EDAN20
Final Examination

Pierre Nugues

October 24, 2013

The examination is worth 188 points. The distribution of points is indicated with the questions. You need 55% to have a mark of 4 and 70% to have a 5.

1 Closed Book Part: Questions

In this part, no document is allowed. It is worth 90 points.

Chapter 1. Cite three applications that use natural language processing to some extent. 3 points

Chapter 1. Annotate each word of the sentence *The mice have eaten the cheese* with its lemma and part of speech. 4 points

Chapter 1. Draw the parse trees/graphs of the sentence *The mice ate the cheese* using constituents and dependencies. 4 points

Chapter 1. What is the CoNLL format? 2 points

Chapter 2. Describe what a concordance is and give all the concordances of the string *nomine*: 3 points

> De kan vinna Augustpriset
> Kultur & Nöjen. Per Olov Enquist kan få Augustpriset för tredje gången. Det stod klart när nomineringarna presenterades på måndagen. Men Jonas Gardell blev utan nominering för sin romansvit "Torka aldrig tårar".
>
> Per Olov Enquist nomineras för "Liknelseboken". Författaren tog hem priset såväl 1999 som 2008 med "Livläkarens besök" respektive "Ett annat liv". De övriga nominerade i skönlitterära klassen är Lena Andersson med "Egenmäktigt förfarande - en roman om kärlek", Sven Olov Karlssons "Porslinsfasaderna" och "Hägring 38" av Kjell Westö samt de två poeterna Athena Farrokhzad med "Vitsvit" och Katarina Frostenson med "Tre vägar"
>
> *Sydsvenskan.se*, Retrieved October 21, 2013.

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

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

1. `ring␣*`
2. `ring␣?`
3. `ring␣+`
4. `\w{17}`
5. `\"\w[aeiou]`
6. `\d+\"`
7. `\d+`
8. `\d*\"\.`
9. `(\")\w+\1`
10. `^[A-Z]+[a-z]{3,}`

Chapter 2. Write a regular expression that extracts all the person's names in the text above. Try to be generic so that it could be applied to other texts. 4 points

Chapter 2. Let us imagine that we want to extract the two first columns of a CoNLL file that consists of three columns separated with a tabulation: `\t`. Write a regular expression that matches the contents of the two first columns of a line and stores them in two regex variables (stores). You can decide to match or not to match the third column. 3 points

Chapter 2. What does this Unix command do?
`sort -nr <file.txt` 1 point

Chapter 2. Edit distance is sometimes used to measure the distance between cognates, terms having a common origin. Use the three operations, copy, delete, and insert to transform the Swedish term *obegränsad* into the German *unbegrenzt*, and the French term *illimité* into the English *unlimited*. You will use the following codes for the operations: c for copy, d for delete, and i(letter) for insert a letter. Represent graphically the corresponding alignments. 6 points

Chapter 3. Given a set with two classes, P and N, with the respective frequencies $C(P)$ and $C(N)$, give the entropy of the set. 3 points

Chapter 4. Give the probabilistic model of the sentence *Enquist kan få Augustpriset för tredje gången* using no n-gram approximation (chain rule). 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 *Enquist kan få Augustpriset för tredje gången*. 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 *Enquist kan få Augustpriset för tredje gången.* 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 *Pierre kan missa Augustpriset för tredje gången* 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 *för* with the estimation that it represents 1.5% of the corpus.

Table 1: Word and bigram counts retrieved from Bing.com on October 22, 2013 with the search limited to Sweden (site:se).

Words	**Unigram counts**	**Bigrams**	**Bigram counts**
pierre	164 000		
kan	15 700 000	pierre kan	2 580
missa	479 000	kan missa	41 500
augustpriset	41 000	missa augustpriset	0
för	20 500 000	augustpriset för	4 480
tredje	587 000	för tredje	65 200
gången	1 180 000	tredje gången	54 500

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. 4 points

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

Chapter 4. Give the definition of the mutual information association measure. 2 points

Chapter 4. Describe what the vector space model is and represent the two following sentences using binary vectors (vectors consisting of zeros and ones). 4 points

D1: Chrysler plans new investments in Latin America.

D2: Chrysler plans major investments in Mexico.

Chapter 6. Table 2 shows a sentence from the CoNLL 2000 corpus of English (Tjong Kim Sang and Buchholz, 2000). Using the parts of speech: det(erminer), adv(erb), adj(ective), noun, pre(position), pro(noun), and verb: 4 points

1. Annotate each word with its correct part of speech;
2. Give two words in this sentence that have an ambiguous part of speech.

Chapter 6. Describe two simple rules to disambiguate these two words. 2 points

Chapter 8. Table 2 contains annotations for the parts of speech and the groups (chunks). 8 points

1. Underline the noun groups and the verb groups. Use different colors or styles for each group.
2. Use IOB2 tags to tag the verb groups and the noun groups.
3. Write phrase-structure rules using the parts of speech det(erminer), adv(erb), adj(ective), noun, pre(position), pro(noun), and verb to extract the verb groups and the noun groups.

Table 2: An excerpt from the CoNLL 2000 corpus of English (Tjong Kim Sang and Buchholz, 2000).

Words	**Parts of speech**	**Chunks**
At		
the		
same		
time		
,		
he		
remains		
fairly		
pessimistic		
about		
the		
outlook		
for		
imports		

Chapter 11. Given the sentence *The mice ate the cheese*, extract a triple consisting of the subject, the verb, and the object. 1 point

Chapter 11. Nivre's parser uses two data structures and four parsing actions, left-arc, right-arc, shift, and reduce. Give the name of these data structures and the definition of the four actions. 3 points

Chapter 11. Parse manually the sentence *The mice ate the cheese* using the two data structures and the four parsing actions. For each action, you will represent graphically the content of the two data structures and the graph being constructed. You will need from 8 to 10 steps. 7 points

2 Problem

In this part, documents are allowed. It is worth 98 points.

