Unsupervised Language Learning

Computer Lab "Regular Expressions"

6-2-2014

1 Introduction

In the last decades, many cognitive and behavioral scientists, including linguists, have started creating large databases with data on human and animal behavior. In linguistics, such databases are called 'corpora', and they have become an important source of empirical evidence in many research projects on language use, acquisition and change. The availability of data from many different researchers in systems such as Childes MacWhinney (2000), has led to many new research possibilities, some of which you can exploit in your mini-project for this course.

In today's computer lab we will look at a powerful technique, called 'regular expressions', to search for specific patterns in corpora. Regular expressions can be used in many different programs, including the programming language Perl (where they originate), Excell (with the right package installed), the text editor Emacs and several specific programs such as RegExCoach for Windows. On linux computer there is the standardly installed program 'grep', which we will use today.

2 Getting started

Download the file 'corpora.zip' from `http://www.illc.uva.nl/laco/clas/ull14`

Move the file to your working directory and unzip it.

```
unzip corpora.zip
```

Now type:

```
grep -E 'ZZZ' ../corpora/welcome
```

If you see a couple of lines that all start with 'ZZZ' everything is working well. You have just run the grep program with a regular expression `ZZZ` on the file `welcome`. The programs returns by default only the lines of the file that contain a *match* with your regular expression. With `less welcome` you can see the full contents of that file (use `q` to return).

You can also let grep count the number of matches by adding `-c` to you command, or return the precise matches with `-o`, or return the lines that do not contain a match with `-v`. Try!

3 Regular expressions

3.1 Concatenation & alternation

With regular expressions, we can thus search for a specific word, but more interestingly, we can search for more abstract patterns. For instance, the file `ts-text-english.txt` contains the text of the novel Tom Sawyer; if we try to count the number of times the name 'Tom' occurs, we quickly discover the name is also written as 'TOM'. The regular expression `'TOM|Tom'` matches both alternatives. Alternatively, we can write that as `T(O|o)(M|m)`.

Question 1 *How often does the name appear in Tom Sawyer? How often in its Dutch translation? How often with an exclamation mark added?*

But there is a catch, because there might be occurrences of the sequence T-O-M that do not correspond to the name, but are part of larger words. For instance try:

```
grep -E 'TOM' ts-text-english.txt
```

We can make sure that our regular expression only matches complete words, by using the special character for 'beginning of word' `\<` and the special character for 'end of word' `\>`.

```
grep -E '\<T(O|o)(M|m)\>' ts-text-english.txt -c
```

Because we need this frequently, grep offers a shortcut with the option `-w`.

3.2 Sets & Repetitions

Dutch spelling involves many cases where two characters are used to represent one sound. For instance, 'oo', 'aa', 'ie' are very frequent. To get an idea which characters from the set 'aeuioj' can follow an 'i', we can use the regular expression notation `[aeuioj]` to indicate 'any member of that set'. Try the following command, which uses grep's option `-m` to restrict the number of lines from the input file it looks at:

```
grep -E 'i[aeuioj]' ts-text-english.txt -o -m 10
```

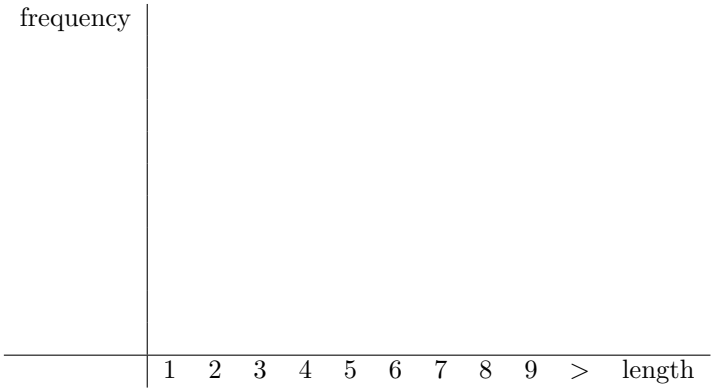
We can use this way of writing down sets in combination with special symbols for repetition to look for repetitions of characters from a specific set. For instance, `e[aeuio]+` matches every 'e' followed by any number of repetitions of another vowel; `e[aeuio]{2,}` matches every 'e' followed by *at least* two repetitions of another vowel. Look up in the grep documentation the meaning of remaining repetition operators `?`, `*`, `{n}`, and `{n,m}`.

Question 2 *Which triplets of vowel characters exist in Dutch? And in English?*

Question 3 *What is the longest consonant cluster in these languages?*

To specify sets, we can also make use of ranges: e.g., `[A-Z]` means 'all capitals' and `[0-9a-z]` means 'the set of all single digits or lower case characters'. Furthermore, some sets are frequently needed and therefore defined with a special name; e.g., `[[:punct:]]` matches all punctuation characters (see documentation). Finally, two important special symbols are the dot `.`, which represents the set of all possible characters, and `^`, which is used to indicate everything that is *not* in the set. That is, `(.+)` matches anything between parentheses (the brackets included), while `[^()]` matches any string that doesnot contain brackets.

