Loi de Zipf et loi de Heaps

Guillaume Wisniewski
guillaume.wisniewski@linguist.univ-paris-diderot.fr

septembre 2019

L'objectif de ce TP est de mettre en évidence deux lois fondamentales de la linguistique de corpus, la loi de Zipf et la loi de Heaps, à l'aide de commandes shell. Vous devrez m'envoyer par mail un compte-rendu de votre travail avant le **30 septembre à 8h**. Ce compte-rendu, au format pdf, comportera la réponse aux 16 questions de ce sujet.

1 Loi de Zipf

Cet exercice a pour objectif de mettre en évidence une loi empirique concernant la fréquence des mots dans un texte, la loi de Zipf. Nous utiliserons pour cela le corpus `WikiText-103` qui rassemble l'ensemble des articles de la Wikipedia anglaise des catégories « Good »[1] et « Features »[2] de l'encyclopédie et contient plus de 100 millions de mots. Ce corpus peut être téléchargé à l'url `https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-103-v1.zip`.[3] Les données ont été *tokenizées* de sorte qu'il y ait systématiquement une espace avant et après chaque mot (p. ex. des espaces ont été insérées avant les signes de ponctuations).

L'exercice se compose de deux parties. Dans une première partie, nous allons utiliser le shell pour compter le nombre d'occurrence des différents mots du corpus. Puis nous utiliserons un tableur pour représenter graphiquement le lien entre la fréquence d'un mot et son rang. Il peut être utile, avant de commencer, de se documenter sur les commandes shell suivantes : `sort`, `uniq`, `tr` et `unzip`.

Le corpus est suffisamment gros pour que l'on ne puisse pas travailler directement sur celui-ci :

— les commandes traitant la totalité du corpus sont relativement longues ;

— les sorties des commandes seront conséquentes (plusieurs millions de lignes) et il est difficile, voire impossible, de s'assurer que celles-ci fonctionnent correctement ;

1. `https://en.wikipedia.org/wiki/Wikipedia:Good_articles`
2. `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`
3. Il a été originalement collecté par Stephen Merity. La page « officielle » du corpus est `https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/`.

C'est pourquoi, il est fortement conseillé de travailler dans un premier temps sur un « petit » fichier (p. ex. les 10 premières lignes du corpus) et de n'appliquer les traitements à l'ensemble du corpus uniquement quand vous serez sûr que tout fonctionne correctement.

1.1 Compter les occurrences

① Créez un répertoire `tp_shell` (votre répertoire de travail) dans lequel vous stockerez tous les fichiers utilisés au cours de ce TP et exécuterez toutes les commandes demandées.

② Télécharger le corpus à partir de la ligne de commande et stockez-le dans votre répertoire de travail ?

③ Affichez le contenu du fichier de manière à ce qu'il y ait un mot par ligne (cela revient à remplacer les espaces par le caractère spécial « retour chariot » `"\n"`). Il est possible de supprimer les lignes vides en utilisant un pipe et la commande `grep -v -e "^$"`.

④ Affichez la liste des mots *uniques* du fichier triés par ordre alphabétique décroissant (les mots commençants par `"z"` avant les mots commençant par `"a"`)

⑤ Affichez pour chaque mot du vocabulaire, son nombre d'occurrences (il suffit d'ajouter une option à la commande précédente).

⑥ Triez les mots du vocabulaire par nombre d'occurrences *décroissant* et sauvez le résultat dans un fichier `word_frequency.csv`.

⑦ Affichez les 20 mots les plus fréquents du corpus.

Vous indiquerez, dans vos comptes-rendus, les lignes de commandes permettant de répondre à chaque question et la sortie de la dernière question. Vous expliquerez également ce que réalise chaque commande (en justifiant, par exemple, les options que vous utilisez).

1.2 Mise en œuvre de la loi de Zipf

L'objectif de cette partie est représenter graphiquement la relation entre le *rang* d'un mot et sa fréquence. Le rang d'un mot est défini comme l'ordre du mot dans la table des fréquence lorsque celle-ci est triée par ordre croissant : le mot le plus fréquent a le rang 1, le second mot le plus fréquent le rang 2 et ainsi de suite.

Il est possible de générer un fichier contenant le rang de chaque mot et la fréquence associée à l'aide de la commande :

```
head␣-1000␣word_counts.csv|awk␣-F"␣"␣'{print␣NR","$1}'␣>␣rank_versus_counts.csv
```

où `word_counts.csv` est le fichier généré lors de l'étape précédente. Comme pour la partie précédente, nous commençons par ne considérer qu'une partie du fichier afin de faciliter la mise au point de notre traitement.

