UNIVERSITÉ CÔTE D'AZUR

I.A. et Langage :
Traitement automatique du langage naturel

Elena CABRIO

elena.cabrio@univ-cotedazur.fr

Serena VILLATA

villata@i3s.unice.fr

Shell Unix

Téléchargez le fichier pinocchio.zip et copiez-le sur votre bureau.

```
cat pinocchio.txt
```

```
tr "A-Z" "a-z" < pinocchio.txt
tr [A-Z] [a-z] < pinocchio.txt
```

```
tr -sc '[A-Za-zéèùàòì]' '[\012*]' < pinocchio.txt
```

Que se passe-t-il si nous utilisons cette commande?

Construire une liste de unigrammes

```
tr -sc '[A-Za-zéèùàòì]' '[\012*]' < pinocchio.txt | sort
```

Que se passe-t-il si nous utilisons cette commande?

Avec la commande

```
man sort
```

trouvez l'option à donner pour trier et obtenir la liste dans l'ordre inverse (pour sortir du manuel, appuyez sur la touche q)

Construire une liste de unigrammes

```
tr -sc '[A-Za-zéèùàòì]' '[\012*]' < pinocchio.txt |
sort | uniq
```

Que se passe-t-il si nous utilisons cette commande?

Avec la commande

`man uniq`

trouver l'option à donner à `uniq` pour obtenir une liste dans laquelle les fréquences sont également indiquées (pour quitter le manuel, appuyez sur la touche `q`)

Construire une liste de unigrammes

- Je souhaite obtenir une liste numérotée mais ordonnée en fonction de la fréquence des mots, par ordre décroissant (du plus fréquent au moins fréquent)

- À ce stade, je peux connaître la taille de mon vocabulaire à l'aide de la commande `wc -l`

- Comment savoir combien de tokens sont dans le document?

Construire une liste de unigrammes apres processus de racisination (stemming)

- Acceder a la page https://snowballstem.org/demo.html, et copiez-collez le texte de pinocchio.txt, pour obtenir sa version ``racinisée'' (si vous avez votre propre ordinateur, vous pouvez installer le stemmer et le lancer sur le document)
- Comparez-la avec le texte de départ

- Obtenez après la liste numérotée de unigrammes ordonnée en fonction de la fréquence des mots, par ordre décroissant (du plus fréquent au moins fréquent), et comparez-la avec celle obtenue sans racinisation. Que remarquez-vous?

Construire une liste de bigrammes

Bigrammes:

- N-grammes constitués des paires de mots consécutifs présents dans un corpus.
- Généralement associé à la fréquence dans le corpus (nombre d'occurrences)
- Utile dans l'étude des collocations et pour la création de modèles de la langue markovienne

Es.

```
8 jours difficiles
13 linguistique computationnelle
19 cours de
```

Construire une liste de bigrammes (shell)

Sauvegarder dans le fichier `pinocchio1.txt` la liste des token:

```
tr -sc '[A-Za-zéèùàòì]' '[\012*]' < pinocchio.txt > pinocchio1.txt
```

Apres, on fait une copie de la meme liste sauvegardee dans pinocchio1.txt en partant de la deuxieme ligne (on saute la premiere ligne):

```
tail -3215 pinocchio1.txt > pinocchio2.txt
```

Taille de votre fichier -1

Ensuite, nous combinons ligne par ligne les deux fichiers avec la commande `paste`

```
paste pinocchio1.txt pinocchio2.txt > bigrammes.txt
```

Construire une liste de bigrammes (shell)

Question:

Quels sont les bigrammes les plus frequents dans le texte?

Construire une liste de bigrammes (shell)

Nous essayons maintenant d'obtenir les cinq bigrammes les plus courantes en anglais pour voir le degré de correspondance entre les deux langues.

Au lieu de répéter la séquence de commandes, nous créons un `script shell`. De cette façon, il suffira d'appeler le programme depuis le terminal pour qu'il exécute une série de commandes en séquence.

Recherches textuelles avec *grep*

grep (general regular expression pattern matcher):

Commande pour rechercher des chaînes dans un ou plusieurs textes

```
grep ‘chaine’ inputFile
```

Es. Rechercher toutes les occurrences de 'bois' dans le texte pinocchio.txt

```
grep ‘bois' pinocchio.txt
```

Vous obtenez toutes les phrases contenant le motif recherché.

```
grep -3 ‘bois' pinocchio.txt
```

Vous imprimez également les 3 lignes avant et 3 après celles trouvées (important lorsque vous souhaitez effectuer une recherche sur le contexte des mots)

Recherches textuelles avec *grep*

- Imprimez maintenant toutes les phrases qui ne contiennent pas le motif recherché.

