Traitement automatique des langues
TD2 (Analyse d'un corpus)

Youna Froger, Alice Gyde, Mathis Quais

22 octobre 2024

ThredUP

1 Réponses au questionnaire

1.1 What is the Europarl corpus, and why is it relevant ?

Le corpus en question est constitué d'un ensemble de documents provenant des transcriptions de débats au Parlement Européen, traduits dans plusieurs langues de l'Union européenne (Baisa et al., 2016). Ce corpus est particulièrement utile pour la recherche en traitement automatique des langues (TAL) en raison de la diversité de langues qu'il offre. L'intérêt de ce corpus réside également dans sa taille considérable, la cohérence de traduction entre les écrits et l'utilisation du langage soutenu. Il est souvent utilisé pour des applications de traduction automatique, comme Google Traduction.

1.2 What is a confusion matrix, and what is it used for ?

La matrice de confusion est un tableau utilisé en machine learning pour comparer les résultats obtenus par un modèle avec les valeurs réelles. Elle comporte quatre résultats possibles lors de la comparaison entre les prédictions et les valeurs réelles : vrai positif, vrai négatif, faux positif et faux négatif (Jedha, 2024). Dans le cas d'un système de classification multiclasse, la matrice peut comporter autant de classes que nécessaire.

1.3 What do Type I and II errors refer to ?

L'erreur de Type I correspond à un faux positif, c'est-à-dire lorsque la machine prédit qu'une personne est atteinte d'une pathologie alors qu'elle ne l'est pas. L'erreur de Type II, quant à elle, correspond à un faux négatif : dans ce cas, la personne est effectivement atteinte de la pathologie, mais la machine prédit qu'elle ne l'a pas. (Scribbr, 2024)

1.4 What do accuracy, precision, recall, and F1 stand for, and what do they measure ?

Accuracy (exactitude) :

$$\text{Exactitude} = \frac{VP + VN}{VP + VN + FP + FN}$$

Cela représente le pourcentage des bonnes prédictions (vrais positifs et vrais négatifs) sur le nombre total de prédictions.

Precision (précision) :

$$\text{Précision} = \frac{VP}{VP + FP}$$

Cela mesure le pourcentage de prédictions positives correctes sur l'ensemble des prédictions positives.

Recall (rappel) :

$$\text{Rappel} = \frac{VP}{VP + FN}$$

Cela représente le pourcentage des vrais positifs sur tous les éléments réellement positifs.

F1 Score :

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} = 2 \times \frac{VP}{(FP + VP) + (FN + VP)}$$

Le F1 Score est la moyenne harmonique entre la précision et le rappel. Ces formules permettent d'évaluer l'efficacité d'un programme. Le choix entre ces formules dépend des objectifs de l'étude et de la répartition des classes positives et négatives. Par exemple, pour un système de détection de spam, la précision sera plus pertinente si l'on souhaite éviter un trop grand nombre de faux positifs.

1.5 What does "Precision at k" (P@k) stand for, and what does it measure

La précision à k représente le pourcentage de résultats pertinents parmi les k premiers retours d'une recherche. Elle se calcule en divisant le nombre de résultats pertinents parmi les k premiers par k. C'est une mesure couramment utilisée pour évaluer les moteurs de recherche, par exemple.

2 Exercice du TP

Voici le lien GitHub : https://github.com/younafroger/TAL$_H$NM2/
Sur GoogleColab, les fichiers se suppriment automatiquement. Il faut les télécharger sur sa machine via le dépôt GitHub.

2.1 Fonction read corpus

Cette fonction prend un fichier texte en entrée, lit son contenu, convertit tout en minuscule, puis nettoie les caractères non-alphanumériques en les remplaçant par des espaces. Elle supprime également les espaces en trop. Cela évite que, lors de l'analyse des mots les plus utilisés, la "," arrive en première place. Sortie : tout le texte en minuscules sans ponctuation.

2.2 Fonction split corpus

Cette fonction divise le texte en une liste de mots (tokenisation) en utilisant le module nltk. Le paramètre lang (défini à "french" ici) permet de spécifier la langue pour que la division soit efficace. Sortie : une liste contenant chaque mot du texte original, exemple ['mot1', 'mot2', 'mot3', ...].

2.3 Fonction content to dict et content to list

La fonction content to dict crée un dictionnaire où les clés sont les mots du texte et les valeurs correspondent à leurs fréquences d'apparition. Le résultat est un dictionnaire avec chaque mot et son nombre d'occurrences. Exemple 'mot1': 5, 'mot2': 3, ...
La fonction content to list prend le résultat de la fonction précédente et transforme le dictionnaire en une liste contenant les couples mots/occurrences, et les trie par ordre décroissant de fréquence de mots, afin de mettre en avant le mot le plus utilisé dans le texte.

