EDAN20
Final Examination

Pierre Nugues

October 25, 2012

The examination is worth 183 points. The distribution of points is indicated with the questions. You need 60% to have a mark of 4 and 80% to have a 5.

1 Closed Book Part: Questions

In this part, no document is allowed. It is worth 83 points.

Chapter 1. IBM Watson is a pioneering system in language technology. Describe what it can do. 3 points

Chapter 1. Describe what is the result of dependency parsing on the sentence *The boy hit the ball* with the grammatical relations: subject, verb (or root), and object. 4 points

Chapter 1. Name the typical components of a parsing pipeline to carry out the analysis of the previous sentence. You can think of the steps (columns) used in the CoNLL corpora. 4 points

Chapter 2. Identify what the regular expressions in the list below will match in the text (identify all the matches unless specified, or write no match if there is no match): 10 points

> Tysklandsfärjor blev svenska
> Ekonomi. Färjorna mellan Trelleborg och Tyskland ägs nu av Stena Line.
> Formellt lämnade tyska Scandlines GmbH över fem färjelinjer till Stena i förra veckan. Säljaren fortsätter dock att köra linjerna under en övergångsperiod. För skåningarna märks ägarbytet på linjerna Trelleborg–Rostock och Trelleborg–Sassnitz. Köpet omfattar dock även även linjerna från tyska Travemünde till de båda lettiska hamnarna Ventspils och Liepaja. Också linjen Nynäshamn-Ventspils ingår i paketet. Stena Line kör därmed 22 linjer med 37 färjor i norra Europa.
>
> *Sydsvenskan.se*, Retrieved October 15, 2012.

List of regular expressions. The symbol ␣ means a white space (visible white space):

1. `Lin*e`
2. `Lin?e`
3. `\d+`
4. `␣\w{11,12}␣`
5. `\d.+\d`
6. `\d.+\w`
7. `\d\.\w`
8. `(\w+)␣\1`
9. `\.$`
   With no modifier
10. `[^abc]de`

Chapter 2. Write a regular expression that will match the word to the left of *GmbH* and store it in a memory store. 3 points

Chapter 2. What does this Unix command do?
`tr 'åäéöÅÄÉÖ' 'aaeoAAEO' <file.txt` 1 point

Chapter 2. Represent graphically two possible alignments of the strings *language* and *lineage* using the edit operations copy and delete/insert. The alignments do not need to correspond to a minimal edit distance. 6 points

Chapter 3. Describe briefly how UTF-8 encodes Latin accented characters such as *Ä* or *Ö*. 3 points

Chapter 3. Describe briefly how Unicode sorts (collates) strings. 3 points

Chapter 4. Give the probabilistic model of the sentence *Stena Line kör därmed 222 linjer* using no no n-gram approximation. You will ignore possible start and end of sentence symbols. 2 points

Chapter 4. Using a unigram approximation, give the probabilistic model of the sentence *Stena Line kör därmed 222 linjer*. You will ignore possible start and end of sentence symbols. 2 points

Chapter 4. Using a bigram approximation, give the probabilistic model of the sentence *Stena Line kör därmed 222 linjer*. You should have exactly the same number of terms as in the previous question. You will ignore possible start and end of sentence symbols. 2 points

Chapter 4. Using the counts in Table 1, you will compute the probability of the sentence *Stena Line kör därmed 222 linjer* using a unigram and a bigram approximations. You will use fractions to represent the terms in the product and you will not try to reduce them, *i.e.* you will write $\frac{1}{3}$ and not 0.33. 6 points

You may need the total number of words in the corpus. You will use the count of *kör*, 7 200 000, with the (very rough) estimation that it represents $\frac{1}{3}$‰ ($\frac{1}{3}$ per thousand) of the corpus.

Chapter 4. Describe the Laplace method to estimate a bigram probability and apply it to the sentence. 2 points

Table 1: Word and bigram counts retrieved from Bing.com on October 16, 2012 with the search limited to Sweden.

