Information Extraction of Seminars

Brendan Hart

December 5, 2016

1 Introduction

This report will discuss the ideas used to extract information from seminar data. The goal of the system was to extract the speaker, location, start time and end time of the seminar. As well as this, it was to construct an ontology, grouping the seminars into their respective topics or subtopics.

2 Seminar Tagging

2.1 Preparing seminar data for tagging

The first step of the `tag` method was to split the seminar data into its header and abstract components, allowing for the data from the header to be easily extracted and for the abstract to be easily searched and considered seperately. The abstract is then processed by the NER tagger, which essentially performs chunking on the abstract. This allows the ability to search through named entities in each sentence which is useful for extracting the location or speaker.

2.2 Extracting information

Now that the seminar data was in a more useful and accessible form, the next step was to extract the relevant information.

2.2.1 Sentences and Paragraphs

Paragraphs in the data seemed to be seperated by two new lines. As a result, to preserve the location of these new lines and to know which sentences should be grouped as paragraphs, the abstract is split on `\n\n`. For each element in the spit, it is replaced by the result of `nltk.sent_tokenize` with the element as the argument to the call. This now not only gives where paragraphs should be as a result of the original split, but `nltk.sent_tokenize` splits this further into sentences.
The system loops through each paragraph split, also looping through each sentence within that paragrph. At the start of each paragraph, an opening paragraph tag is added. Following this, it is necessary to determine whether a sentence produced by `nltk.sent_tokenize` should be tagged as a sentence in the text or not. For example, lines with just " SEMINAR " should not be tagged as a sentence. The system therefore contains rules to prevent against this when processing the sentences. Once the system is satisfied that the sentence is indeed a sentence, it encloses it with sentence tags and adds it to the new text, otherwise it is simply added to the new text without modification. Once we have reached the end of an element from the original split, a closing paragraph tag and the new lines are added to the text.

2.2.2 Time

The header always contains "Time:", which indicates at least one important time for the seminar. If two times are found in the header file, they are generally the start time and the end time in that order. If only one is found, it is likely to be the start time. Based on this, the system finds all times given by "Time:" in the header using a regular expression, which was `(\d{1,2}(:\d{1,2}(\s*[AaPp]\.?[Mm])?|\s *[AaPp]\.?[Mm]))`. It matches any time, such as "3pm" or "3:00". The amount of times found is then considered, and if the amount of different times found is two, then the start time is assigned as the first and the end time is assigned as the second. If there is only one time found, then it is assigned as the start time.
Providing we did not find the end time in the header, it is then necessary to look in each sentence of the abstract. Since end times generally occur after start times, the system proceeds to look for a time in the abstract which is the same as the start time in the header. To do this, it uses a method called `compareLooseTime` which will match times such as 3:00PM to 3pm. If it finds the start time, it then continues to look for the next time in the sentence, which it assumes to be the end time if found. In the event that the end time is continues for each sentence in the abstract.
Once the start time and, if it exists, the end time has been determined, the times can then be replaced throughout the seminar. To do this, a method called `looseTimeReplace` will

find all times which match the start time or end time roughly in a similar to `compareLooseTime` checks for equivalence This returns a pair of lists with one list for the start times and one for the end times. The start times are then looped through and replaced in the text with the "stime" tag enclosing them, and similarly for end times but with "etime" enclosing them.

2.2.3 Person and Location

3 Ontology Construction

4 Evaluation

The system was trained with 70% of the data and tested on 30% of the data. For comparison, the results of tagging every entity on the same 30% as an organisation are also displayed. Two types of tests were carried out, one for when the named entities were extracted from the text perfectly and their classification was correct and another for when, for example, "United States" was picked up and classified correctly but "United States of America" was the target.

References