Information extraction for classified advertisements

David FAURE
david.faure.401@student.lu.se

Claire MORLON
claire.morlon.612@student.lu.se

1. Abstract

This report describes our work on the project part of the course Language Processing and Computational Linguistics. It presents a java written program that extracts six important pieces of information from French job advertisements. The inputs of the system are advertisements taken from the internet and converted as text files.
The results presented in this paper show that the extraction mechanism is reliable and robust.

2. Introduction

Everyone entering the job market knows how time consuming the search for a job is. The main contribution of this waste of time is the time spent while reading the advertisements in order to see if the proposed job can fit with one's competences and requirements. From this observation, we guess how useful a program that extracts automatically the interesting information from these job advertisements could be.
This paper will present a java written program whose role is to extract six pieces of information from some internship advertisements found on the Internet. The six characteristics of interest for the internships are: its subject (`subject`), its duration (`duration`), the required study level (`studylevel`), the company (`firm`) and the place (`city`) where it takes place and finally the date when it starts (`beginning`).

This paper is organized as follows: After this short introduction, section 3 presents the source texts that can be used with the program, section 4 deals with the functioning of the written java program. The results obtained with this program as well as its evaluation are presented in section 5.

3. Source texts

The program was developed to extract information from French advertisements. The advertisements used throughout the project were taken from the internet: at first, we took them on some companies' web sites. As some information were not explicit in this case (the name of the company was for example supposed to be known), we finally chose to work with "general" web sites for job advertisements. The advertisements, chosen such that they had at least two pieces of interesting information, where then converted into text files (.txt) in order to get rid of all web formats (tables, headers …). These files were finally used as inputs for the java program.
We used 15 advertisements as a working set during the development of the program. We then applied the final version on 14 other unread texts in order to evaluate the program more objectively, and to measure how independent from the texts the results were. All these 29 texts were hand marked to make easier the comparison between the theoretical words of interest and those found by the program.

4. The information extraction program

1. Mechanism

The information extraction mechanism in our program works in two steps:

- **Division of the text into blocks**

The first task consists in dividing the text into smaller blocks corresponding to the different pieces of information (subject, beginning…), to make the search of useful information easier and quicker.

In order to do that, we first look for keywords in the text, which will delimit the different blocks. The program runs through a list of keywords, and performs pattern matching for each of them.

For instance, let's consider a text composed by the sentence:

> *Nous recherchons deux futurs ingénieurs pour une durée de <duration>6 mois</duration> sur notre site de production dans le centre ville de <city>Quimper</city>.*

Three key words are found in this text : *durée* (duration), *site* (site) and *ville* (city). The first one will thus introduce a block where there is a high probability to find an information of duration. The two other ones are both related to the `city` information; they will thus delimit two blocks corresponding to `city`. The blocks obtained for this example are finally:

> *1. durée de 6 mois sur notre*
> *2. site de production dans le centre*
> *3. ville de Quimper.*

Since several keywords can be found for the same piece of information (such as `city` in the example), all the blocks corresponding to a same information are stored into a table. Finally, we store all these tables into a hashtable whose keys are the different kind of information that are looked for. The final hashtable corresponding to the example is represented in Table 1.

information	blocks
`duration`	*durée de 6 mois sur notre*
`city`	*site de production dans le centre*
	ville de Quimper

Table 1: Hashtable listing the different blocks found in the example and the pieces of information they are related to

The idea behind this system is to try to isolate the relevant information to avoid searching the whole text. But it is also a good way to chose the better solutions in a list of several propositions, which leads to a better accuracy. For example, if several names of cities are present in the whole text, a global search would normally detect all of them. But then, how to choose the one where the internship really takes place? With our solution, only the names detected in the blocks related to `city` will be kept, as it's very likely that the relevant city is mentioned in those blocks.

- **Pattern matching**

Secondly, the useful information has to be detected in the relevant blocks. This is done with pattern matching. The first step of this stage consists in removing the keywords at the very beginning of the blocks. Then, for each piece of information, we look for patterns inside the corresponding block(s). If one pattern matches, we keep it and store it in a hashtable. The next section presents some of the patterns used in the program. If no pattern is found in any of the blocks, the same pattern matching is applied in the whole text, in case the block division would be inappropriate. It's however good to

keep in mind that the patterns found in the blocks give in general a better accuracy than the ones found after a search in the whole text.
Since several patterns can be found for one piece of information, only the first matched pattern is stored in the hashtable. Indeed, the useful and relevant information are often present at the beginning of the advertisement.

The resulting hashtable corresponding to the example is represented in Table 2.

Information	Result
`duration`	*6 mois*
`city`	*Quimper*

Table 2: Hashtable listing the final results for the example

2. Some patterns used

Here are some examples of the patterns we used to find the relevant information in the blocks or in the text.

