Compter des mots et un peu plus ...

(voir ce sujet sans les cadres: [ici])

Dans ce TP, vous allez manipuler des textes à l'aide de commandes shell élémentaires. L'idée d'utiliser les commandes shell pour analyser des données textuelles est souvent boudée au profit de languages de programmation copmme Perl, Python, Javascript, C++ ou encore Java. Pourtant, les commandes shell permettent très souvent de réaliser très rapidement des outils (imparfaits) mais suffissants pour une première anaylse.

Vous avez une série de questions qui s'accomplissent à l'aide d'une dizaine de commandes unix qu'il est bon de connaître dans les grandes lignes. La section `liens` pointent vers de la documentation de ces commandes. Ne vous inquiêtez pas si vous ne parvenez pas à réaliser toutes les questions ! Si en revanche vous y parvenez, alors bravo!!! et inscrivez vous au DIRO et venez travailler avec moi !

Ressource

J'ai téléchargé de la Biliothèque Universelle (ABU) quelques textes sur lesquels vous pouvez travailler. Ces textes sont encodés en iso-latin1 et ressemblent tous à celui-ci: zola1.txt. Le texte commence par la license ABU, suivi d'une entête, puis du texte de l'oeuvre (ici *L'argent* d'Émile Zola). Le début du texte est marqué par une ligne qui commence par:

```
------------------------ DEBUT DU FICHIER
```

Vous pouvez télécharger cette collection de textes abu.tar.gz (puis `tar -zxvf abu.tar.gz`) ou travailler seulement avec zola1.txt que vous pouvez consulter dans un terminal directement en tapant:

```
cat /home/www-perso/usagers/felipe/HTML/CampInfo-2014/ressources/zola1.txt
```

Je vous recommande de le copier dans votre répertoire de travail:

```
cd
mkdir taln
cp /home/www-perso/usagers/felipe/HTML/CampInfo-2014/ressources/zola1.txt taln/
cd taln
cat zola1.txt
```

La suite du sujet fait l'hypothèse que les textes traités se trouvent dans votre repertoire de travail.

Cahier des charges

1) découper un texte en mots

Il s'agit d'une tache qui en pratique peut s'avérer très difficile (qu'est-ce qu'un mot?), mais en première approximation, utilisez la commande shell tr pour remplacer tous les caractères non alphabétiques par un retour chariot. Vous pouvez éliminer le texte de la licence et de l'entête à l'aide de la commande:

```
cat zola1.txt   | awk 'BEGIN {ok=0} \
                       /DEBUT DU FICHIER/ {ok = 1} \
                       /FIN DU FICHIER/ {ok=0} \
```

```
{if (ok>10) print $0; else if (ok) ok++}'
```

Si vous trouvez cela incompréhensible: pas d'inquiétude: vous êtes normal ! Cette commande permet d'écrire le texte qui se trouve entre les balises `DEBUT DU FICHIER` et `FIN DU FICHIER` présentes dans tous les textes distribués dans l'ABU, et élimine les 9 premières lignes qui contiennent le titre et l'auteur. La commande scorie permet de ne pas avoir à taper cela (vous pouvez l'utiliser directement depuis l'endroit où elle se trouve ou alors la copier dans votre répertoire de travail, comme suggéré ici):

```
cp  /home/www-perso/usagers/felipe/HTML/CampInfo-2014/ressources/scorie .
./scorie zola1.txt
```

Vous êtes libres de découper les mots comme vous le souhaitez, les mettre ou non en minuscule, etc. Votre commande doit produire un mot par ligne. Voici un exemple de sortie possible où les ponctuations ont été retirées, les mots ont été transformés en minuscule et tout caractère non alphabétique a servi de séparateur de mot:

```
onze
heures
venaient
de
sonner
à
la
bourse
lorsque
saccard
entra
chez
champeaux
dans
...
```

2) calculer une table de fréquence

Vous savez maintenant découper un texte en mots. Utilisez cela de façon à calculer la fréquence d'occurrence de chaque mot dans le texte. Voici par exemple les 10 mots les plus fréquents dans le texte `zola1.txt`, précédés de leur fréquence:

```
6972 de
4222 la
3201 il
2989 et
2930 le
2769 l
2729 à
2464 les
2446 d
2247 un
```

Indice: utiliser les commandes sort et uniq.

3) lister tous les mots d'un texte qui ...

