1 Introduction

Our system detects named entities from a text and adds wikipedia links to the entities. It generates also HTML files with the result that can be viewed in a browser. We used an object oriented programming approach making it easy to use code again developed earlier in the course. We have chosen for detecting named entities to train and the Stanford NER classifier.

We did this by splitting our development dataset in a 80% train file and a 20% test file. We then processed our train file by removing wrong annotations (e.g. using - or other characters instead of a space) and output it in a format required for training the classifier. The trained classifier was then used in our main script *(Ner.py)* that uses the test file.

After all documents are classified we use the inherited measures script to calculate precision recall and f-score. During classifying we also look for Wiki links that will be explained in the next section.

2 system

Our system consists of three python files:

- prepareTraindata.py
- Ner.py
- Measures.py

prepareTraindata.py Is the file that formats the given dataset to a format needed for training the classifier. It does that by checking the number of elements on one line if we split that line on whitespace. A line with a length of 6 is not a named entity, with 7 is a named entity without a link and 8 elements means the token is a named entity with a wikipedia link. The data format needed to train the classifier is:

```
New         CIT
York        CIT
is          O
a           O
great       O
city        O

Groningen           CIT
is          O
a           O
great       O
city        O
```

We used this file to train the classifier using the standard Java software. We used the same features as the standard classifiers use.

Ner.py is our main script that uses the trained classifier. We have a testset (20% of our development dataset) that we used to test our system. We read our testfile that looks like this:

```
g7.2 0 9 1001 President NNP PER <URL>
g7.2 10 14 1002 Bush NNP PER <URL>
g7.2 15 19 1003 said VBD
g7.2 20 27 1004 nations NNS
g7.2 28 34 1005 around IN
g7.2 35 38 1006 the DT
g7.2 39 44 1007 world NN
g7.2 45 49 1008 need VBP
g7.2 50 52 1009 to TO
g7.2 53 58 1010 stand VB
g7.2 59 63 1011 with IN
g7.2 64 72 1012 moderate JJ
g7.2 73 82 1013 reformers NNS
```

We make a dictionary with as key the document id and add a list of each line. If the line has no category and link we add a empty value so each list has the same length (8). The second step is that we classify all tokens with our trained classifier. We append the classification to our list. So we get at index 8 our expected category. Add index 9 we add our expected wikipedia link and score as can be seen in the next example:

```
['g14.7', '483', '488', '5004', 'prime', 'NN', '', '', 'O', '']
['g14.7', '489', '497', '5005', 'minister', 'NN', '', '', 'O', '']
['g14.7', '497', '498', '5006', ',', ',', '', '', 'O', '']
['g14.7', '499', '504', '5007', 'Juhan', 'NNP', 'PER', '<url>', 'PER', '<url>
['g14.7', '505', '510', '5008', 'Parts', 'NNP', 'PER', '<url>', 'PER', '<url>
```

We find the Wikipedia page by making query strings of the tokens that are classified as a named entity. In the example above the words *Juhan* and *Parts* are both annotated as a person. We wrote a function that joins entities that belong to each other to find the wiki link and assigns the found link to both tokens. We use the Wikipedia API to search pages related to *Juhan Parts*. For now we select the first result because Wikipedia sees that at the most logical result for that query. We use that link with a score of 1 to append to our list. The final result is saved as HTML files that can be viewed in a browser.

Measures.py Is a script that is used (a part) in previous assignments and used again. We use the class in our Ner.py script. The Measures class has a function "calculate" that requires the dictionary described in the previous section. It uses the annotated category and url and expected category and url

to calculate the precision and recall of our classifier and the accuracy of our Wikipedia links. The results can be found in the next chapter.

3 results

Our system has a f-score of 0.72 for detecting entities. Compared to other researches this is reasonable. The test set size is too small for further analysis.

Table 1: Confusion matrix

	CIT	COU	NAT	O	ORG	PER
CIT	<5>	2	.	1	.	2
COU	.	<17>	.	7	.	.
NAT	..	.	<.>	4	.	.
O	.	1	.	<507>	6	.
ORG	.	.	.	3	<1>	.
PER	1	1	.	7	.	<9>

The precision and recall of all categories:

Table 2: Precision and recall

	Precision	Recall	f-score
COU	0.81	0.71	0.76
CIT	0.83	0.50	0.62
NAT	0	0	0
PER	0.82	0.50	0.62
ORG	0.14	0.25	0.18
ANI	0	0	0
SPO	0	0	0
ENT	0	0	0
TOTAL	**0.63**	**0.84**	**0.72**

The accuracy of our Wikipedia system is 0.51. We think we can improve this by looking at all search results and selecting the best one.