

Entity tagging of seminar announcements

Csongor Barabasi – 1636980

1. **Outlining the Task**

The task was, given 300 tagged emails, to build a tagger which will be able to tag 300 untagged emails accurately, in the same way the tagged emails were tagged. The corpus was tagged specifically with stime, etime, sentence, paragraph, location and speaker tags.

1. **Data Pre-Processing**

The first step was reading the corpus. This was done using *WordListCorpusReader* from *NLTK*. Then, using the *raw* property of the corpus reader, we could retrieve all the emails, concatenated. Before splitting it into email chunks, a cleaning process was done, where sequences of two or more characters like *‘=, ~, %, -, \_, \*,* |*’* were removed. Next step involved splitting this long text of 300 emails into emails chunks, and then split up each email into a header and a body part, which was then stored in an *Email* object. Finding each email was done with the regular expression

split the text according to each email’s header. Then, by splitting each email according to the word *‘Abstract’*, I was able to split each email into a header and body part. The result of the pre-processing part was a list of *Email* objects.

1. **Sentence and Paragraph Tagging**

In the given corpus paragraphs can be identified as being separated by two newline characters. Hence, I could obtain a list of paragraphs from each email body by splitting the text as:

Sentence tagging was as simple as paragraph tagging. Using *NLTK’s* built-in *sent\_tokenize* function, which uses an instance of *PunktSentenceTokenizer*, I was able to split the text into sentences and tag them accordingly.

1. **Information Extraction from Header**

The header was a valuable source of information, since it provided details about start- and end-time, as well as details about the speaker. Most of this extraction was done using regular expressions.

* 1. **Time Tagging**

First I extract the line which gives information about time using the regex

Then we extract all the existing timestamps from this line using the regular expression

Next, I checked whether there were one or two timestamps found. If there was only one, it was tagged as *stime*, if there were two than the first was tagged *stime* and the second as *etime*. Next I scanned through the body and looked for all the occurences of the times found in the header, and tagged them in the body accordingly (eg. if there was one timestamp in the header, that was tagged with *stime*, so all occurences of that timestamp in the body were tagged as *stime*).