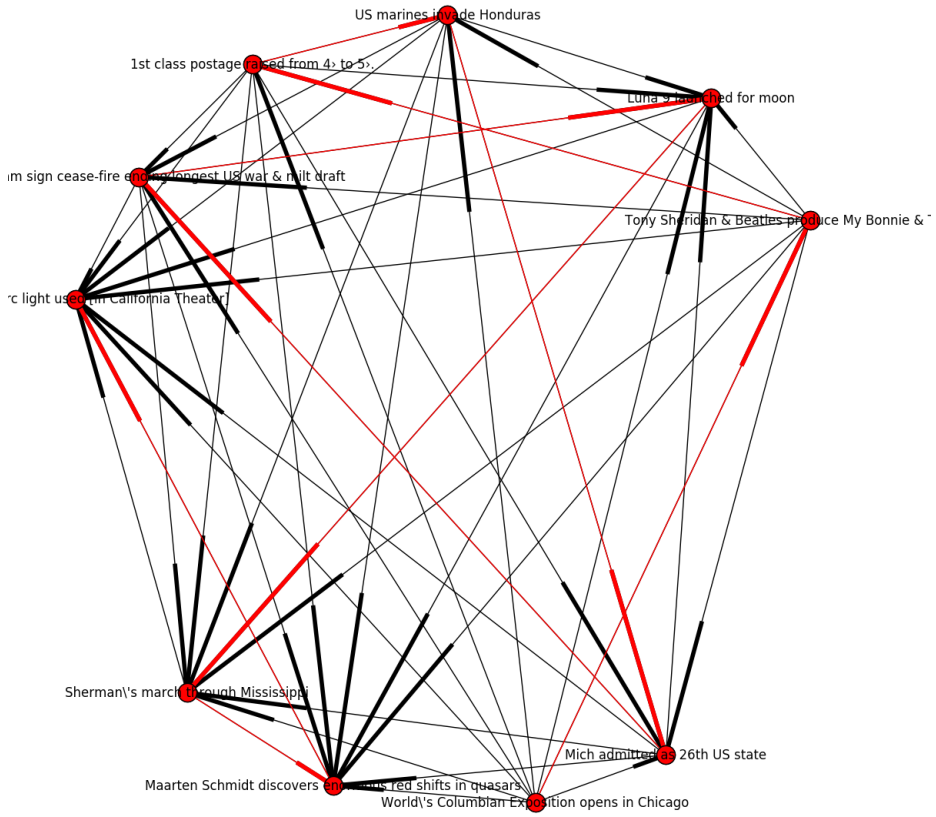

Figure 4.1: Most Probable path through all nodes

Using a small subset of our test data, we can show an optimum path through a graph of our data in figure 4.3. From our initial testing, we found that the path finding algorithm retains the accuracy of our classifiers.

Chapter 5

Conclusion

As we can see, the project is progressing well. There are a few aspects still to be looked at before completion (see below), but the project is moving along to completion on schedule.

5.1 Still To Be Done

- Word Embedding
- Structured Perceptron - Code Complete, but evaluation still to be carried out
- Logistic Regression

Bibliography

[1] Kendal tau metric. https://www.encyclopediaofmath.org/index.php/Kendall_tau_metric. Accessed: 25-01-2017.

[2] Wikipedia: Size comparison. https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons. Accessed: 23-01-2017.

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

[4] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.