- In order to find the location of the internship, we used an alphabetical list of all the French cities. The program runs through this list and for each city, looks in the block or in the text if it can find it.
- To find the `beginning` information, one of the pattern we used was "*month 200[0-9]"* where *month* was one of the twelve months of the year.
- For the `studylevel` information, we took advantage of the French system of qualification which has the type {*bac +number between 0 and 7}* (Ex.: *bac +5*). We thus used as pattern : *"bac.{1,5}[0-9]"*
- We noticed that the `subject` information corresponds often to a complete sentence introduced by a `subject` keyword (*"sujet : ", "intitulé : ",* …). Therefore, we took the first sentence of the `subject` block if there is such a block, or the first sentence of the whole text in the opposite case.
- To find the `duration` information, we used the pattern *"[0-9].mois"* (where "mois" means month in French).
- To find the name of the company, we took the first line of the firm block, the name included in the email address (pattern: *"@(.*?)\\."*), or in the web address (pattern: *"(www\\.)(.+)(\\.com|\\.fr)"*).

5. Results

In order to have a quantitative evaluation of the performance of our program, we computed two parameters (precision and recall) for each of the six desired pieces of information.
The precision gives an evaluation of the correctness of the proposed instances; it is defined by:

$$\text{Precision} = \frac{\text{nbr of instances proposed correctly}}{\text{nbr of proposed instances}}$$

The recall measures how well the program finds what was expected; it is defined by:

$$\text{Recall} = \frac{\text{nbr of instances proposed correctly}}{\text{nbr of instances possible to find}}$$

1. Results for the working set of texts

The precision and recall average values for each piece of information obtained with the 15 texts of the working set are presented on Fig. 1 and Fig. 2.
We can notice that the `city` and the `firm` information get good results (precisions over 80% and largest recalls). For the firm information, this can be explained by the fact that the name of the company is often well isolated in the advertisement and introduced by regular keywords; thus our system of blocks works quite well for this

information. For the city information, this shows that the use of the city list is very efficient.

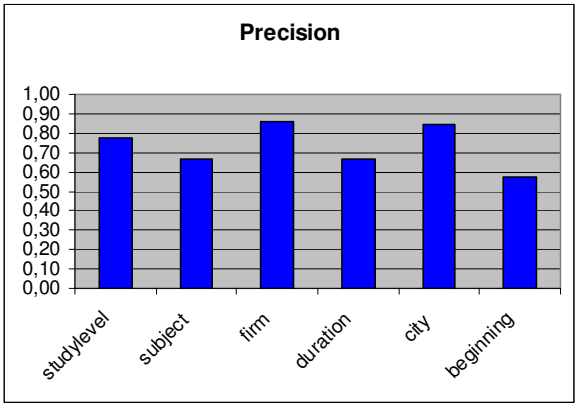

Fig. 1: Precisions obtained with the 15 working texts for the 6 parameters of interest

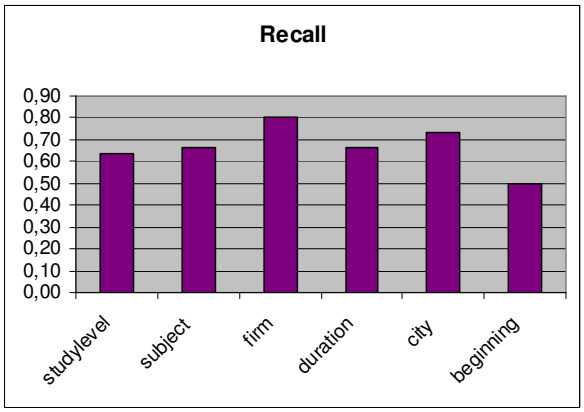

Fig. 2: Recalls obtained with the 15 working texts for the 6 parameters of interest

The other parameters get less good results mainly because of the difficulty to isolate and delimit very precisely the correct answers. Indeed, when we have a look to the proposed instances, we notice that the program sometimes proposes either only a part of the correct answer or a bit more than it.

Fig. 3 and Fig. 4 represent two lightly modified versions of the precision in order to quantify more precisely the amount of "inexact" answers.

In the case of precision 1, an answer is considered as correct if the instance proposed by the program contains the theoretical value. We see on the graph that only the subject information is concerned by this case. Indeed, it's easy to find the beginning of the `subject` (with keywords) but difficult to know where it ends. Consequently, the program sometimes takes more words that needed.

At the opposite, precision 2 is obtained by considering an answer as correct if the instance proposed by the program is included in the theoretical value. In this case, the information `studylevel`, `duration` and `beginning` are concerned, especially because the pattern matching system is too rigid.

