Web Search, Renmin University of China

Lab 3: programmers

Fabian Suchanek

15 July 2011

Labs can be made individually or by groups of two. Make sure to send by email to `pierre@senellart.com`, either at the end of the lab session or at the latest at midnight of the same day, an informal report about what you did in the lab, and the results you obtained. You do not have to finish all exercises, if you advance reasonably during the lab session but do not go to the end of the assignment, you will still get a passing grade.

The purpose of this lab session is to extract structured information from a natural language text corpus.

Data Set

As in the other labs, our corpus will be the Simple English Wikipedia (`http://simple.wikipedia.org/`), a simpler and smaller encyclopedia than the regular English Wikipedia). The whole content of the encyclopedia can be downloaded from `http://dumps.wikimedia.org/`. For these labs, we provide the data in the following forms (on the course Web site):

- A preprocessed version containing only the text. The format of this file is as follows: The first line is the title of the first article, while the following lines (up to the first blank line) form the content of this article, in plain text format. The second article comes after the next blank line, and so on. There are 50,441 articles in total.
- A preprocessed version containing only the text, with POS annotations. The format of this file is as above, but each word is followed by "_XXX", where XXX identifies the POS, as explained in the lecture.
- The full version, including the Wiki markup. The format of the file is as explained in the lecture.

1 Information Extraction

For all of the following, go for precision rather than recall.

1. We first need a class `Page`, which has two `String` fields: the page title and the page content. In this class, as well as in all other classes, remember to implement the standard methods `toString()`, `hashCode()`, `equals()` and potentially `compareTo()`. Write a class `Parser`, which has a constructor that takes as argument the filename of the corpus. The class should have a method `next():Page`, which delivers the next page in the corpus. Remember to close the file at the end. If you want, the class `Parser` can implement the interfaces `Closeable` and `Iterator<Page>`. Implement this class for the preprocessed text-only corpus. Create a small sample corpus of two or three articles and test your class.

2. Create a class `Triple`, which has three `String` fields: `subject`, `predicate` and `object`. Create an abstract class `Extractor`, which has a method `extract(Page):Collection<Triple>`. Create a simple extractor, `NameExtractor`, which produces triples of the form `<PageTitle, hasName,"PageTitle">`. Create a class `InformationExtractor`, which has only one method: `run`. This method takes as input a corpus file, a target file and a list of extractors. It iterates over all pages in the corpus, calls all extractors and writes the triples, TAB-separated, into the target file. Test this method with the `NameExtractor` on the small sample corpus.

3. Write an extractor `DateExtractor` that uses a regular expression to find the first date mentioned in the article. Let it return a triple of the form `<PageTitle, hasDate, Date>`. Try the extractor with the pages of Elvis Presley and Alan Turing. If you are adventurous, try normalizing the dates you extract to the form `[-]YYYY-MM-DD`. Regular expressions work as follows in Java:

```
Pattern pattern=Pattern.compile(EXPRESSION);
Matcher matcher=pattern.matcher(STRING);
while(matcher.find()) {
// matcher.group(N) holds the N-th group of the match
// matcher.group() holds the entire match
}
```

   Use named regular expressions.

4. Write an extractor `TypeExtractor` that extracts the type of the article entity ("Leicester is a city"). Manually exclude terms that are too abstract to be an ontological type ("member of…", "way of…"). If you are adventurous, improve your type extractor by running it on the POS-tagged corpus. For this purpose, extend the class `Page` and write a new `TypeExtractor`. Write an extractor `LocationExtractor` that extracts the location of a place ("Hollywood is a district in Los Angeles"). Alternatively or additionally: Write a `TypeAndLocationExtractor`, which first calls the `TypeExtractor`, checks if the article entity is a city, district, etc., and, if so, extracts the location.

2 To go further

1. Extend your information retrieval system. Whenever a document contains an article entity X, and if you know that <X, locatedIn, Y>, add an artificial word to the index "placeInY". See whether you can find "Abdou Diouf" if you search for "president placeInWestAfrica".

2. Write extractors that work on the full version of the simple Wikipedia. In this version, you can use the infoboxes to extract birth dates (e.g., for Elvis Presley), or population numbers (e.g., for Paris).