The objective of this part is to investigate and program a system to automatically create categories of things (classes), like birds, birds of Africa, birds of Europe, birds of America, etc. from a corpus.

You will follow Paşca (2013)'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.

You will also ignore the case, all the words are supposed to be in lowercase letters, and the punctuation signs.

2.1 The Task

2.1.1 Outline

8 points

Read the introduction and summarize it in 10 to 15 lines.

2.1.2 Summary of the Steps

10 points

Read Sect. 2 and summarize the different steps of the system for the:

1. Extraction of class labels;
2. Extraction of instances.

2.2 Extraction of Class Labels

2.2.1 Initial Vocabulary

8 points

You will implement the extraction of the initial vocabulary as explained in Paşca (2013, Sect. 2.1). Each element of this vocabulary may consist of multiple tokens. You will suppose that you have access to the query corpus in your program in the form of a list of strings; each string corresponding to a single query. In Java, use this declaration:

```
List<String> queries;
```

Following the description in the article, write a program that produces a map of class labels:

```
Map<String, Integer> classLabels;
```

where each `classLabel` contains an element of the initial vocabulary, a string, and its count in the query corpus.

2.2.2 Generation via Phrase Similarities

In this exercise, you will represent the web as a big string consisting of words separated by spaces:

```
String web;
```

You will ignore the punctuation signs.

To compute the phrase similarities, you will follow the vector space model and you will represent each phrase by a vector. Each dimension of the vector space will correspond to a word in the corpus. For each phrase, you will set to zero the vector parameters representing words not in this phrase.

1. Write a program that extracts all the words (unigrams) and their counts from the web corpus. In Java, use the declaration: 4 points

   ```
   Map<String, Integer> unigrams;
   ```

2. Write a program that extracts all the bigrams and their counts from the corpus. In Java, use the declaration: 8 points

   ```
   Map<String, Integer> bigrams;
   ```

3. Given the unigram *companies*, the bigram *software companies*, and this short text simplified from `https://en.wikipedia.org/wiki/Software_industry`:

   > Business models of software companies have been widely discussed. Network effects in software ecosystems are an important element in the strategy of software companies.

   (Please, ignore the case and the punctuation.)

   - Write manually the vector representing the complete text using the vector space model; use the alphabetical order to mark the positions and the counts; 3 points
   - Write manually the vector representing the contextual features of the unigram as described in Paşca (2013, Sect. 2.1); however, you will use the word counts to represent the context words and as window, you will use one word to the left of the unigram and one word to the right of it; 3 points
   - Write manually the vector representing the contextual features of the bigram; you will use counts to represent the context words and as window, you will use one word to the left of the bigram and one word to the right of it (if any); 3 points

4. Write a program that produces a feature vector for each unigram in the web corpus. In Java, you can store the map: 20 points

   ```
   Map<String, Integer[]> unigramContext;
   ```

   consisting of a string and an array representing the string context (a vector containing frequencies).

5. Outline how you would compute the feature vectors for the bigrams and other n-grams. No program is needed. 5 points

6. Outline how you would write a program to compute the pointwise mutual information[1] between an n-gram and a context word w: 6 points

$$I(n\text{-gram}, w) = \log_2 \frac{P(n\text{-gram}, w)}{P(n\text{-gram})P(w)}.$$

No programming is needed in this exercise!

Note that even if an n-gram contains more than one word, you will consider it as a single word.

You will suppose that you have a map that contains the n-grams and their counts in the map:

```
Map<String, Integer[]> ngramContext;
```

and your program would create a map of context that contains a vector of pointwise mutual information instead of counts:

```
Map<String, Double[]> ngramContextMI;
```

7. Following Paşca (2013, Sect. 2.1), write a program that computes the context similarity of two n-grams, $n1$ and $n2$, using the cosine: 4 points

$$\cos(\vec{n1}, \vec{n2}) = \frac{\sum_{i=1}^{n} n1_i n2_i}{\sqrt{\sum_{i=1}^{n} n1_i^2} \sqrt{\sum_{i=1}^{n} n2_i^2}}.$$

8. For each n-gram, find its most similar n-gram using the context. In Java, you can use: 6 points

```
Map<String, String> similarNgram;
```

9. Write a program that generates phrases similar to the class labels. For each element in: 6 points

```
Map<String, Integer> classLabels;
```

generate phrases similar to the string. Your program will replace one word (unigram) of the original class label with its most most similar n-gram from `similarNgram`. In Java, use the map:

```
Map<String, List<String>> similarQueries;
```

to store the result. You will generate all the phrases similar to a class label, where a generated phrase and the original class label will differ by only one substitution.

10. In what the last program is different from the procedure described in the article? 4 points

2.2.3 Syntactic Filtering

This question is optional and will give you extra points

You will suppose that you have a part-of-speech tagger that you have applied to the strings in the map:

```
Map<String, List<String>> similarQueries;
```

1. Propose a data structure to hold the results of the tagging;
2. Write a program that applies the syntactic filtering described in the article.

2.2.4 Query Filtering

This question is optional and will give you extra points

Using the original list of queries:

```
List<String> queries;
```

write a program that implements the query filtering described in the article.

2.3 Extraction of Instances

Should you have the time, you can start programming the extraction of instances. You will be given extra points for this.

References

Paşca, M. (2013). Open-domain fine-grained class extraction from web search queries. In *EMNLP*, pages 403–414.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon.

[1] In the textbook, *pointwise mutual information* is called *mutual information*.