2.4 Fonction compute zipf

Cette fonction ajoute à chaque entrée de la liste une estimation de la fréquence attendue si on suit la loi de Zipf vue pendant le premier TD. Cette loi indique que la fréquence des mots dans un texte décroît de manière exponentielle. La liste rendue par cette fonction contient une troisième information, qui est la fréquence attendue par rapport à cette loi.

2.5 Fonction print stats

Cette fonction affiche un tableau des n premiers mots avec leur fréquence réelle dans le texte et la fréquence obtenue dans la fonction précédente.

2.6 Fonction plot zipf

Cette fonction génère un graphique qui montre comment les mots du texte respectent (ou non) la loi de Zipf en prenant en compte les indicateurs calculés précédemment.

2.7 Remove xml tags

En plus de ces fonctions que nous avions réalisées durant le TP, nous en avons créé deux supplémentaires. Tout d'abord, une fonction remove xml tags qui retire les tags xml à l'aide d'une expression régulière. Localement, nous concaténons les différents textes du corpus en un seul fichier par langue, nommé par le code iso. Il est nécessaire de renommer ce fichier par le nom de la langue en anglais (fi.txt en finnish.txt par exemple).

2.8 Process corpus folder

Enfin, une fonction process corpus folder opère toutes les fonctions ci-dessus sur nos fichiers. Elle prend en argument le dossier dans lequel se trouve notre corpus. Cette fonction enregistre à chaque fois l'image générée pour chaque langue, puis génère un graphique avec toutes les courbes. Par ailleurs, nous avons dû limiter la taille des fichiers sur lesquels ces opérations s'effectuaient du fait de leur taille (des centaines de mots chacun).

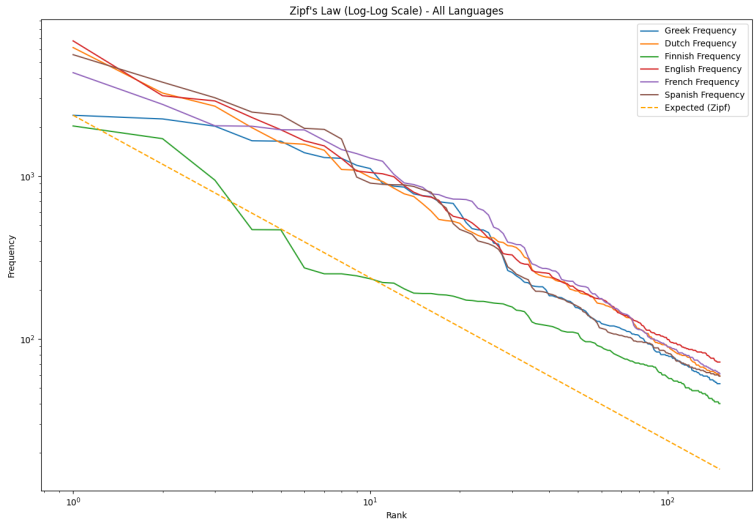

Figure 1: Graphique généré comparant le comportement des langues par rapport à la Loi de Zipf (log=True)

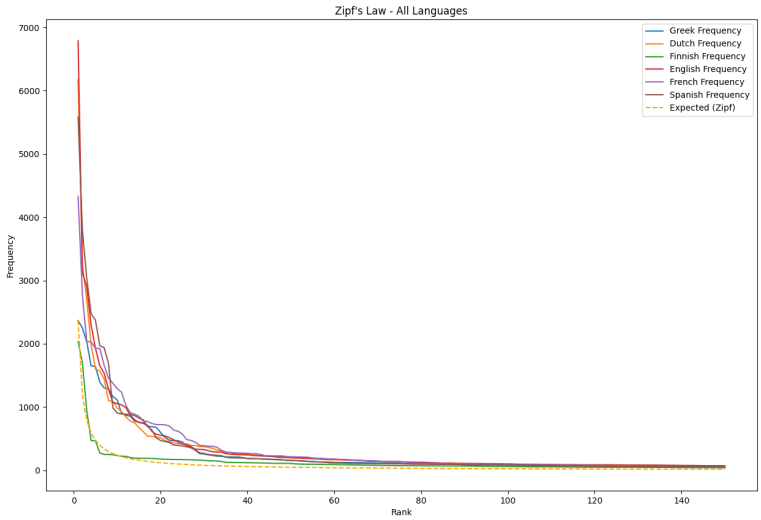

Figure 2: Graphique généré comparant le comportement des langues par rapport à la Loi de Zipf (log=False)

En sortie de ces fonctions, nous avons deux graphiques (Figure 1 et 2). La différence entre les deux graphiques réside dans le fait que le premier graphique a l'option log=True, ce qui applique une échelle logarithmique aux axes, permettant de mieux visualiser la relation entre le rang et la fréquence des mots sur une large plage de valeurs.

