Exercise Sheet 9

Information Extraction II

*Submit your solutions until **Monday, 09.05.2016, 12h00** by uploading them to ILIAS. Later submissions won't be considered. Every solution should contain the **name(s), email adress(es)** and **registration number(s)** of its (co-)editor(s).*

1 Named Entity Classification (NEC) (10 Points)

1.1 Manual Annotation (5 Points)

In the following text find all **named entities** and annotate them. Write down the complete named entity, the line where you found it and its type (Person, Location or Organization):

```
1| Justin Drew Bieber (born March 1, 1994) is a Canadian pop musician,
2| actor, and singer-songwriter. Bieber was discovered in 2008 by American
3| talent manager Scooter Braun, who came across Bieber's videos on YouTube
4| and later became his manager. Braun arranged for him to meet with
5| entertainer Usher Raymond in Atlanta, Georgia, and Bieber was signed to
6| Raymond Braun Media Group (RBMG), and then to an Island Records
7| recording contract offered by record executive L.A. Reid.
```

1.2 Automatic NEC (5 Points)

Download the documents attached to this exercise from ILIAS, open the document **justin.txt** as a corpus and run the ANNIE pipeline on it. Compare the resulting annotations of named entities with the ones created by you: What are the differences in the annotations? Which types of errors occur in the automatic annotation?

2 Information Extraction (31 Points, 10 Bonus Points)

On the previous exercise sheet you learned about some basic functionality of the software GATE (blue!80`http://gate.ac.uk`). This exercise will deal with JAPE and pattern matching in GATE. JAPE (Java Annotations Patterns Engine) is a system that can add annotations based on regular expressions. It is controlled with JAPE rules that define patterns to match and annotate parts of the document: blue!80`http://gate.ac.uk/sale/tao/splitch8.html#chap:jape`.

In ILIAS you find some texts about musicians and bands along with a `.jape` file. The beginning of the JAPE file looks like this:

```
Phase: Birthdate
Input: Person Organization Unknown Token
Options: control = appelt

Rule: Birthdate
(
  ({Person}):person
  {Token.string == "("}
  ({Token.string == "born"})
  ({Token}):month
  ({Token.kind == "number"}):day
  ({Token.kind == "punctuation"}) //the comma after the day
  ({Token.kind == "number"}):year
  {Token.string == ")"}
)
-->
  :person.Birthdate = {rule = "Birthdate", birthyear = :year.Token.string,
  birthmonth = :month.Token.string, birthday = :day.Token.string}
```

It starts with defining a phase. A phase has a name (*Birthday*), a defined input of annotations that are considered to be matched (*Person Organization Unknown Token*) and options (in this case defining the control style). Each phase consists of rules having a left hand side (LHS) and a right hand side (RHS) which are divided by a right arrow - ->. The LHS defines a pattern to be matched, the RHS describes the annotations to be made. Variables start with a colon (*:person*). Each annotation has properties that can be accessed using the dot notation, e.g. the text string property of *Token* annotations is acces by "Token.string" while the kind of the token is accessed by "Token.kind". Exactly one annotation is described within curly brackets { }, braces () are used to form groups. Only groups can be assigned to variables[1] that can be accessed on the RHS to add annotations using the dot notation. For example the line *:person.Birthdate = ...* adds the annotation *Birthdate* to the span of text that is matched by *:person*.

If you load a `.jape` file into GATE [2] and append it to the ANNIE processing pipeline, each phase is used once for each document in the corpus by applying the rules in that phase. Phases are loaded in the order they are specified. For details have a look at this JAPE tutorial: blue!80`http://gate.ac.uk/sale/thakker-jape-tutorial/index.html`

[1] That is why we have to write "({Person}):person" instead of "{Person}:person".
[2] right click on "Processing Resources" => new => JAPE transducer

2.1 Happy Birthday (4 Points)

Load the file `relex.jape` as a transducer and run it on the documents in the provided directory. Inspect the new "Birthdate" annotations:

1. Which files contain the annotation of type "Birthday"?
2. For one of the files report the complete information of the annotation (type, start, end, and features).[3]