Question 4 *People learning Dutch or – worse – German, often complain about the insanely long words. Using regular expressions and the two versions of Tom Sawyer, calculate the distributions of word lengths for Dutch and English? Do they differ substantially?*



3.3 Backreference

If we mark a specific part of a regular expression with brackets `()`, we can refer back to it with `\`n, where n is 1,2,3,... referring to the first, second, third, ... part between brackets. For instance, in `(.+) \<\1\>`, the first part matches any substring; the second part must match the *same* substring and be surrounded by word boundaries. Try:

```
grep -E '(.+) \<\1\>' ts-text-english.txt -m 20 -o
```

Question 5 *Which word repetitions do you find in Tom Sawyer? Are there words repeated thrice?*

4 Annotated Corpora

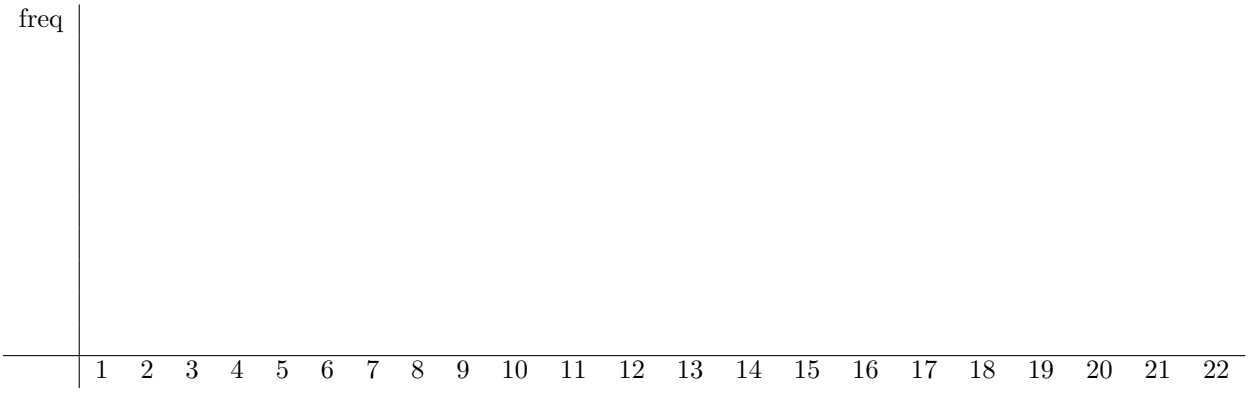
There are now also many large corpora available where the sentences are annotated with important syntactic information, for instance about the categories of words. An example is the file `BNCmini.txt`, which is extracted from the much larger *British National Corpus*. Have a look at this file using the command `less` (and quit it with `q`).

We can use corpora like this to start addressing some interesting linguistic questions. For instance, it has often been observed that there is a relation between frequency and (ir)regularity. Irregular verbs are almost always very frequent.

Question 6 *Check for 4 regular and 4 irregular verbs what their frequency is in the BNCmini corpus. Make sure you also count inflections of those verbs, and don't count homonyms that are not verbs (i.e. use the fact that syntactic categories are annotated in the BNC).*

	verb	frequency
irregular		
regular		

Question 7 *It has been suggested that the same holds for number words. E.g. 'thirteen' is still irregular, but twenty-one, twenty-two etc. is perfectly regular. Test this hypothesis by calculating the frequencies of use of (a representative sample of) the numbers 1-22, counting both numbers like 2 and numerals like 'two'.*



4.1 Treebanks

Some corpora are annotated with even more syntactic information, up to complete phrase-structure trees. Such corpora are called 'treebanks', and the most well known is the Penn 'Wall Street Journal' corpus Marcus *et al.* (1993). Have a look at this treebank with `less penn-wsj-line.txt`.

Treebanks allow us to get data on the frequency of specific syntactic phenomena, such as, e.g., the relative frequency of high PP attachment (as in "he ate the pizza with a fork") vs. low PP attachment (as in "he saw the woman with the handbag"). We can search for some examples of PP attachment with the following command (note the small `-e`, needed because we now want to match brackets):

```
grep -e '(VP (VB. [^()]\+) (NP.*(PP' penn-wsj-line.txt -m 4 -o
```

Question 8 *Can you get counts of the number of high and low PP attachments in this corpus?*

Question 9 *Regular expressions define* finite-state languages. *How is this fact related to your (in)ability to get precise counts of PP-attachments?*

References

MacWhinney, B. (2000). *The CHILDES project: Tools for analyzing talk. Third Edition.* Mahway, NJ: Lawrence Erlbaum Associates.

Marcus, M., Santorini, B. & Marcinkiewicz, M. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19**.