Tutorial 4: Unix Tools for Corpora[1]

Matthew W. Crocker

Computerlinguistik
Universität des Saarlandes
8 May 2001

Using and Combining Unix commands for Texts

`cat inputfile` write the contents of file to <standard output>
`less inputfile` display the contents of a file, one screen at a time
`less < inputfile` can also take its input from <standard input>

Pipes allow us to connect commands by using <standard output> from one command, as <standard input> to another.

E.g `cat inputfile | less`

Counting words with 'wc'

`wc inputfile` returns the number of line, words and characters in a file.

Using 'tr' to tokenise

The `tr` command allows us to translate characters from an input file to an output file. The general usage is as follows:

```
tr chars1 chars2   < inputfile   > outputfile
tr chars1 chars2   < inputfile   | less
```

parameters:
-c complement set
-s squish duplicates

Some examples:

`tr ´a-z´ ´A-Z´` converts all lower case characters to upper case
`tr ´.,:;?!´ ´.´` converts all punctuation to a period
`tr –c ´[0-9a-zA-Z]´ ´_´` converts all non-characters to _
`tr –s ´a-zA-Z´` squish all consecutive multiple characters

It is often helpful to tokenise a file into words, such that each word token appears on its own line in the output. We can use tr to do this by converting all non characters to <cr> and squishing consecutive <cr>s:

Simple tokeniser:

```
tr –sc ´0-9a-zA-Z´ ´\012´ < inputfile > outputfile
```

[1] Much of these notes is inspired by the unpublished manuscript "Unix for Poets", by Kenneth Ward Church, AT&T Bell Laboratories.

Sorting with 'sort'

The `sort` command allows us to sort lines of text. By default is sorts according to the first word on line, but can sort on subsequent words or columns.

Some examples:

`sort inputfile`	sorts line starting at the beginning of each line
`sort +2 inputfile`	sorts line starting at the third word/column of each line

parameters

-d	dictionary order
-f	fold case
-r	reverse order
-n	numeric

Once lines are sorted, the `uniq` command can be used to remove duplicate lines (that immediately follow each other).If used with the `–c` parameter, it will count the total number of occurrences of a line.

Finding tokens and types:

Using and combining the above tools we can do several useful things (in all cases the output goes to standard output; remember to add > outputfile if you want to send it to a file:

A tokeniser which puts each word on a separate line:

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile
```

We can now sort the list of words in the corpus:

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile | sort
```

If we want to investigate the word types and their frequencies, we can now do this easily by using uniq –c to count and remove multiple occurrences of each type:

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile | sort  | uniq –c
```

Finding N-Grams

Whether you just want to look at collocations, or build statistical NLP system, n-grams are often imports. Simply, n-grams are all the word sequences of length *n* that occur in a corpus. For example, bigrams are simply all pairs of adjacent words that appear in a corpus. One way to create a list of bigrams is to first create a tokenised list of words for the corpus, as shown above, and save it to a file:

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile   > out1
```

Then create an almost identical list of words, but offset by one word, so that the list of words begins with the second word of the corpus, not the first.

```
tail +2 out1 > out2
```

Finally, we can glue these two lists (i.e. columns) for words together to get a list of all the bigrams:

```
paste out1 out2
```

Filtering

It is always possible to filter the output of any command by using `grep` to find the lines of output you are interested in. Regular expressions are used to define the pattern of interest.

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile | sort  | uniq –c |
grep ´.*ly$´
```

See the attached page for examples of how to use `grep`, and basic regular expressions.

The string editor `sed` can be used to process a file in numerous ways (including doing complex pattern matching and string replacement). One simple task it can do is restrict the number of lines output:

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile | sort  | uniq –c |
sed 10q
```

The following with replace occurrences of ‘ly’ at the end of a string with ‘-ly’

```
tr –sc ´a-zA-Z´  ´\012´  < inputfile | sed ‘s/ly$/-ly/g/’
```

Finally

More details on these commands can be found in the standard Unix manual pages:

```
man <command_name>
```

Exercises

For each of the following, **give both the commands you used, as well as the actual answer**. Collect your answers in a **single** text file, and then e-mail them to: crocker@coli.uni-sb.de
You may work in pairs, submissions are due: **midnight, Monday 14 May**.

1. Show the frequency of each distinct vowel sequence in ‘example’ (e.g. ‘a’ or ‘ie’). Do not distinguish upper and lower case.
2. What is the total number of word types for both corpora, using the tokeniser above?
3. Change the simple tokeniser, so that it allows for numbers as well as words. Now what is the total number of types (i.e. distinct word/number forms) in both corpora.
4. Build on the tokeniser above, so that all uppercase characters are translated to lower case. For the german corpus, compare the 20 most frequent words with the 20 most frequent when case is left unchanged.
5. What are the 10 most frequent English words ending in ‘ing’?
6. The POS tagged Brown corpus is a bit of a mess. Tokenise in the usual way, leaving tags attached to their words, and use sort/uniq to determine the frequencies. Then use tr to put the tags for each token in a separate column, and use egrep to make sure you only process tagged words, and show the 10 most frequent words along with their tags.
7. Use ‘sed’ to do simple stemming (i.e. remove: ly, ed, ing) of the types in English corpus, then determine the new number of word ‘types’ found in the corpus, and compare this with your results from question 2.
8. List the bigrams from the English sample in descending order of frequency. What are the bigrams with a frequency between 90 and 100.
9. For the German corpus, list all the bigrams with a frequency of 4 or more, which contain the word ‘werden’.
10. **BONUS:** Find the POS bigrams in the Brown corpus, and list the 10 most frequent.