- Qu'avons-nous? Que signifie l'option -o?

```
grep –o 'boi.*' pinocchio.txt
```

Recherches textuelles avec *grep*

```
grep –o ‘plai.*' pinocchio.txt
```

Seules les chaînes (parties de phrases) correspondant exactement au motif recherché sont imprimées.
La recherche effectuée par des caractères génériques est appelée « **greedy** » (gourmande): les expressions régulières recherchent toujours **la chaîne la plus longue** possible parmi celles potentiellement identifiées dans la recherche.

Recherches textuelles avec *grep*

- Et si je voulais juste les mots qui contiennent "plai"?
- Effectuez une recherche de casse insensible, qui consiste à rechercher toutes les chaînes qui correspondent à l'expression régulière sans faire de distinction entre les majuscules et les minuscules
- Combien de token y a-t-il dans le texte pinocchio.txt commençant par une lettre majuscule?

Recherches textuelles avec e*grep*

- Combien de phrases dans pinocchio.txt contiennent deux occurrences de 'bois'?

- Que se passe-t-il si j‘ajoute l'option -o?

- Pour info, Si j'ai un dossier contenant des documents texte et que je souhaite rechercher tous les fichiers du dossier avec egrep, utilisez simplement l'option -r (récursive)

Substitutions textuelles avec la commande *sed*

sed = *Stream Editor*. Il permet non seulement de rechercher mais également de modifier les chaînes d'un fichier. Si *grep* correspond a “cherche”, sed correspond a “cerche&remplace”.

```
sed 's/is/was/’ < pinocchio_en.txt
sed 's/is/was/g’ < pinocchio_en.txt
```

Quelle est la différence entre les deux commandes?

Substitutions textuelles avec la commande *sed*

```
sed 's/Geppetto/father/g' pinocchio_en.txt
```

De même que grep, sed requiert également l'option g (global) pour traiter l'ensemble du fichier, même en cas d'occurrences multiples.

Le caractère spécial **&**:

Il est utilisé avec sed pour remplacer la chaîne trouvée avec elle-même.

Ex.

```
sed 's/very/& &/' pinocchio_en.txt
```

Si nous voulons voir les modifications apportées:

```
sed 's/very/& &/' pinocchio_en.txt | grep "very"
```

Substitutions textuelles avec la commande *sed*

sed vous permet de faire des substitutions en utilisant des expressions régulières

```
sed 's/[a-z]/bla/g' pinocchio_en.txt
```

Quelle est la différence avec la commande suivante?

```
sed 's/[a-z]*/bla/g' pinocchio_en.txt
```

Substitutions textuelles avec la commande *sed*

Il est également possible de ne remplacer qu'une partie de la chaîne trouvée. Dans ce cas, vous allez utiliser **\1**

```
sed 's/\([A-Za-z]\).*/\1/' pinocchio_en.txt
```

Je cherche la première occurrence de chaînes constituées de caractères et je ne garde que la première lettre.

Notez que la partie de chaîne référencée par \ 1 doit être placée entre parenthèses précédée de barres obliques inverses.

Substitutions textuelles avec la commande *sed*

Un dernier exercice pour la manipulation de texte

Dans le fichier pinocchio_en.txt, je souhaite remplacer toutes les occurrences de « wooden Marionnette » par la chaîne « wooden Pinocchio »

Chatbot (Regex)

- Amusez-vous dans l’implementation d'un chatbot simple basée sur des expressions régulières, sans utiliser des algorithmes d'apprentissage automatique ou d'apprentissage profond. Donc, évidemment, notre chatbot sera OK, mais pas très intelligent…
- Réfléchissez aux phrases ou questions possibles qu'une personne pourrait poser à votre chatbot. Généralisez-les en catégories distinctes et définissez des règles pour le pattern matching

Pour approfondir le sujet

K. Church: “Unix for Poets”

Un tutoriel qui explique pas à pas comment utiliser les commandes du shell Linux pour manipuler des documents texte (y compris les commandes non traitées dans ces leçons et les scripts de shell pour calculer pointwise mutual information).