2.2 Adding Information with the *IsA* Relation (27 Points, 10 Bonus Points)

After detecting limited, predefined set of types or relations like birthday, let's look at the task of finding additional information on entities that is more detailed than the main classes person, organization and location. For example, we would like to know that Metallica is an american heavy metal band. Have a look at the *IsA* rule.

```
01| Phase: IsA
02| Input: Person Organization Unknown Token
03| Options: control = appelt
04|
05| Rule: IsA
06| (
07| ({Person} | {Organization} | {Unknown}):entity
08| ({!Person, !Organization, !Unknown})*
09| ({Token.string == "is"})
10| ({Token.string == "a"} | {Token.string == "an"})
11| (({Token.category == "JJ"})*({Token.category == "NN"})+):isa
12| )
13| -->
14| :entity.IsA = {rule = "IsA", IsA = :isa@string}
```

[3] Available from the *Annotation List*-view

a) Alternatives, Negation, Sequences (20 Points)

This rule uses some new concepts like alternatives, negation and sequences. Find out how they work to answer the following questions:

1. What does the operator | ("bar") do in line 7? Give two examples of a match for the pattern in line 7.
2. What do the operators ! and $*$ ("kleene star") do in line 8? Describe in your own words what is matched by the complete pattern in this line.
3. What does the operator + do in comparison to $*$?
4. Which kind information is provided by *Token.category*? Give two examples of a match for the pattern in line 11.
5. Which information is accessed by the meta property *@string*? What happens to the *isA* annotations if you change line 14 to the following:

```
:entity.IsA = {rule = "IsA", IsA = :isa.Token.string}
```

6. Give a short description what is matched by the complete *IsA* rule.
7. How could Watson take advantage of this information? [4]

b) Debugging (7 Points)

1. Why can't the rule find an *IsA* relation for Nirvana? Fix the rule in order to find the relation.
2. Why can't the rule find the relation *IsA - singer-songwriter* for Madonna?[5] Why would it not be a good idea to change the **rule** in order to find the relation for Madonna. How should this error be fixed instead?
3. Why is there no birthday annotation for "David Eric "Dave" Grohl"?

c) Persons with a Nickname (10 Bonus Points)

Write a new rule that is able to match "David Eric "Dave" Grohl", annotate the match with the annotation type *Person* and add the property *nickname* with the value "Dave" to it as shown in Figure 1.

[4] For information about Watson refer to Task 3.
[5] HINT: Have a look at the *Token* annotations in the text.

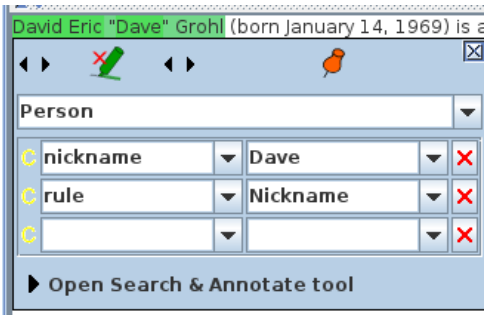

Figure 1: Nickname annotation.

3 Watson (14 Points)

Watson is IBMs question answering system that was developed to compete with humen contestants in the game shoe *Jeopardy*. The original paper describes the creation of Watson and can be found at blue!80`http://aaaipress.org/ojs/index.php/aimagazine/article/download/2303/2165` (Ferrucci et al., 2010). Other useful resources might be this video blue!80`http://www.youtube.com/watch?v=dQmuETLeQcg` or the Watson website at blue!80`http://www-03.ibm.com/innovation/us/watson/index.shtml`. Answer the following questions:

1. What is the definition of lexical answer types (LAT)? You may cite the paper or use your own words to answer this question.
2. Why is extracting the LAT of a question important to answer it?
3. How does named entity classification relate to LAT?[6]
4. What are or could be practical applications of a system like Watson?

References

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[6] HINT: Look at the top entries of Figure 1 in the paper by (Ferrucci et al., 2010)