1. ont exactement 4 caractères

   ```
   aile aîné airs amas anti âtre bade bals béat bébé béné béni bloc boit ...
   ```

2. commencent par le préfixe `p`

   ```
   % commencent par [abom]:
   abominablement abominations abomination abominables abominable
   ```

3. terminent par le suffixe `s`

   ```
   % terminent par [lissait]:
   désemplissait établissait faiblissait glissait pâlissait emplissait
   ```

4. contiennent au milieu (ni au début, ni à la fin) la chaîne `a`

   ```
   %contiennent au milieu [glou]:
   englouties engloutir engloutis engloutissait engloutissement
   ```

5. sont vus au moins `n` fois dans le texte

   ```
   % vus au moins 2000 fois:
   un d les à l le et il la de
   ```

6. exactement `n` fois dans le texte

   ```
   % vus exactement 108 fois:
   devait faisait monsieur paris
   ```

7. lus à l'envers sont encore des mots

   ```
   % mots d'au moins 4 lettres dont la lecture à l'envers est un mot:
   alla
   elle
   sexes
   écart / tracé
   ivres / servi
   port / trop
   tort | trot
   ```

Indice: utiliser la commande grep pour résoudre les 4 premières questions, et la commande awk pour les questions suivantes.

4) respirer

À ce moment, vous devez réaliser que sans programmer, vous êtes déjà capable de réaliser des choses non triviales. Pour celles et ceux qui savent programmer, réfléchissez au code que vous auriez écrit pour obtenir la réponse aux questions précédentes. Combien de lignes auriez vous produites pour cela?

5) calculer les bigrammes et leur fréquence

Écrire un programme capable de lister l'ensemble des **bigrammes** d'un texte et d'afficher leur fréquence d'occurrence

```
% 10 bigrammes les plus fréquents:
    842 de      la
    681 qu      il
    503 de      l
    477 d       un
    463 à       la
    410 d       une
    333 c       est
    298 qu      elle
    276 c       était
    274 à       l
```

Indice: utiliser la commande paste.

6) afficher les mots qui suivent un bigramme donné

Écrire un programme capable de lister l'ensemble des **trigrammes** d'un texte et d'afficher leur fréquence d'occurrence

```
% 10 trigrammes les plus fréquents:
     90 tout    de      suite
     84 il      n       y
     74 qu      il      avait
     70 de      l       universelle
     63 est     ce      que
     62 n       est     ce
     61 est     ce      pas
     61 il      y       avait
     57 ce      qu      il
     55 au      milieu  de
```

7) afficher les mots qui suivent un mot donné

Écrire une commande qui affiche les mots qui suivent un mot donné dans un texte. Ces mots doivent être triés en ordre décroissant de probabilité de suivre le mot spécifié (dans l'exemple, il est plus probable que `monde` suivre `le` dans le texte `zola1.txt`

```
% les 10 mots les plus probables pouvant suivre [le]:
monde temps plus petit cours jour premier marché voir cabinet
% les 10 mots les plus probables pouvant suivre [bel]:
homme air appétit emportement équilibre et ornement
```

Retrouvez dans le texte les passages qui contiennent un bigramme donné.

```
% contextes contenant [bel homme]:
s craintes de cataclysme. Quant à Salmon, un très bel homme luttant c
 éventail demeurait l'unique gloire, une barbe de bel homme qui faisa
```

8) afficher les mots qui suivent un bigramme donné

Écrire une commande qui affiche les mots qui suivent un bigramme donné dans un texte. Ces mots doivent être triés en ordre décroissant de probabilité.

```
% les mots pouvant suivre [la belle]:
chambre gaieté opération sécurité soeur unanimité
```

Retrouvez dans le texte les passages qui contiennent un trigramme donné.

```
% contextes contenant [comtesse de Beauvilliers]:
oline savait leur histoire. La comtesse de Beauvilliers avait beaucoup souffert de son
lle s'était rencontrée avec la comtesse de Beauvilliers ; mais celle-ci ne lui avait a
ue là par une apparition de la comtesse de Beauvilliers et de sa fille, dans le jardin
hambre entrait dire que Mme la comtesse de Beauvilliers demandait à être reçue. Saccar
Et, justement, la comtesse de Beauvilliers se trouvait là, avec sa fille
L'accueil que Saccard fit à la comtesse de Beauvilliers fut d'une brusquerie d'homme t
les, les désolés visages de la comtesse de Beauvilliers et de sa fille, qui le regarda
aient, les profils pâles de la comtesse de Beauvilliers et de sa fille Alice. Ces jour
L'avant-veille, le samedi, la comtesse de Beauvilliers s'était résignée à abandonner
y avait pas deux heures que la comtesse de Beauvilliers était installée, le samedi, lo
es yeux aveuglés de larmes, la comtesse de Beauvilliers ne songeait plus à cet homme m
```

9) afficher les bigrammes et trigrammes caractéristiques d'un texte

Ce qu'on appelle caractéristique est ici laissé à votre sagacité. On peut s'entendre sur le fait que le trigramme `il n y` n'est pas particulièrement spécifique à un texte donné.

