OVERVIEW AND EVALUATION OF INFORMATION EXTRACTION SYSTEM

STUDENT ID 1685780

This is a report detailing an evaluation of the methods and performance of information extraction system, for information on how to run the system see the README file.

Overall evaluation scores:

Accuracy	Precision	Recall	F1
70.52%	62.92%	68.71%	64.13%

SENTENCE AND PARAGRAPH TAGGING:

	Accuracy	Precision	Recall	F1
Paragraphs:	39.92%	40.94%	71.27%	49.35%
Sentences	61.84%	63.04%	67.13%	64.33%

The paragraph tagging was achieved using regexs, with quite a bit of preprocessing, for example removing indented lines or lines by themselves: this prevented it tagging each block separated by a newline that was not an actual paragraph of English text but a line with details about the seminar. these were very common throughout the corpus so filtering them out made a large improvement.

However many invalid paragraphs were still tagged, as almost every email contains blocks of text without newline breaks that are not actual paragraphs, which the system would tag as such.

One way to deal with this would be to attempt to detect of the paragraph is made of regular English text, perhaps by frequency analysis of parts of speech. if the sentence tagging was more reliable and didn't tag things that weren't valid English sentences then one method to tag paragraphs would be to tag blocks made up of sentences. this would help prevent many of the false positives the system currently produces.

The sentence tagging performed better than the paragraphs, this is probably because I only tagged things already inside paragraphs as sentences, so many false positives were avoided. This is because I used the nltk punkt tokenizer, which tokenizes what it is given, it doesn't try to detect if something is a sentence, it just splits what it is given into the most probable sentences. This means quite a lot of preprocessing is required to decide what should be given to the text before giving it to the tokenizer. In this case I sent different paragraphs separately to prevent sentences spanning multiple paragraphs but a better measure would have been to actually check if what was being tagged was composed of valid sentences before tokenizing it into sentences, perhaps by parsing the POS tags.

TIME TAGGING:

Time tagging performed well thanks to it relying on simple regular expressions to help with classifying which times were which.

Accuracy	Precision	Recall	F1
86.26%	67.41%	68.25%	67.40%

Many examples of not correctly identifying start and end times were due to overreliance on the fact that 'time – time' indicates a start and end, in reality this is sometimes a window of time for people to arrive or eat lunch. As the system performed well there isn't a real need for improvement, but the addition of a knowledge base and examining the surrounding text would help identify examples where the time isn't actually of a seminar. Also the decision process when given more than 2 times would benefit from actually being able to examine the semantics of the text around it (currently it is very naive).

SPEAKER TAGGING:

This is the part of the assignment I am the most happy with and spent the most time on. Whilst it didn't get as high scores as locations I'm very satisfied with the performance and results.

Accuracy	Precision	Recall	F1
74.15%	63.29%	61.92%	62.5%

The speaker picking consists of 2 stages: one to pick out people from the text and another to process these and decide on the speaker.

The first stage consisted of a pipeline that would gather a large number of possible candidates from the text and filter out unwanted items such as dates, acronyms and entities that were not people. The first step of the pipeline is to use regular expressions and a POS tagger to pick capitalised phrases without picking words that could not represent people.

It is important to note that an important stage in picking people from all possible candidates is filtering against a list of names, based on whether > 1/3 of the phrase consists of names. This was largely taken from the nltk names corpus, but with custom modifications. This included adding abbreviations of middle names such as 'A.', adding titles and removing words that provided false positives (the original list classified 'urban violence' as a name). This stage in the filtering is probably the most selective which is good and bad. The upside is that as a result the first stage is very good at only picking people, when looking through the list of people it finds in a document there are almost 0 false positives. The negative consequences are that it is poor at recognising names it does not have in the corpus. This is not entirely an issue since it accepts any phrase that it recognises > 1/3 of to be a valid name (so Mr unrecognised-surname is still valid as Mr is recognised), but especially in the testing data it fell down when classifying foreign names. Additionally the system is very sensitive to the constant of 1/3, change it either higher or lower and there is a significant hit to performance. I in fact believe that most errors in accuracy and precision are due to the system not being able to recognise entirely foreign names not in its dictionary (I tested this by counting files where there was no intersection between the list of actual speakers and people I had detected).

In an attempt to remedy this I found a github repository with a corpus of names from other languages (https://gist.github.com/mbejda/7f86ca901fe41bc14a63#file-indian-male-names-csv for example) and added this to the names corpus, but because the corpora were for a specific nationality and the names my system could not recognise were spread from all over the world it would mean hunting down corpora for every country and language. additionally it increased false positive rate, as within 14000 male Indian names are quite a few English words that I do not have the time to manually remove. So adding these corpora damages precision whilst not having a significant enough impact on accuracy to be worth leaving in.

