Informatics 1B
Data and Analysis: Lab Session 3
Introduction to CQP (Corpus Query Processor)

Manuel Marques Pita and Frank Keller

February 16, 2007

1 Introduction

This lab session is largely based on the CQP tutorial developed by Stefan Evert available at `http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQPTutorial/html/`

2 Setting up the working environment

Before you get started in this lab session, the first thing you need to do is to tell your system where to locate the *corpus* you will be using today (which consists of a number of stories written by Charles Dickens). In order to do this, type the following command on a terminal window

```
export CORPUS_REGISTRY=/group/corpora/public/corpus_workbench/registry
```

Now you can get `cqp` started by typing simply `cqp`

You will then see the following in your command prompt

```
[no corpus] >
```

This means that `cqp` is active but no specific *corpus* has been selected. To select the `DICKENS` corpus, type

```
[no corpus] > DICKENS;
```

You will then see the following

```
DICKENS >
```

From this point you can get some more specific information about this particular corpus by typing

`DICKENS > info;`

Press the `space bar` to display the next page and `q` to get back to the command prompt. Please take some time to read this information and make some notes of what you think is relevant and/or related to anything you have seen in the lectures so far.

3 Searching for words

Now that the working environment has been set, we can start performing some queries on the DICKENS corpus. The most basic type of token we may want to look for is a simple word. To search for a specific word (for example the word "dear") you need to type

`DICKENS > "dear";` (again, press `space` for next page, `q` for coming back to command prompt)

Question 1. What would be the use of a query like this, providing the results (word in context) given by `cqp`? Think for example about what the use of this word (or other words) may help understand information in the corpus. Also think how this would help a writer when writing a story!

Add your answer in the space below

Sometimes we will want to search for words that are very similar, so that we can get a better sense of context, for instance think about a query like the previous one, but returning all the instances in context of some words starting such as *regular* such as *regulars, regularly* and *regularity.* We can do this in one query, by typing

`DICKENS > "regular(s|ly|ity)";` (Try it!)

The syntax of the query above is based on the use of a Regular Expression, like the ones you studied last semester.

Question 2. How can we interpret or paraphrase this query in plain English?

In some cases, you may want to search for the given pattern ignoring if the letters are in lowercase or uppercase. In order to make your search case insensitive you need only append a %c before your semicolon, like this:

`DICKENS > "dear" %c;` (Try it!)

4 Display Options

When working with `cqp` you will want to customise how the results of your queries are displayed to suit your needs. Perhaps the most important thing we may want to customise is for a query result to display (or not display) the annotations made on the corpus. This is easily achieved by running the following:

`DICKENS > show +pos +lemma;` (If you want query results to *show* annotations) or

`DICKENS > show -pos -lemma;` (If you want query results to *hide* annotations)

By default, this switch is off. Turn it on, and try one or two queries so you can see how this corpus has been annotated.

Question 3. Can you determine what kinds of tokens have been annotated? (Find at least two!)

5 Accessing linguistic annotations

One of the reasons we annotate text data is to enable us with the possibility of searching tokens annotated with some determined tag. For example, we might be interested in searching all *adjectives, verbs* or *nouns* in a given corpus. In order to do that, we need to know the actual set of tags used to annotate the corpus. In `cqp` you can see this by running the `show cd` command. To understand what some of these tags actually mean you can access the following website

`http://www.mozart-oz.org/mogul/doc/lager/brill-tagger/penn.html`

If you want to find out what *adjectives* have been annotated in the DICKENS corpus, you need to run the following query:

`[pos = "JJ"];` (JJ corresponds to adjectives, Try this!.)

Notice (in the web page mentioned earlier) that different types of *noun* are annotated. You can use regular expressions in `cqp` to find a given pattern of tag.

Question 4. How can we retrieve all noun tokens in a `cqp` query?

Question 5. How can we retrieve all tokens of any type except noun in a `cqp` query? (Hint, try using != instead of = in your query...)

Question 6. With your knowledge of regular expressions (*regexp*), how can you design a query that retrieves all instances of *the* and *The*? Remember that in a regexp, the character "." will match *any* character and that we can define alternatives for a character in square brackets. For example a regexp `["Marque[sz]"]` will retrieve all instances of *Marques* and *Marquez*.

6 Sequences of words

Now we will recall a few other operators used in Regular Expressions. Whenever a (*) is added to a regexp, the preceding expression will be matched zero or more times. a (+) sign works in a similar way but will match the preceding expression once or more, and finally (?) will match zero or once only.

Question 7. Using all the operators described so far, how can we build a `cqp` query to retrieve all prepositions (tagged as `IN`) followed by a determiner (tagged as `DT`)? Try different options and describe the differences between them.

7 Named query results

Question 8a. Now we will work with a series of incrementally more complex queries. First of all write a query that finds all the instances of the word *brother* or *sister*. This should be easy. Now what we want is to *save* that query. In order to do that please run the following

`DICKENS > Siblings01 =` *add your 8a query here, followed by* `[];`

This is going to assign the results of your query to the variable Siblings01 (it must start with uppercase letter!). When you do this, the next thing you will see is the corpus prompt again. How many tokens were returned by your query?

```
DICKENS > size Siblings01
```

This should be 1724.

Question 8b. Our next query wants to find all the instances of the words *Brother*, *brother*, *Sister* and *sister* and save it in Siblings02 (use regular expressions). What is the syntax of the query? How many tokens have been retrieved this time?

Question 8c. Create a Siblings03 query in which you retrieve all the same tokens as in Siblings02 but including all the plural cases for each of the words in Siblings02 (use regular expressions). What is the syntax of the query? How many items are there in this new query?