Informatics 1B: Data and Analysis

Lecture 8: Semi-structured Data: Querying Corpora

Frank Keller

School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

February 7, 2005

Reading: lecture notes; CQP manual.

Getting Information from Corpora

Last lecture:

- acquisition of corpus data (sampling and balancing);
- pre-processed (tokenization and sentence boundary detection);
- annotation (parts of speech, syntactic structure).

This lecture:

- how to do something useful with corpus data and its annotation;
- how to extract statistics that are useful for linguistic questions or NLP applications;
- how to use regular expressions for queries, obtain concordances, extract collocations from corpora.

Concordances

Concordance: all occurrence of a given word, displayed in context:

- generated by concordance programs based on a user keyword;
- keyword (search query) can specific annotation (POS, etc.);
- output displayed as *keyword in context:* matched keyword in the middle of the line, predefined context to left and right.

Example

Assume a linguist is interested in the usage of the word *remember*. This can be studies by compiling a concordance of this word.

Example: Concordance of *remember*

Keyword in context output for *remember* extracted from the Dickens corpus (corpus of novels; used in lab exercises):

```
 's cellar . Scrooge then <remembered> to have heard that ghost
, for your own sake , you <remember> what has passed between
e-quarters more , when he <remembered> , on a sudden , that the
corroborated everything , <remembered> everything , enjoyed eve
urned from them , that he <remembered> the Ghost , and became c
ht be pleasant to them to <remember> upon Christmas Day , who
its festivities ; and had <remembered> those he cared for at a
wn that they delighted to <remember> him . It was a great sur
ke ceased to vibrate , he <remembered> the prediction of old Ja
as present myself , and I <remember> to have felt quite uncom
```

Concordance Programs

Corpus Query Processor (CQP): program that generates concordances automatically. Will be used in this course (incl. lab and assignment).

CQP's query engine searches corpora based on user queries over words, parts of speech, or other markup.

Regular expressions make the CQP's query language powerful. Remember regular expressions from Inf 1A?

Regular Expressions

A regular expression consists of the following elements:

- *Symbol:* A symbol a in a regular expression R matches the string a.
- *Sequence:* A sequence R_1R_2 of two regular expressions matches the concatenation of the string matched by R_1 and the string matched by R_2.
- *Choice:* A choice $R_1|R_2$ of two regular expressions matches the string matched by R_1 or the string matched by R_2.
- *Repeat:* A regular expression R^* matches a sequence of 0 or more instances of R. The star in R^* is also referred to as the *Kleene star.*

Choice Operator

Example for CQP query:

(1) `[word="remember"];`

`word`: is a *positional attribute* (markup at a fixed position in the corpus (here: word form);

Value of the attribute is matched against the right hand side of the query (here: `remember`).

Right hand side of query can contain a regular expression, e.g., *choice operator* `|`:

(2) `[word="remember|remembers|remembered|remembering"];`

returns all forms of the word *remember* (see previous slide).

Kleene Star, Dot Operator

CQP regular expressions can also contain the *Kleene star:*

(3) `[word="blaa*"];`

matches `bla`, `blaa`, `blaaa`, etc. (`*` binds only the previous letter, not the whole expression).

CQP offers *additional regex operators* that make queries simpler. These include the *dot operator* matches any character:

(4) `[word="s.ng"];`

matches `sing`, `sang`, `sung`, but also `szng` and `s6ng`.

List and Repetition Operators

List operator `[...]` matches all characters in the list:

(5) `[word="s[iau]ng"];`

Abbreviations for subsets allowed, e.g., `[a-d]` or `[1-6]`.

Repetition operators: `*` zero or more repetitions, `+` matches one or more repetitions, and `?` matches zero or one instance.

Round brackets `(` and `)` *group characters* together; operator apply to the content of brackets:

(6) `[word="[Rr]emember(s|ed|ing)?"];`

matches the words `remember`, `remembers`, `remembered`, and `remembering`, and their capitalized forms.

Boolean Expressions, Sequences

Positional attribute `word` (word form) is available in every corpus.