Compressing the different names and identifying people really helped the system. Once the first stage has identified a list of peoples names it then goes through and uses overlapping parts (excluding titles) to decided which names represent the same person. The output of this process is a dictionary of numbers to lists of unique people.

e.g the initial list of people: **['Oren Etzioni', 'Raul Valdes-Perez', 'Oren Etzioni', 'Raul Valdes-Perez', 'Cal', 'Cornell', 'Mike Perkowitz', 'Professor Etzioni', 'Phyllis']**

becomes the following dictionary: **{0: ['Oren Etzioni', 'Oren Etzioni', 'Professor Etzioni'], 1: ['Raul Valdes-Perez', 'Raul Valdes-Perez'], 2: ['Cal'], 3: ['Cornell'], 4: ['Mike Perkowitz'], 5: ['Phyllis']}**

I am really happy with this part of the system, it performs well and it helps in speaker picking. The reason it is useful is that instead of choosing whether each name is a speaker or not the system can decide whether each person is a speaker considering all instances of their names, and then tag those. This means that if it identifies that a name is in the 'Who:' header it will also tag every other instance that represents the same person. This massively simplifies the process of picking out speakers.

When picking the speaker out of the people the system has detected it mainly relies on simple assumptions / heuristics. First it filters out people who sent the email by cross referencing the 'PostedBy' line, or the last line. Then I assume that as long as at least one person has been detected there is one speaker, this is the person with their name mentioned most in the text, which relies on the previously mentioned system of mapping names to people.

Then the system looks for identifying phrases surrounding the names like 'name will'. I tested a large number of these phrases such as 'by name' or looking for names appearing on their own line, or that were indented. However all of these except 'name will' only increased accuracy by about 1% but decreased precision and recall by around 5%, so in the end the only one I left in was the 'name will' pattern.

One method I would have liked to get working is using a parse tree of the POS tagged emails to try and determine the speaker by examining the relationships between tags (e.g look at the verb they are associated with and see if that is synonymous with giving a lecture).

LOCATION TAGGING:

The location tagging was statistically the most successful part of the assignment with great accuracy, precision and recall.

Accuracy	Precision	Recall	F1
74.69%	75.45%	75.45%	75.45%

The performance on testing data is actually improved from that on the training data, this is because one of my methods to find locations was to use a regex that picked up room abbreviations such as 'WeH 5409' and also prefers locations in this form when picking the seminar location out of all places detected. because the test data locations are mostly in this form (more so than the training data) that makes the system very successful when picking up locations. Also attributing significance to locations with 'Room' or 'Hall' in the name significantly improves performance.

For location tagging I did not use POS tagging to help find locations in the end, I did initially use it for the first stage, to collect initial matches to filter and pick from, but found that it provided too many false positives, whereas focusing on using regular expressions provided higher precision.

TOPIC TAGGING:

The topic tag is attached to help the ontology classify emails. The 3 places this system looks for information on the topic is the 'Type:' line, which frequently identifies the department, the 'Topic:' line which contains useful keywords and also it looks for titles of talks in speech marks within the body. Anything within speech marks or topic line is put together, split into words and filtered. The filtering removes common English stopwords (found on github), a list of custom stopwords like synonyms for seminar, numbers and punctuation. Then a list of subject words (taken from a UCAS website list of university subjects and customised) is used to check against the tokenised text, any common words are also added to the list of topic words e.g 'physics'.

An example is from email 309.txt: **'<topic>planetary walking computer science robotics robot planning exploration architecture cs cmu autonomous</topic>'** which gives a pretty good indication of what this email is about with little redundant information.

PART 2 ONTOLOGY CLASSIFICATION:

The system uses a tree structure made from dictionaries to categorise emails, it goes through each email's topic tag (my topic tag, not the email header) and calculates a similarity score with a list words describing each possible ontology leaf node, and inserts into the most similar.

In my opinion the ontology system performance is not really good enough to be used in an actual system. It does a good job at distinguishing between higher levels of classification (it places all the emails correctly within computer science) but is poor at determining sub categories.