Words	**Unigram counts**	**Bigrams**	**Bigram counts**
stena	458 000		
line	6 110 000	stena line	126 000
kör	7 210 000	line kör	268
därmed	5 200 000	kör därmed	322
222	1 750 000	därmed 222	1
linjer	884 000	222 linjer	0

Chapter 4. Describe a simple back off method to estimate a bigram probability and apply it to the sentence. You will use the same back off as in the labs. 2 points

Chapter 4. The back off method of the previous question is not a probability. Describe why. 2 points

Chapter 6. Using parts of speech: determiner, adjective, noun, pronoun, modal, and verb, and the sentence: *The can rusted*, annotate each word with: 4 points

1. its correct part of speech;
2. an incorrect but possible part of speech given by a lookup in a dictionary.

Chapter 6. Tell how to disambiguate the part of speech of the word *can* in the previous question. You can imagine a method using rules you would write or derived from an annotated corpus. 2 points

Chapter 8. Table 2 shows a sentence from the CoNLL 2003 corpus of English (Tjong Kim Sang and De Meulder, 2003). It contains annotations for the parts of speech, noun groups, verb groups, and prepositions (the chunks), as well as named entities. The chunk annotation in Table 2 uses the IOB tag set. It is different from the IOB2 tag set that we used in the lab. Describe in which way. 2 points

Chapter 8. Write the sentence in Table 2 and underline all the groups (chunks) with either simple, double, or triple underlines or with colors depending on their category. 2 points

Chapter 8. Write the sentence in Table 2 and underline all the named entities. 2 points

Chapter 8. In this exercise, you will write a simple named-entity detector. You will set aside the programming details and suppose you have access to the words, parts of speech, and chunk tags:

1. Using the parts of speech in the second column, write simple phrase–structure rules or regular expressions to detect the named entities in the sentence shown in Table 2. The NNP tag corresponds to proper nouns. The detection does need to be perfect. 2 points

Table 2: An excerpt from the CoNLL 2003 corpus of English (Tjong Kim Sang and De Meulder, 2003).

Words	Parts of speech	Chunks	Named entities
Famous	NNP	I-NP	O
names	NNS	I-NP	O
like	IN	I-PP	O
Tarmac	NNP	I-NP	I-ORG
Plc	NNP	I-NP	I-ORG
,	,	O	O
Costain	NNP	I-NP	I-ORG
Group	NNP	I-NP	I-ORG
Plc	NNP	I-NP	I-ORG
and	CC	I-NP	O
ARC	NNP	I-NP	I-ORG
,	,	O	O
a	DT	I-NP	O
unit	NN	I-NP	O
of	IN	I-PP	O
conglomerate	JJ	I-NP	O
Hanson	NNP	I-NP	I-ORG
Plc	NNP	I-NP	I-ORG
,	,	O	O
have	VBP	I-VP	O
all	DT	O	O
been	VBN	I-VP	O
targeted	VBN	I-VP	O
.	.	O	O

2. Give the definition of the precision and recall metrics. 2 points
3. Compute the precision and recall of your rules when applied to the sentence in Table 2. 2 points

Chapter 11. Given the rules:

```
s --> np, vp.
np --> det, noun.
vp --> verb, np.
det --> [the].
noun --> [waiter].
noun --> [meal].
verb --> [brought].
```

parse manually the sentence *The waiter brought the meal* using the shift/reduce algorithm. You will represent graphically the stack and the queue at each parsing step. 5 points

Chapter 14. Table 3 shows two sentences from the CoNLL 2011 corpus (Pradhan et al., 2011). The Chain column contains the coreference chains of entities. Give all the coreference chains appearing in these two sentences as well as their list of mentions. You will use the notation:

$$\begin{array}{lcl} CorefChain(0) & = & \{Mention1, Mention2, ...\}, \\ ... & & \\ CorefChain(x) & = & \{Mention1, Mention2, ...\} \\ ... & & \end{array}$$

5 points

Table 3: Simplified annotation of two sentences in the CoNLL 2011 corpus. After Pradhan et al. (2011).