Many corpora contain additional annotation, e.g., the *part of speech attribute* `pos`. Then we can say:

(7) `[pos="NN.*"];`

returns all nouns: `NN.*` matches NN for regular nouns, NNP and NNPS for singular and plural proper nouns, etc.

Regexes can be combined using *Boolean operators* `&` (and), `|` (or), and `!` (not):

(8) `[(word="like.*") & (pos=!"NN.*")];`

returns all words starting with `like` not tagged as noun.

Example: Concordance of *adjective* + *tea*

Queries can refer to *sequences of words:*

(9) `[pos="JJ.*"] [word="tea"];`

matches all instances of *tea* preceded by an adjective (`JJ`).

Example output:

```
now , notwithstanding the <hot tea> they had given me before
.' ' Shall I put a little <more tea> in the pot afore I go ,
o moisten a box-full with <cold tea> , stir it up on a piece
tween eating , drinking , <hot tea> , devilled grill , muffi
e , handed round a little <stronger tea> . The harp was there ; t
e so repentant over their <early tea> , at home , that by eigh
rs. Sparsit took a little <more tea> ; and , as she bent her
s illness ! Dry toast and <warm tea> offered him every night
of robing , after which , <strong tea> and brandy were administ
rsty . You may give him a <little tea> , ma'am , and some dry t
```

Collocations

Collocations: sequences of words that occur together.

Examples for collocations

- *run amok:* the verb *run* can occur on its own, but *amok* can't;
- *strong tea:* sounds much better than *powerful tea*, even though meaning is (roughly) the same;
- phrasal verbs: *make* can occur with *up*, *off*, *out*; but not *in*.

Task: automatically identify collocations in a large corpus. For example collocations with the word *tea* (see last slide):

- *strong tea* occurs in the corpus, while *powerful tea* doesn't;
- however, a lot of noise: *more tea* (not a collocation).

Collocations and CQP

Hypothesis: collocation occur more frequently than non-collocations.

Use CQP to compute *bigram frequencies* for all words that occur with *strong* and *powerful*. Define two named queries, Q1 and Q2:

```
(10)    Q1 = [word="strong"] [];
        Q2 = [word="powerful"] [];
```

Use the group command to obtain frequencies:

```
(11)    group Q1 matchend word by match word;
        group Q2 matchend word by match word;
```

groups together the values of word at the position matchend and sorts result by word at position match.

Example

strong	,	52	powerful	,	5
	and	31		effect	3
	enough	16		sight	3
	.	16		enough	3
	in	15		mind	3
	man	14		for	3
	emphasis	11		and	3
	desire	10		with	3
	upon	10		enchanter	2
	interest	8		displeasure	2
	a	8		motives	2
	as	8		impulse	2
	inclination	7		struggle	2
	tide	7		grasp	2
	beer	7		friends	2

Filtering Collocations

The bigram table shows:

- neither *strong tea* nor *powerful tea* frequent enough to make it into the top 15;
- but: potential collocations for *strong*: *strong desire*, *strong inclination*, and *strong beer*;
- potential collocations for *powerful*: *powerful effect*, *powerful motives*, and *powerful struggle*;
- *problem:* *strong and*, *strong enough*, , *powerful for*, bigrams with punctuation are also highly frequent.
- need to filter out "noise", i.e., distinguish collocations and non-collocations.

Statistical Tests

Problem: words like *for*, *and* are highly frequent on their own: they occur with *tea* *by chance.*

Solution: use statistical techniques to distinguish collocations from chance co-occurrences: *hypothesis testing:*

- *null hypothesis* H_0: there is no association between the two words are interested in;
- compute the probability p that we see the two words together given that H_0 is true;
- if p is sufficiently low, then reject H_0, and assume that the two words form a collocation.

We will look at one statistical test in detail: χ^2 test (next lecture).

Summary

- concordances show all occurrences of a word in a corpus, using keyword-in-context format;
- CQP automatically computes concordances using regular expression queries;
- CQP regexes can include: choice operator, repetition operators, list operator, grouping;
- queries can be combined using Boolean operators and sequencing;
- collocations are linguistically relevant word combinations;
- bigram frequency is not sufficient to identify collocations in a corpus; need statistical tests to filter out chance co-occurrences.