The main reasons for this are that the use of word2vec to determine similarity scores is not good enough for this specific a domain. Word2vec is an amazing resource, especially given it is open domain

and performs well on large swathes of the english language. However this problem and especially the test data is within a very specific domain. almost all the test emails are within the category of computer science. This means that the similarity scoring system needs to be able to distinguish between the different fields of computer science well enough to categorise the emails, however it is not. The main issue here is that it is better at associating general words such as 'design', 'system' and 'architecture', this means it largely associates computer science emails with the cs departments with most generic words. A considerable number of the emails are in robotics and word2vec isn't good enough at distinguishing between the concepts of robotics design and engineering. Additionally it can also be too sensitive, noticing the word 'materials' in an email despite most words being compsci related led to a physics classification (in an earlier version of the system). Because of word2vec it is difficult to predict which words associated with topics will classify well, for example including 'design' and 'build' in engineering emails misclassifies many computer science emails. Attempting to weight the system to prefer subject words does not seem to result in a drastic performance increase, although in the end I did manage to make it recognise cs emails as belonging in cs.

The way to avoid these problems is to rewrite the ontology hierarchy to be more coarse; this arguably results in a less useful system since if an email is sent to the cs mailing list knowing that it is about computer science is redundant. But doing so will remove the issue of classifying subdomains of subjects, and as the system currently is the classification system would not be useful in determining whether an email concerns robotics, general computer science or nlp.

An interesting extension to the project would be to use the topic tagged words and try to cluster the data to examine how well the words picked by the topic tagger distinguish the emails. This would help give an indication of how useful the topic words would be when considered by word2vec.

I briefly experimented with using naïve bayes to sort emails into predicted classes (type field) using the text in their body, it had a success rate of 217/300 but the type field is not a useful enough classification. Also to improve it would require supervised training data.

The appendices contain an illustration of the produced classification tree.

OVERALL:

I am very happy with the system, its performance with tagging is good and even though it isn't always great at distinguishing nlp from robotics it is still good at classifying subjects as within cs.

The system is incredibly domain specific: it is hard coded to look for the types of locations frequently seen in these emails. This means if the corpus was changed the system would likely not generalise well and perform poorly. Perhaps it is overly reliant on hand coded rules based off the training data, specifically regular expressions. I did try to fit in some of the more advanced NLP techniques such as wikification, or using wordnet as well as word2vec but frequently I found that these actually hurt performance and I didn't want to include them for the sake of it. In total the system used 74 regexs.

APPENDICES:

Note: f1 scores outputted by system are sometimes incorrect by 1 or 2 % (only in the NER and time scores) for this report I have recalculated them by hand using the precision and recall.

All evaluation statistics are generated using Eval.py (and corpus averages calculated in Code.py)

To see the generated statistics for yourself at run time run Code.py and the overall stats will be displayed at the end

Paragraph scores:
accuracy: 0.399254103118
precision: 0.409358244413
recall: 0.712700569358
f1: 0.493462114329

Sentence scores:
accuracy: 0.618363390927
precision: 0.630353989492
recall: 0.671342200971
f1: 0.643340620787

Time scores:
accuracy: 0.862635869565
precision: 0.674094202899
recall: 0.682518115942
f1: 0.673964803313

NER scores:
accuracy: 0.741496304296
precision: 0.632925724638
recall: 0.619202898551
f1: 0.608580368906

Location scores:
accuracy: 0.74687068668
precision: 0.754528985507
recall: 0.754528985507
f1: 0.754528985507

Overall:
accuracy: 0.705209370692
precision: 0.629225891641
recall: 0.687135147712
f1: 0.641306949359

Pretty print of final ontology dictionary tree:
(I have removed parts of tree nothing was inserted into and topicwords attached to each topic)

{ 'Top':
'science': {
'CS': { 'general-compsci': { 'talks': ['files/my_tagged/301.txt',
'files/my_tagged/303.txt', 'files/my_tagged/311.txt', 'files/my_tagged/316.txt', 'files/my_tagged/320.txt',
'files/my_tagged/323.txt', 'files/my_tagged/333.txt', 'files/my_tagged/353.txt', 'files/my_tagged/360.txt',
'files/my_tagged/368.txt', 'files/my_tagged/370.txt', 'files/my_tagged/375.txt', 'files/my_tagged/380.txt',
'files/my_tagged/400.txt', 'files/my_tagged/405.txt', 'files/my_tagged/412.txt', 'files/my_tagged/415.txt',
'files/my_tagged/418.txt', 'files/my_tagged/419.txt', 'files/my_tagged/420.txt', 'files/my_tagged/424.txt',
'files/my_tagged/425.txt', 'files/my_tagged/426.txt', 'files/my_tagged/427.txt', 'files/my_tagged/439.txt',

'files/my_tagged/441.txt', 'files/my_tagged/443.txt', 'files/my_tagged/449.txt', 'files/my_tagged/451.txt',
'files/my_tagged/452.txt', 'files/my_tagged/465.txt', 'files/my_tagged/469.txt', 'files/my_tagged/474.txt',
'files/my_tagged/475.txt', 'files/my_tagged/482.txt'],