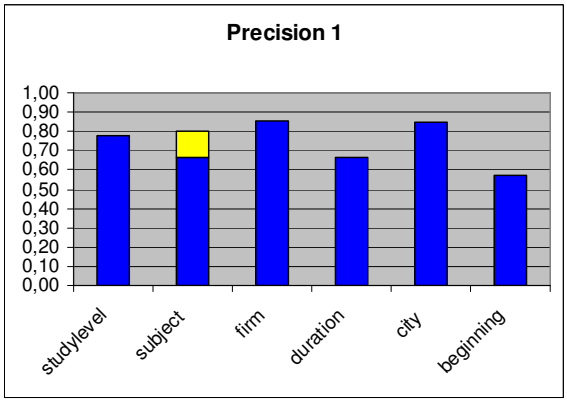

Fig. 3: Precisions obtained if "unprecise" solutions are accepted

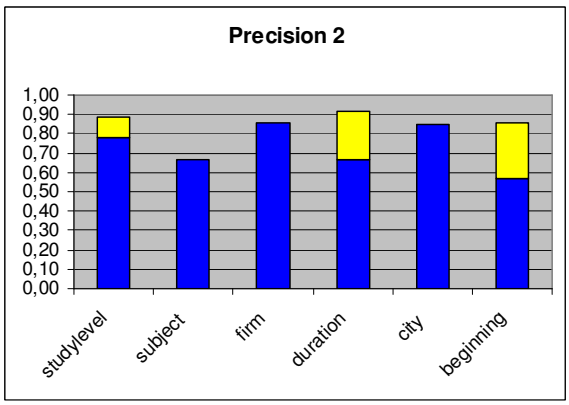

Fig. 4: Precisions obtained if partial solutions are accepted

For instance, for the `beginning` information, the program finds "janvier 2006" when the expected answer was "fin janvier 2006". Likewise, the study levels Bac +4/+5 are partially detected (we only obtain Bac +4). However, a more precise pattern matching system which could have improved the results and detected those kind of missed words would have be too "hard wired". It was indeed very difficult

to find general rules that could have taken into account those exceptional cases.

2. Results with new texts

When applied on unread texts, the program gives the results on Fig.5 and Fig. 6. The results obtained with the new texts are plotted in green on the graphs, while those considered previously are still in blue.

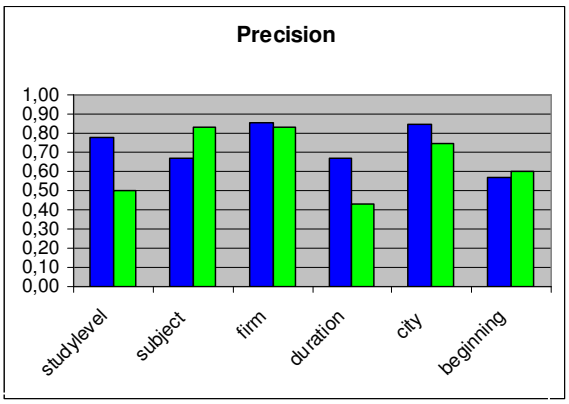

Fig. 5: Precisions obtained with the working texts (blue) and the unread texts (green) for the 6 parameters of interest

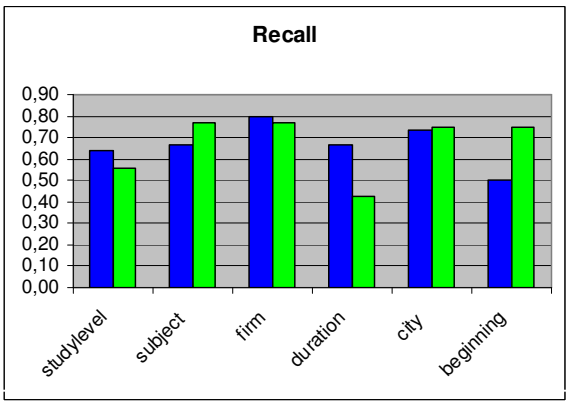

Fig. 6: Recalls obtained with the working texts (blue) and the unread texts (green) for the 6 parameters of interest

We can notice that the results are very similar, at least on average: some pieces of information obtain larger precision and recall with the working texts, whereas others have better results with the new texts. This shows that the results are quite independent on the texts, and this can prove than our program is robust enough to be applied on unknown texts.

The small differences in the results for the two sets of texts are probably due to the fact that we used in fact a small amount of texts, for both development and evaluation. The results would consequently be much more reliable if a larger amount of advertisements was used.

6. Conclusion

According to the results, the program that we implemented is reliable enough to get a good overview of a French job advertisement. However, all the useful pieces of information are not always found, especially because each advertisement has its own format. It's therefore very difficult to establish general rules for pattern matching. Some improvements, such as taking advantage of the HTML format (exploiting the tables to delimit easier the blocks, using headers to find the most important information such as subject and company,…), could have been done if we had more time.

References

Hugo Etiévant 2004. *Expressions régulières en Java avec l'API Regex* http://cyberzoide.developpez.com