**Information Extraction from Seminar Announcements**

The program sends each email through a number of function in order to extract the relevant information and tag it. It finds the paragraphs, sentences, start (and if applicable end) times for the seminars, the speakers and the location that the seminar is held.

Firstly the email is sent to the function speaker, along with a list of all the past speakers from previous emails and to begin with the program searches through the lines of the email using a list of common words which would indicate that the speaker’s name is on that line; words like “Who”, and “Speaker”. If the indicator words are found, then the line is stripped using regex so that just the name is left. If no indicator words are found, then a list of surnames, male and female names is used to search for names within the text, and if found, the line is again stripped using regular expressions. Finally if none of the above methods have produced a name, the list of past speakers is used to search for names within the text, as often speakers have spoken at more than one seminar. Regular expressions are then used to enclose every occurrence of the name in tags to indicate the speaker.

Next, the email and a list of past locations are sent to a function, to find the location of the seminar. A list of indicator words are again used to try and find the line which contains the location, and if it’s found, the line is stripped using regular expressions, so that only the location is left. If this doesn’t work, another list containing common words linked to locations; e.g. hall, room, theatre, are used to search the email. If a line contains one of these words and is quite short in length, i.e. not in a paragraph, then that phrase is taken as the location and regex is used to strip it down. Finally, the past locations are used in the same way as with the past speakers to try and find the room. Regular expressions are then used to tag every appearance of that location in the text.

The program finds the times of the seminar by using regular expressions to search for the word “Time” within the email. This word always occurs, so it’s a simple way to classify it. Regex is used to decide whether the time phrase uses AM and PM and whether there is also an end time given. Group and search regex methods are used with regular expressions to reduce the line down to just the times and then all occurrences of the times are tagged in the text, again using regex substitution.

Sentences within the email must be within the abstract section of the email, so regular expressions are used to check whether a line is in the abstract, before the program decides if the line is a sentence. On principle, all sentences within a paragraph must be tagged so regex is used to check whether the current line is in a paragraph. Also a sentence must start as soon as a paragraph starts, and end as soon as the paragraph ends, regular expressions are again used to find these cases. One sentence stops and another begins when there is a full stop (or similar punctuation) then a capital letter (or a new line). Regular expressions have been used to find these instances and tag appropriately to show that one sentence has ended and another has begun. Finally, sentences can also exist outside of paragraphs if the entire sentence is on one line of the email, and regular expressions are used to search and tag these occurrences.

Finally, paragraphs must be within the abstract section of the email. A variable “para2” is used to look for the start of a paragraph using regular expressions. It must always be at the beginning of the line, and this line must not already be in a paragraph. Another variable “para” is used to look for the end of a paragraph, again using regex. This will always be at the end of the line, and the line must already be in a paragraph. Regular expressions are used to tag the paragraphs.

**Evaluation:**

My code appears to tag the relevant data well. It’s very effective at tagging the paragraphs and sentences, and gets the start time of the seminar most of the time. It struggles more with the speaker when there isn’t a “who” or “speaker” in the text to indicate where the name is, and if it gets the speaker tags wrong, it has often tagged the date instead. The location function tags slightly better than the speaker one, as a location is more distinguishable than uncommon names. The program often gets the end time as well, if it appears at the beginning with the start time (i.e. if it is added later on, the code doesn’t always pick it up).