'hci': { 'talks': ['files/my_tagged/397.txt', 'files/my_tagged/402.txt',
'files/my_tagged/413.txt', 'files/my_tagged/432.txt', 'files/my_tagged/438.txt', 'files/my_tagged/461.txt',
'files/my_tagged/481.txt'],
'nlp': { 'talks': 'files/my_tagged/306.txt', 'files/my_tagged/314.txt',
'files/my_tagged/366.txt', 'files/my_tagged/372.txt', 'files/my_tagged/377.txt', 'files/my_tagged/385.txt',
'files/my_tagged/387.txt', 'files/my_tagged/389.txt', 'files/my_tagged/390.txt', 'files/my_tagged/392.txt',
'files/my_tagged/393.txt', 'files/my_tagged/394.txt', 'files/my_tagged/395.txt', 'files/my_tagged/396.txt',
'files/my_tagged/398.txt', 'files/my_tagged/399.txt', 'files/my_tagged/401.txt', 'files/my_tagged/403.txt',
'files/my_tagged/404.txt', 'files/my_tagged/407.txt', 'files/my_tagged/408.txt', 'files/my_tagged/410.txt',
'files/my_tagged/411.txt', 'files/my_tagged/414.txt', 'files/my_tagged/417.txt', 'files/my_tagged/421.txt',
'files/my_tagged/422.txt', 'files/my_tagged/423.txt', 'files/my_tagged/428.txt', 'files/my_tagged/429.txt',
'files/my_tagged/430.txt', 'files/my_tagged/431.txt', 'files/my_tagged/433.txt', 'files/my_tagged/434.txt',
'files/my_tagged/435.txt', 'files/my_tagged/436.txt', 'files/my_tagged/437.txt', 'files/my_tagged/440.txt',
'files/my_tagged/442.txt', 'files/my_tagged/444.txt', 'files/my_tagged/445.txt', 'files/my_tagged/446.txt',
'files/my_tagged/447.txt', 'files/my_tagged/450.txt', 'files/my_tagged/453.txt', 'files/my_tagged/454.txt',
'files/my_tagged/455.txt', 'files/my_tagged/457.txt', 'files/my_tagged/458.txt', 'files/my_tagged/459.txt',
'files/my_tagged/460.txt', 'files/my_tagged/462.txt', 'files/my_tagged/463.txt', 'files/my_tagged/464.txt',
'files/my_tagged/466.txt', 'files/my_tagged/467.txt', 'files/my_tagged/468.txt', 'files/my_tagged/470.txt',
'files/my_tagged/471.txt', 'files/my_tagged/472.txt', 'files/my_tagged/473.txt', 'files/my_tagged/476.txt',
'files/my_tagged/477.txt', 'files/my_tagged/478.txt', 'files/my_tagged/479.txt', 'files/my_tagged/480.txt'],

'robotics': { 'talks': ['files/my_tagged/302.txt', 'files/my_tagged/304.txt',
'files/my_tagged/305.txt', 'files/my_tagged/307.txt', 'files/my_tagged/308.txt', 'files/my_tagged/309.txt',
'files/my_tagged/310.txt', 'files/my_tagged/312.txt', 'files/my_tagged/313.txt', 'files/my_tagged/315.txt',
'files/my_tagged/317.txt', 'files/my_tagged/318.txt', 'files/my_tagged/319.txt', 'files/my_tagged/321.txt',
'files/my_tagged/322.txt', 'files/my_tagged/324.txt', 'files/my_tagged/325.txt', 'files/my_tagged/326.txt',
'files/my_tagged/327.txt', 'files/my_tagged/328.txt', 'files/my_tagged/331.txt', 'files/my_tagged/332.txt',
'files/my_tagged/334.txt', 'files/my_tagged/335.txt', 'files/my_tagged/336.txt', 'files/my_tagged/337.txt',
'files/my_tagged/338.txt', 'files/my_tagged/339.txt', 'files/my_tagged/340.txt', 'files/my_tagged/341.txt',
'files/my_tagged/342.txt', 'files/my_tagged/343.txt', 'files/my_tagged/344.txt', 'files/my_tagged/345.txt',
'files/my_tagged/346.txt', 'files/my_tagged/347.txt', 'files/my_tagged/348.txt', 'files/my_tagged/349.txt',
'files/my_tagged/350.txt', 'files/my_tagged/351.txt', 'files/my_tagged/352.txt', 'files/my_tagged/354.txt',
'files/my_tagged/355.txt', 'files/my_tagged/356.txt', 'files/my_tagged/357.txt', 'files/my_tagged/358.txt',
'files/my_tagged/359.txt', 'files/my_tagged/361.txt', 'files/my_tagged/362.txt', 'files/my_tagged/363.txt',
'files/my_tagged/364.txt', 'files/my_tagged/365.txt', 'files/my_tagged/367.txt', 'files/my_tagged/369.txt',
'files/my_tagged/371.txt', 'files/my_tagged/376.txt', 'files/my_tagged/379.txt', 'files/my_tagged/381.txt',
'files/my_tagged/384.txt', 'files/my_tagged/388.txt', 'files/my_tagged/391.txt', 'files/my_tagged/406.txt',
'files/my_tagged/416.txt'],