Le fichier ainsi généré peut être importé dans n'importe quel tableur (par exemple Google Sheet [4]). Il est important, lors de l'importation, de bien préciser que :

4. Vous pouvez naturellement utiliser le tableur de votre choix.

— les différentes colonnes sont séparées par une virgule
— il faut *interpréter* les valeurs (pour que les nombres soient bien reconnus comme des nombres)

La figure 1 montre les différentes options utilisées lors de l'importation du fichier sur Google Sheet.

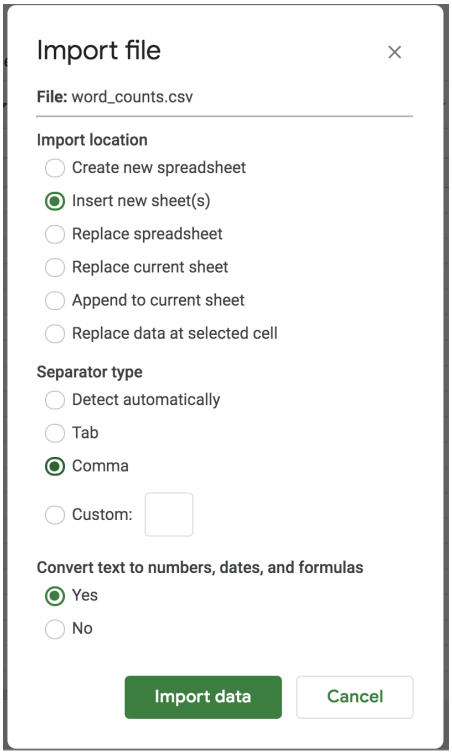

FIGURE 1 – Options choisies pour importer le fichier sur Google Sheet (accessible après avoir choisi la commande `File>Import`).

⑧ Représentez graphiquement la relation entre le rang et le nombre d'occurrences (en utilisant, par exemple, un graphe de type *scatter plot*).

⑨ Que se passe-t-il si on utilise des échelles logarithmiques sur les deux axes (pour Google Sheet, il suffit d'aller dans l'onglet `"Customize"` des propriétés du graphe).

⑩ Représentez sur le graphe la courbe d'interpolation correspondant aux données [5]. À quoi correspond une interpolation ?

⑪ Combien de fois apparait le mot le plus fréquent du corpus ? le 10^{e} mot le plus fréquent ? le 100^{e} mot le plus fréquent ? Comparez dans chaque cas la valeur observée et la valeur *estimée* par l'interpolation. Qu'en concluez vous ?

5. Pour Google Sheet, il suffit d'aller dans le menu `Customize` après avoir double-cliquer sur le graphe, puis d'activer l'option `Trendline` dans la partie `Series` en définissant une interpolation de type `Power Series` ; il est possible d'afficher le résultat de l'interpolation en définissant `Use Equation` comme étiquette du graphe

2 La loi de Heaps

Nous allons maintenant chercher à mettre en évidence une seconde loi empirique, la loi de Heaps, qui décrit la relation entre la longueur d'un document (le nombre de mots contenus dans celui-ci ou techniquement, le nombre de *tokens*) et le nombre de mots *distincts* utilisés (techniquement le nombre de *types*).

Nous utiliserons, dans cet exercice, le corpus contenant différentes œuvres écrites par Zola. Ce corpus est disponible à l'url `https://gw17.github.io/zola_books.tar.bz2`. Comme dans l'exercice précédent, tous les documents ont été tokenizés de sorte que chaque mot est suivi et précédé d'une espace.

⑫ Comptez le nombre de types (mots uniques) et le nombre de tokens du fichier `l_argent.txt` [6].

⑬ À l'aide du boucle `for` générez un fichier `n_tokens.csv` dans lequel chaque ligne sera composée d'une œuvre de Zola et du nombre de tokens dans celle-ci. Ces deux champs devront être séparés par une virgule et les lignes triées alphabétiquement.

⑭ Générez un fichier `n_types.csv` suivant la même structure que celui de la question précédente dans lequel vous stockerez le nombre de types de chaque œuvre.

⑮ À l'aide de la commande `paste` générez le fichier `type_token.csv` dont chaque ligne suivra le format :

```
nom_œuvre,n_types,nom_œuvre,n_tokens
```

Pourquoi est-il utile de répéter le nom de l'œuvre ?

⑯ À l'aide d'un tableur représentez le nombre de type en fonction du nombre de tokens. La loi de Heaps affirme qu'il existe une relation de type :

$$\text{n\_types} = K \times \text{n\_tokens}^{\beta}$$

Est-ce que cette relation est vérifiée pour le corpus considéré ?

6. Il s'agit de deux commandes distinctes