Document	Inx	Word	POS	Parse bit	Type	Chain
wsj_0771	0	“	“	(TOP(S(S*	*	-
wsj_0771	1	Vandenberg	NNP	(NP*	(PERSON)	(8|(0)
wsj_0771	2	and	CC	*	*	-
wsj_0771	3	Rayburn	NNP	*)	(PERSON)	(23)|8)
wsj_0771	4	are	VBP	(VP*	*	-
wsj_0771	5	heroes	NNS	(NP(NP*)	*	-
wsj_0771	6	of	IN	(PP*	*	-
wsj_0771	7	mine	NN	(NP*))))	*	(15)
wsj_0771	8	,	,	*	*	-
wsj_0771	9	”	”	*)	*	-
wsj_0771	10	Mr.	NNP	(NP*	*	(15
wsj_0771	11	Boren	NNP	*)	(PERSON)	15)
wsj_0771	12	says	VBZ	(VP*	*	-
wsj_0771	13	,	,	*	*	-
wsj_0771	14	referring	VBG	(S(VP*	*	-
wsj_0771	15	as	RB	(ADVP*	*	-
wsj_0771	16	well	RB	*)	*	-
wsj_0771	17	to	IN	(PP*	*	-
wsj_0771	18	Sam	NNP	(NP(NP*	(PERSON*	(23
wsj_0771	19	Rayburn	NNP	*)	*)	-
wsj_0771	20	,	,	*	*	-
wsj_0771	21	the	DT	(NP(NP*	*	-
wsj_0771	22	Democratic	JJ	*	(NORP)	-
wsj_0771	23	House	NNP	*	(ORG)	-
wsj_0771	24	speaker	NN	*)	*	-
wsj_0771	25	who	WP	(SBAR(WHNP*)	*	-
wsj_0771	26	cooperated	VBD	(S(VP*	*	-
wsj_0771	27	with	IN	(PP*	*	-
wsj_0771	28	President	NNP	(NP*	*	-
wsj_0771	29	Eisenhower	NNP	*))))))))))	(PERSON)	23)
wsj_0771	30	.	.	*))	*	-
wsj_0771	0	“	“	(TOP(S*	*	-
wsj_0771	1	They	PRP	(NP*)	*	(8)
wsj_0771	2	allowed	VBD	(VP*	*	-
wsj_0771	3	this	DT	(S(NP*	*	(6
wsj_0771	4	country	NN	*)	*	6)
wsj_0771	5	to	TO	(VP*	*	-
wsj_0771	6	be	VB	(VP*	*	(16)
wsj_0771	7	credible	JJ	(ADJP*)))))	*	-
wsj_0771	8	.	.	*))	*	-

2 Problem

In this part, documents are allowed. It is worth 100 points.

The objective of this part is to investigate and program a system to disambiguate named entities. A complete program would link names (also called mentions) such as *Göran Persson* in this text:

> The target — announced in September by Prime Minister Göran Persson — has been met with applause from environmental organizations, but also with great skepticism from some experts who think the target is unrealistic.[1]

to entities in the form of unique identifiers, here the address (URL) of the wikipedia article on Göran Persson. Such links might be ambiguous. For example, wikipedia contains two *Göran Persson*:

1. `http://en.wikipedia.org/wiki/Göran_Persson`, a former Swedish prime minister born in 1949.
2. `http://en.wikipedia.org/wiki/Göran_Persson_(politician_b._1960)`, a Swedish politician born in 1960.

You will follow Hoffart et al. (2011)'s paper that describes such a system and you will implement parts of it in this examination. As programming language, you can use Java, Prolog, Perl, or Python. You will focus on the program structure and not on the syntactic details. You can ignore the Java packages for instance.

2.1 The Task

Read the introduction (Motivation), Sect. 1.1, and summarize it in 10 to 15 lines. 6 points

Read Sect. 3 and represent the system architecture using a picture. You will describe each module with a one to two sentence summary of their function. 8 points

2.2 Popularity Prior

You will implement the computation of the popularity priors as explained in Sect. 4.1. You will suppose that you have access to the whole wikipedia dump in your program. You will represent the encyclopedia as a list, where each list element is an article stored as a single string. The length of the list will then correspond to the number of articles. In Java, use this declaration:

```
List<String> wikipedia;
```

where the content of `wikipedia` is filled by another program that is not part of this examination.

This exercise requires regular expressions. Should you use Java as programming language, you can use Perl regexes as in the textbook and embed them in your Java code. You should then mark clearly when you use Perl code and pass the results to the rest of the Java program. You can also use Java regexes if you know their syntax.

[1]Retrieved from `nbcnews.com`, 2/7/2006. The correct spelling *Göran* has been restored from the name *Goran* as it shows in the original text.

