Inf1B Data and Analysis
Tutorial 6 (week 8)

26 February 2010

- Please answer all questions on this worksheet in advance of the tutorial, and bring with you all work, including printouts of code and other results. Tutorials cannot function properly unless you do the work in advance.
- Data & Analysis tutorial exercises are not assessed, but are a compulsory and important part of the course. If you do not do the exercises then you are unlikely to pass the exam.
- Attendance at tutorials is obligatory; please let your tutor know if you cannot attend.
- *Recommended Reading: Chapter 2 of* `Corpus Linguistics` *by T. McEnery and A. Wilson until the end of section 2.2.2*

Introduction

In this tutorial we will familiarise ourselves with a real corpus and learn how to query it with the use of CQP (Corpus Query Processor), a query engine that searches corpora based on user queries over words, parts of speech, or other markup.
This tutorial is largely based on the CQP tutorial developed by Stephan Evert available at
`http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQPTutorial/cqp-tutorial.pdf`

Setting up the working environment

This tutorial needs to be carried out on a DICE machine in the Informatics computer labs. 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, please type the following command on a terminal window:

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

Please be very careful formulating the export instruction, as it is case sensitive and there are two underscores. Now you can get `cqp` started by typing '`cqp -e`'. 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 yet. In order 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. Furthermore, typing `exit;` will allow you to quit the `cqp`-program. 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.

1 Concordances

Concordance of a given word.
Now that the working environment has been set, we can start performing some queries on the DICKENS corpus. Searching for concordances (i.e. all occurrences of a given word, displayed in context) is a basic form of querying a corpus. To search for a specific word, such as "dear", you need to type:

```
DICKENS> "dear";
```
or
```
DICKENS> [word = "dear"];
```

As previously, press `space` for next page, `q` for coming back to command prompt.
Note that in case we prefer ignoring whether the letters are in lowercase or uppercase (i.e. in order to make our search case insensitive), the CQP query has as follows:

```
DICKENS> "dear" %c;
```

Constructing more complex queries.
The use of **regular expressions** makes the CQP query language very powerful. CQP makes use

of the following format for regualr expressions:

`exp1 exp2`: first `exp1` then `exp2` in sequence
`exp*`: zero or more occurrences of `exp`
`exp?`: zero or one occurrences of `exp`
`exp+`: one or more occurrences of `exp`
`exp1|exp2`: either `exp1` or `exp2`

For example, the following query retrieves the following variations of the word “regular”: regular, regulars, regularly, regularity.

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

The **dot** (‘`.`’) can be used as a wild card in CQP. As such, it will match *any* character. For instance, the following query would find words such as “than”, “then” and “thin”:

```
DICKENS> "th.n";
```

while the following query would find words starting with `un`, followed by any number of characters, and ending with `ly`, such as “unusually” and “unsuccessfully”:

```
DICKENS> "un.*ly";
```

The use of **logic** can also make the CQP query language very powerful. The boolean operators `&` (and) and `|` (or) can be used to combine two tests, while `!` (not) can be used to set inequality. For instance, the following query would find all occurrences of words matching “un.*ly” but different from “unlikely”:

```
DICKENS> [(word = "un.*ly") & (word != "unlikely")];
```

Similarly, the following query would find all occurrences of either the word “man” or the word “woman”:

```
DICKENS> [(word = "man")|(word = "woman")];
```

Question 1 - Concordance of a given bigram. Retrieve all occurrences of the bigram “great deal” in the corpus. Is this query affected by making it case insensitive?

Question 2 - Concordance of variations of a given word. Retrieve all variations of the word “kiss” in the corpus. Is this query affected by making it case insensitive?

2 Annotations

2.1 Displaying linguistic annotations

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.

2.2 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. Take note of the annotations used for the pos tags and try to identify the various different annotations that are used for verbs, adjectives and nouns. 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` In order to find out about the actual set of tags used to annotate the corpus, type `show cd` in `cqp`.

Question 3. Can you spot a pattern common to the pos annotations used for all the different types of verbs? Use a regular expression to formulate a query that retrieves all ocurrences of verbs in the corpus.

Question 4. Retrieve all tokens of any type except `verb`.

Question 5. Retrieve all occurrences of the word "lock" in which it is used as a `noun`.

Question 6. The `lemma` annotation in `cqp` contains the "headword" of a word, that is the principal dictionary word associated with the word. Use this information to formulate an alternative query for all variations of the word "kiss" in the corpus. Does this return the same results as your answer to Question 2?

3 Frequencies of a given word

Frequency information obtained from corpora can be useful for answering scientific or engineering questions. In order to find the total number of tokens in the corpus (i.e. obtain the token count) you need to type:

```
q1 = [];
size q1;
```

In order to obtain the absolute frequency of a specific word (for example the word "dear") you need to type:

```
q2 = [word = "dear"];
size q2;
```

Similarly, in order to obtain the absolute frequency of all bigrams consisting of the word "dear" followed by a noun you need to type:

```
q3 = [word = "dear"] [pos = "N.*"];
size q3;
```

Notice, however, that in order to obtain the absolute frequency of a bigram grouped by some criterion (for example grouped by the noun that follows the word "dear"), you need to type:

```
q3 = [word = "dear"] [pos = "N.*"];
count q2 by word;
```
or
```
q3 = [word = "dear"] [pos = "N.*"];
group q2 matchend word by match word;
```

A tricky point: Now let's suppose we want to find the most common word in the corpus. A natural query would be:

```
q4 = [word];
count q4 by word;
```

This works, but also puzzlingly includes punctuation marks and white space amongst words. Instead, the following works better:

```
q4 = [word = "[a-zA-Z].*"];
count q4;
```

Here "a-zA-Z" matches any letter, either lower or upper case.

Question 7 - Absolute Frequency. Find the absolute frequency of the word "girl" in the corpus.

Question 8 - Relative Frequency. Calculate the relative frequency of the word "girl" in the corpus.

Question 9. Find the most common bigram in the corpus.

4 Collocations

Question 10. Is the most common bigram a collocation?

Question 11. List the 10 most common adjective-noun pairs. Which of these would you classify as collocations?

Question 12. For a given adjective-noun pair AB (i.e. adjective A followed by noun B) we can create the *contingency table* of frequency observations. In fact, such contingency tables will be used later in the course. The table for AB looks as follows:

	A	$\neg A$
B	$F_{A\,B}$	$F_{\neg A\,B}$
$\neg B$	$F_{A\,\neg B}$	$F_{\neg A\,\neg B}$

where:
$F_{A\,B}$ is the absolute frequency of adjective A followed by noun B
$F_{\neg A\,B}$ is the absolute frequency of any adjective but A followed by noun B
$F_{A\,\neg B}$ is the absolute frequency of adjective A followed by any noun but B
$F_{\neg A\,\neg B}$ is the absolute frequency of any adjective but A followed by any noun but B

Construct the contingency table of frequency observations for the bigram "great deal".

Type `exit;` to quit the `cqp`-program.

5 Tutorial Discussion

- Discuss uses of corpora in Informatics.
- Discuss uses of corpora in Linguistics.
- Discuss how the above uses are related.