Unix Tools for Corpora

Thema: Einfache Unix-Befehle verwenden, um mit Textdaten zu arbeiten.

Voraussetzung: Shell-Grundlagen

Daten:

Link zu den Korpora mit folgendem Befehl erstellen:
`$ ln -s /home/MC/crocker/Mathe3/* .`
(Tilde am Anfang, Stern und Punkt am Ende nicht vergessen)

Befehle

- Sich eine Textdatei anschauen:
  `cat` („concatenate“: eine oder mehrere Dateien anzeigen)
  `more` (seitenweise anzeigen; mit Leertaste weiter; etwas primitiv)
  `less` (Wortspiel: besser als „`more`“; mit „q“ abbrechen)

- `head` und `tail`
  Optionen:
  `-nZAHL`: n Zeilen anzeigen (default: 10)
  `-n+ZAHL` (nur bei `tail`): Ab Zeile n Inhalt anzeigen

- Einlesen von Daten
  `$ BEFEHL DATEI` oder
  `$ BEFEHL < DATEI` (Syntax hängt vom Befehl ab)

- Ausgabe in einer Datei:
  `$ BEFEHL > DATEI`
  erstellt eine neue Datei bzw. ***überschreibt*** eine existierende Datei!
  `$ BEFEHL >> DATEI`
  fügt die Ausgabe des jetzigen Befehls am Ende der Datei an

- Ketten von Befehlen:
  `BEFEHL1 | BEFEHL2` (senkrechter Strich, „Pipe“ genannt)
  (die Ausgabe des 1. Befehls wird als Eingabe für den 2. Befehl weitergeleitet)

- Wörter, Zeilen und Bytes zählen:
  `wc DATEI` („wordcount“)

- „`tr`“ (translate bzw. truncate):
  `$ tr SET1 SET2 < DATEI`: Ersetzt die Buchstaben in SET1 durch die in SET2
  Anmerkung: Ein Wort durch ein anderes ersetzen geht mit `tr` nicht, da die Argumente keine Zeichenfolgen sind.

  Alle Kleinbuchstaben: `'a-z'`
  Alle Großbuchstaben: `'A-Z'`
  Alle Zahlen: `'0-9'`

  Nützliche `tr` Optionen:
  -c („complement set“): Alles, was ***nicht*** in SET1 ist durch SET2 ersetzen, z.B.
  `$ tr -c 'a-z-A-Z' _ < english`
  -s („squish“): Duplikate entfernen

  Ein einfacher „Tokenizer“ (ein Programm, das jedes Wort auf eine Zeile schreibt):

```
tr -sc 'a-zA-Z' '\n' < DATEI | less
```

- Sortieren: `sort DATEI`
  Nützliche Optionen (bitte **man sort** ansehen und selbst notieren):

- `uniq`: Entspricht „`sort –u`“, ist aber mächtiger weil es mehr Optionen hat.
  Welche Optionen können nützlich sein (**man uniq**)?

- Achtung: Um eine Zeichen***kette*** zu ersetzen braucht man aber `sed` oder `perl`:
  `sed "s/OLD/NEW/g"`
  `perl -pe "s/OLD/NEW/g"`

- Filtern
  `grep MUSTER`: Zeigt nur ausgewählte Zeilen an.
  Auswahl mit regulären Ausdrücken, s. „REGULAR EXPRESSIONS“ in „`man`“.
  Nützlich ist vor allem `grep –v`, um das Muster zu invertieren, um zu überprüfen dass das Muster genau das auswählt, was man will.

- `cut`
  Wie gibt man an, welche Spalte man trennen möchte?
  Wie trennt man Spalten, die nicht durch Tabulatoren getrennt sind?

- Rezept, um Bigramme zu finden: (Bigramme: Alle Wortpaare, die in einem Corpus vorkommen)

1. Datei tokenisieren und abspeichern.
2. Die Liste kopieren und das erste Wort löschen (s. `tail` auf der vorigen Seite)
3. Beide Listen nebeneinander ausgeben mit `paste`:
   `paste DATEI1 DATEI2`

- join: Zwei Tabellen nach einem gemeinsamen Element zusammenführen. Beide Dateien müssen nach der Spalte sortiert sein, nach der kombiniert werden muss. Wenn in einer Datei dasselbe Zeichen mehrmals vorkommt, wird das Kreuzprodukt ausgegeben.
  Bsp.:

Lex			Unigramme	
f	Tag	Wort	f	Tag
1	ADJA	weiten	233	ADJA

`join –j1 2 –j2 2`
2. Spalte aus der 1. Datei und 2. Spalte aus der 2. Datei

ADJA	1	weiten	233

DIE GENAUEN OPTIONEN SIND Z.T. BETRIEBSYSTEMABHÄNGIG!

Exercises

Answer the following questions using as few chains of Unix commands as possible. Editing the result by hand is not permitted.

- It is best to build each command line step-by-step, checking after each command that the output is what you expected (for example with `less` and `grep -v`).
- For each question, give **both the commands you used as well as the actual answer.** Collect your answers in a single text file and e-mail them to me.
- **You can work together in groups of up to 3 people.**

1. Show the frequency of each distinct vowel **sequence** in *example.txt* (e.g. *'a'* or *'ie'*; consider only *a,e,i,o,u* as vowels). Do not distinguish upper and lower case.
2. Use the naïve tokenizer (using tr, as described in instuctions, page 2) and combine it with further commands to find the total number of word types for the English and German corpora. Also allow for character sequences containing numbers.
3. Build on the tokenizer in the previous task 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. Present the output side-by-side.

Save the lower-case frequency lists as "german-word-freqs.txt" and "english-word-freqs.txt". You'll need them for a later tutorial.

4. What are the 10 most frequent English words **ending** in *-ing*? Look up how to specify the end of a word string in the man pages if you do not know.
5. Use `sed/perl` to do simple stemming (i.e. remove *-ly*, *-ed*, *-ing*) of the types in the English corpus, then determine the new number of word types found in the corpus, and compare this with your results in question 2.
   Hint: the "|" character means "or" within a regular expression.
6. List the bigrams from the German sample in descending order of frequency. What are the bigrams with a frequency between 60 and 70 (70 not included)?
7. The part-of-speech tagged Brown corpus is a bit of a mess. Tokenize in the usual way, leaving tags attached to their words, and use `sort` + `uniq` to determine the frequencies. Then use `grep` to make sure you only process tagged words, use `tr` to put the words and tags in a separate columns, and show the 10 most frequent words along with their tags.
8. Find the part-of-speech bigrams in the Brown corpus, and list the 10 most frequent.
9. List a) the tag frequencies b) the PoS-bigram frequencies and c) the frequencies of the word-tag pairs in *german_tagged.txt*. Omit the output in your answer to this question.

Save these files as *"german-tag-freqs.txt", "german-POS-bigram-freqs.txt" and "german-word-POS-freqs.txt"* (you'll need them for a later tutorial).