'software-eng': { 'talks': ['files/my_tagged/329.txt', 'files/my_tagged/330.txt',
'files/my_tagged/373.txt' , 'files/my_tagged/374.txt', 'files/my_tagged/378.txt', 'files/my_tagged/382.txt',
'files/my_tagged/383.txt', 'files/my_tagged/386.txt', 'files/my_tagged/409.txt', 'files/my_tagged/448.txt',
'files/my_tagged/456.txt', 'files/my_tagged/483.txt'],

An image of the original empty ontology tree in my IDE (to give an idea of subjects included and topic words used):

```
{ 'Top' :
        { 'arts' :
                { 'art' :
                        { 'topicWords' : [ 'art', 'artist', 'painting'], 'talks' : list()}
                , 'other'   :
                        {'topicWords' : [ 'arts', 'philosophy', 'english', 'language'], 'talks' : list()}
                }
        , 'humanities' :
                { 'politics' :
                        { 'topicWords' : [ 'politics', 'politician', 'law'], 'talks' : list()}
                , 'sociology' :
                        { 'topicWords' : [ 'sociology', 'social', 'customs', 'behaviour', 'communities'], 'talks' : list()}
                , 'history' :
                        { 'topicWords' : [ 'history', 'civilisation', 'ancient'], 'talks' : list()}
                }
        , 'science' :
                 { 'physics' :
                        { 'particle-physics' :
                                { 'topicWords' : [ 'physics', 'particle', 'proton', 'atomic'], 'talks' : list()}
                        , 'astro-physics' :
                                { 'topicWords' : [ 'physics', 'astrophysics', 'astronomy', 'star', 'stars'], 'talks' : list()}
                        , 'general-physics' :
                                { 'topicWords' : [ 'physics', 'science', 'forces'], 'talks' : list()}
                        }
                , 'biological' :
                        { 'medicine' :
                                { 'topicWords' : [ 'medicine', 'medical', 'health', 'pharmaceutical', 'patient'], 'talks' : list()}
                        , 'biology' :
                                { 'topicWords' : [ 'biology', 'cell', 'science'], 'talks' : list()}
                        }
                , 'chemistry' :
                        { 'topicWords' : [ 'chemistry', 'chemical', 'interaction'], 'talks' : list()}
                , 'CS' :
                        { 'hci' :
                                { 'topicWords' : [ 'cs', 'computer', 'science', 'hci', 'HCI','human', 'interaction', 'user'], 'talks' : list()}
                        , 'vision' :
                                { 'topicWords' : [ 'cs', 'computer', 'science','vision', 'AI', 'edge', 'detect'], 'talks' : list()}
                        , 'robotics' :
                                { 'topicWords' : [ 'cs', 'computer', 'science','robotics', 'robot', 'intelligence', 'autonomous'], 'talks' : list()}
                        , 'nlp' :
                                { 'topicWords' : [ 'cs', 'computer', 'science','linguistics', 'language', 'semantics', 'syntax'], 'talks' : list()}
                        , 'software-eng' :
                                { 'topicWords' : [ 'cs', 'computer', 'science','programming', 'system', 'design'], 'talks' : list()}
                        , 'general-compsci' :
                                { 'topicWords' : [ 'cs', 'computer', 'science', 'AI', 'machine', 'learning', 'program', 'system', 'security', 'network']
                        }
                , 'materials' :
                        { 'topicWords' : [ 'materials', 'metals', 'polymer'], 'talks' : list()}
                , 'engineering' :
                        { 'topicWords' : [ 'engineering', 'design', 'build'], 'talks' : list()}

                }

        , 'other' :
                { 'teaching' :
                        { 'topicWords' : [ 'teaching', 'education', 'mam'], 'talks' : list()}
                , 'economics' :
                        { 'topicWords' : [ 'economics', 'monetary', 'policy', 'model', 'fiscal'], 'talks' : list()}
                , 'law' :
                        { 'topicWords' : [ 'law', 'legal', 'court'], 'talks' : list()}
                }
        }
}
```