Je vous propose la définition (discutable) suivante: un **n-gramme** caractéristique d'un texte est une séquence fréquente dans un texte dont les mots sont suffisamment longs (les mots outils sont souvent courts). Avec cette définition, voici une trace possible de votre commande:

```
% trigrammes caractéristiques de zola1.txt:
     70 de      l       universelle
     28 la      baronne sandorff
     24 rue     saint   lazare
     23 cent    mille   francs
     22 marquis de      bohain
% bigrammes caractéristique de zola2.txt:
    202 mme     caroline
    156 mille   francs
    153 elle    avait
    141 l       universelle
     91 toutes  les
% trigrammes caractéristiques de voltaire1.txt:
     12 candide et      martin
```

```
     11 monsieur         le      baron
     11 thunder ten      tronckh
     10 la       belle   cunégonde
      8 carnaval         à       venise
% bigrammes caractéristiques de voltaire1.txt:
     88 dit      candide
     48 la       vieille
     44 candide  et
     39 mlle     cunégonde
     32 dit      martin
```

10) récupérer une liste de mots outils du français

Les mots outils sont les mots d'une langue qui ne véhiculent habituellement pas de sens, mais servent un rôle grammatical, comme en français les articles, les conjonctions, etc. Il existe de nombreuses listes de mots outils (**stop-list** en anglais) disponibles sur le web. Récupérez-en une puis transformez la de manière à avoir un mot outil par ligne.

```
wget http://snowball.tartarus.org/algorithms/french/stop.txt
```

Combien de mots contient votre stop-liste ?

11) retirer les mots outils dans un texte

Mettre à profit votre stop-liste de manière à éliminer les mots outils dans un texte. Il sera plus simple de travailler sur une version **tokénisée** du texte (un mot par ligne).

Voici une sortie possible pour le texte `zola1.txt` et cette stop-liste. Il suffit d'enlever les mots `STOP` de la seconde colonne pour répondre à la question.

```
onze    onze
heures  heures
venaient        venaient
de      STOP
sonner  sonner
à       STOP
la      STOP
bourse  bourse
lorsque lorsque
saccard saccard
entra   entra
chez    chez
champeaux       champeaux
dans    STOP
la      STOP
salle   salle
blanc   blanc
et      STOP
```

12) retrouver les mots manquants

Certains mots d'un texte ont été effacés. Ce sont les mots marqués `GUESS`. Écrivez un programme qui essaye de restaurer ces mots. Ce programme prendra en entrée un texte tokénisé (un mot par ligne) avec certains mots marqués `GUESS` qu'il conviendra de remplacer.

1. Remplacez chaque mot `GUESS` par le mot le plus fréquent.
2. Remplacez chaque mot `GUESS` par le mot qui a le plus de chance de suivre le mot qui précède le mot `GUESS`
3. Suggérez d'autres approches

Afin d'évaluer la pertinence de votre programme, vous pouvez calculer le nombre de mots `GUESS` correctement retrouvés en vous comparant à une **référence** (ici le texte complet). Vous avez accès à une commande evaluation qui calcule le pourcentage de mots `GUESS` incorrectement traités. Voici un exemple d'utilisation où le premier fichier est le texte à traiter, le deuxième le texte traité (par l'approche 1 où le mot le plus fréquent est `le`) et le troisième le texte de référence (cliquez dessus en cas de doute):

```
evaluation zola1.guess zola1.cand  zola1.toks
good: 301 bad: 6990 err: 95.87
```

On observe que l'approche 1 corrige correctement moins de 5% des mots manquants, ce qui n'est pas très bon. Faites mieux !

Note: il s'agit d'une question difficile (à l'exception de l'approche 1.) pour laquelle une solution adéquate serait plus facile à écrire dans un language plus habituel (ex: python).