2.2.1 Counting links

Links in wikipedia (wikilinks) relate a string like *Kashmir* showing in blue in a web browser to another page, here the wikipedia article on Kashmir. Wikipedia links in the source code are enclosed in doubled squared brackets, The link to the Kashmir article will be encoded as `[[Kashmir]]`. The source text:

```
This article covers the history of [[Kashmir]] from earliest
recorded times to the present day.
```

is rendered as

> This article covers the history of **Kashmir** from earliest recorded times to the present day.

by a browser, where clicking on Kashmir will move you to the page: `http://en.wikipedia.org/wiki/Kashmir`.

Links may consist of two parts, the link and the label, when the page name – the link – is different from what shows in the originating page – the label. Its syntax is `[[link|label]]`.

The source code

```
... Led Zeppelin's [[Kashmir (song)|Kashmir]] and was included
in the soundtrack of [[Godzilla (1998 film)|Godzilla]].
```

is rendered as:

> ... Led Zeppelin's **Kashmir** and was included in the soundtrack of **Godzilla**.

with two links, Kashmir and Godzilla, where clicking on the word *Kashmir* will move you to the page: `http://en.wikipedia.org/wiki/Kashmir_(song)`.

1. Write a regular expression that matches `[[link]]` patterns and stores the value of the link; 5 points
2. Write a regular expression that matches `[[link|label]]` patterns and stores the values of the link and of the label; 8 points
3. Propose a data structure that will hold all the labels contained in Wikipedia and for each label, the list of wikilinks and their counts. This means that given the two labels, *Kashmir* and *Göran Persson*, you will design a structure that holds them and for Kashmir (resp. Göran Persson), your structure will hold the counts of Kashmir (The province of Pakistan/India) and the counts of Kashmir (song), the song (resp. the counts of Göran Persson, prime minister and the counts of Göran Persson born in 1960). 5 points
4. Write a function that extracts all the wikilink labels from your corpus; 8 points
5. Extend your function so that it counts all the links and labels and produces a breakdown. Should you have only two labels, the output could look like: 20 points

- Kashmir: 120 occurrences divided into:
  - Kashmir: 110 occurrences

    - Kashmir (song): 10 occurrences.
- Göran Persson: 30 occurrences divided into:
    - Göran Persson: 28 occurrences
    - Göran Persson (politician b. 1960): 2 occurrences.
- ...

2.3 Syntax-based Similarity

Read the section *Syntax-based Similarity* of the article. You will implement a program corresponding to the first paragraph. You will extract subject-verb pairs from a parsed corpus, and for each verb, you will output the count of extracted subjects ranked by frequency. Given the four sentences:

> Page played unusual cords. The guitarist played a few notes. Page played with Donovan "Mellow Yellow." Page began his career as a studio session guitarist.

your system will output:
played:

- Page, 2 occurrences
- guitarist, 1 occurrence

began:

- Page, 1 occurrence

1. Describe the steps you would follow to parse wikipedia using a dependency parser. Once parsed, we suppose the result is in the form of a single file using the CoNLL format as in Table 4. 5 points
2. Write a program to extract all the subject-verb pairs from the parsed corpus, and for each verb, output the count of extracted subjects ranked by frequency.
    - Describe a data structure to store all the verbs in the corpus and for each verb, all its subjects with their counts as described in the first paragraph of this section. 5 points
    - Extract the verb subject-pairs and for each verb give the subject counts by order of frequency. You will suppose you have access to the corpus using this variable:

      ```
      List<List<Word>> sentenceList;
      ```

      and that `Word` contains all the columns of the CoNLL format. You will extract the fields using `getForm()` to get the word value, `getPOS()`, to get the part of speech, `getHead()`, to get the head index, and `getDeprel()` to get the grammatical function. The subject function in the English version of the dependency corpus is `SBJ`. Verbs have the parts of speech: `VB`, `VBZ`, `VBD`, `VBP`, `VBN`, and `VBG`. 15 points
3. Write a program to extract all the subject-verb pairs from the parsed corpus, and for each subject, output the count of extracted verbs ranked by frequency 15 points

Table 4: The CoNLL 2007 format used to represent parsed sentences.

1	Ms.	_	NN	NNP	_	2	NMOD	_	_
2	Haag	_	NN	NNP	_	3	SBJ	_	_
3	plays	_	VB	VBZ	_	0	ROOT	_	_
4	Elianti	_	NN	NNP	_	3	OBJ	_	_
5	.	_	.	.	_	3	P	_	_

References

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., and Xue, N. (2011). CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA. Association for Computational Linguistics.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147, Edmonton.