# Mérési segédlet

## Nyelvi modellezés a beszédfelismerésben című méréshez

Tarján Balázs
BME-TMIT
2016

A mérés során használt virtuális környezet alapértelmezésben nem tartalmaz kódolást segítő grafikus szövegszerkesztőt. Ha ilyet igényelsz például letöltheted ez:

sudo apt-get install gedit
passwd: meres

# 1. feladat

## Token számítás

- A mérés során a ***wc*** parancs sok esetben jön majd jól
- Leírásért lásd: ***man wc***
- Pl: ***cat szoveg.txt | wc –w***

## Type számítás

- Használd a „scripts/type.sh”-t
- Pl: ***cat szoveg.txt | type.sh***

## Szótárnövekedési görbe

- Használd a „scripts/print_N_percent.sh”-t, mely arra képes, hogy az 1. argumentumban megadott fájl sorainak kiírja a 2. argumentumban megadott százalékát
- Pl: ***print_N_percent.sh szoveg.txt 50*** (50 százalékát írja ki)
- **For** ciklus bash-ben:
    - for N in érték1 érték2 érték3 …
    - do
    - *műveletek…*
    - done

# 2-3. feladat

A mérés során az SRI (Stanford Research Institute) által kifejlesztett SRILM nevű programcsomagot fogjuk használni, melyet igen elterjedten használnak általános célú nyelvi modellek tanításához és teszteléséhez. Ez a programcsomag minden gépen telepítésre került.

## Nyelvi modellek tanítása

A nyelvi modellek tanításához tanítószövegre és a tanítást befolyásoló paraméterek megadására van szükség

*ngram-count –text „training.txt” –order „N” –lm „language_model.lm” –„smoothing”*

A nyelvi modellek tanításához az „ngram-count” parancsot használhatjuk.
***-text*** kapcsoló urát megadhatjuk a tanítószöveg nevét

***-lm*** kapcsoló után megadhatjuk a tanítandó nyelvi modell nevét
***-order*** kapcsoló után megadhatjuk a tanítandó modell fokszámát
***-smoothing*** itt különböző nyelvi modell simítási eljárásokat adhatunk meg

> **„”(semmi):** Good-Turing simítás
> ***-addsmooth k****:* Add-k simítás (k=1 Laplace simítás)
> ***-kndiscount -interpolate*****:** Kneser-Ney simítás interpolációval

Megjegyzés: kiindulási alapként használd a ***build_lm.sh*** scriptet!

# ARPA formátum

Ez a formátum, melybe alapértelmezésbe az SRILM toolkit menti a nyelvi modelleket:

```
\data\
ngram 1=n1
ngram 2=n2
...
ngram N=nN

\1-grams:
p       w               [bow]
...

\2-grams:
p       w1 w2           [bow]
...

\N-grams:
p       w1 ... wN
...

\end\
```

Értelemzése:

***data**:** a különböző n-gramok számát megadó header

***p:*** az adott ngram log10 valószínűsége. Például p = log10( P(*wN* | *w1* w2 … wN-1) ).

**[bow]**: back-off súly, ennek a tárgyalása túlmutat a mérés keretein

# Nyelvi modellek tesztelése

A nyelvi modellek teszteléshez a következő parancsot kell futtatni

*ngram -lm „language_model.lm” -order „N” -ppl „test.txt”*

Ahol *–lm* kapcsoló után adjuk meg a kiértékelendő nyelvi modellt, -order után a nyelvi modell fokszámot, -ppl után pedig a tesztszöveget.

Megjegyzés: használd a ***test_lm.sh*** scriptet!

# 4. feladat

## Részletes kiértékelés

*ngram -lm „language_model.lm” -order „N” -ppl „test.txt” **–debug 2***

## OOV modellezés

- OOV modellezéssel rendelkező nyelvi modell előállításához meg kell adni a szótárat is
- Pl: ***build_lm_unk.sh szöveg.txt 3 model.lm szótár.list***

- A szótár metszéséhez hívjad segítségül a ***type_freq.sh*** scriptet, mely a type-ok mellett azt is megadja, hogy hányszor fordultak elő a tanítószövegben
- Pl: ***cat szoveg.txt | type_freq.sh***

- Egy szöveges fájlból ***awk*** segítségével vághatsz ki oszlopokat:
- Pl: ***cat oszlopos.txt | awk '{print $3}'*** a 3. oszlopot írja csak ki

# 5. feladat

- A szöveg generáláshoz használd a ***gen_txt.sh***-t
- Pl: ***gen_txt.sh model.lm 3 1*** 1 db mondatot generál model.lm 3-gram modell alapján

- Mondat befejezéséhez használd a ***gen_prefix.sh***-t
- Pl: ***gen_prefix.sh model.lm 3 mondat.txt*** befejezi a mondat.txt-ben található mondatot model.lm 3-gram modell alapján

  Megjegyzés: Elég az elérhető legnagyobb fokszámú nyelvi modellt betölteni, a 2. paraméter segítségével kisebb fokszámú modellként is tud viselkedni!

# 6. feladat

## Zipf-görbe

- Használd a **type_freq.sh**-t!