Chaque courbe représente la fréquence des mots pour une langue différente, et la courbe en pointillés représente la distribution attendue selon la loi de Zipf (la fréquence du mot le plus fréquent est à peu près inversement proportionnelle à son rang dans la liste des mots les plus utilisés). Toutes les courbes suivent globalement la loi de Zipf, avec une pente négative qui montre que les mots les plus fréquents apparaissent beaucoup plus souvent que les autres. Cela confirme aussi les ressemblances que nous avions constaté lors de notre précédent travail sur les textes du Projet Gutenberg.

La langue finnoise (courbe verte) semble diverger un peu plus par rapport à la loi attendue, avec des variations visibles dans les fréquences. Cette différence ne se voit pas autant dans le second graphique. Cela montre que la distribution des mots dans la langue finnoise est légèrement différente de celle des autres langues. Cela est certainement dû au fait que le finnois est une langue agglutinante, qui utilise beaucoup de suffixes pour créer des mots. Cela crée une grande diversité

dans les mots observés, et des mots globalement plus longs que dans les autres langues. De fait, pour le finnois, nous pouvons une nouvelle fois constater la même chose que durant le précédent TP.

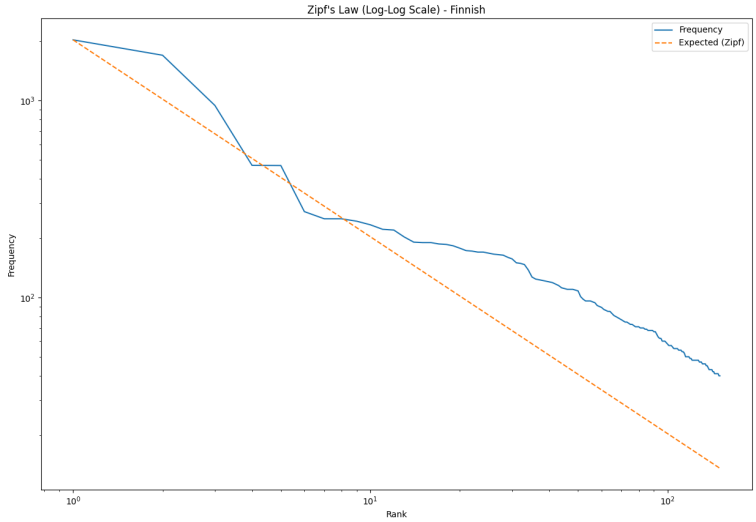

Figure 3: Graphique généré comparant le comportement du finnois par rapport à la Loi de Zipf (log=True)

3 Conclusion

En utilisant les facultés développées lors du premier TP, nous avons approfondi notre analyse des corpus de textes, au regard de la loi de Zipf. Le traitement des fichiers a été fait avec la librairie nltk, qui propose des fonctions utiles au traitement de corpus de texte. Nous avons surtout rencontré des difficultés dans le chargement des corpus EuroParl car ceux-ci pèsent assez lourd, et nos machines ne sont pas forcément assez puissantes pour les traiter efficacement. Cet exercice nous a bien montré les intérêts et les problématiques qui pouvaient émerger de travailler sur des corpus d'une taille conséquente.

References

Baisa, V., Michelfeit, J., Medveď, M., & Jakubíček, M. (2016). Europarl spoken parallel corpus [Accessed October 23, 2024]. https://www.sketchengine.eu/europarl-spoken-parallel-corpus/

Jedha. (2024). Matrice de confusion: Principe et explications [Accessed October 23, 2024]. https://www.jedha.co/formation-ia/matrice-confusion

Learning, I. M. (2024). Recall, precision, f1-score explained [Accessed October 23, 2024]. https://inside-machinelearning.com/recall-precision-f1-score/

Scribbr. (2024). Type i and type ii errors: Definition, examples, & comparison [Accessed October 23, 2024]. https://www.scribbr.com/statistics/type-i-and-type-ii-errors/

Wikipedia contributors. (2024). Europarl corpus [Accessed October 23, 2024]. https://en.wikipedia.org/wiki/Europarl_Corpus