Identification d'articles parallèles dans Wikipedia

Brice THOMAS
Université de Nantes - U.F.R. Sciences et Techniques
λ@etu.univ-nantes.fr $|\lambda \in \{brice.thomas\}$

1 Introduction

On va chercher à faire correspondre automatiquement des articles Wikipédia d'une langue source vers ces mêmes articles mais dans une langue cible. En gros dans notre cas, pour un article en français, on doit être capable de trouver ce même article rédigé en anglais. Et cela en se basant que sur le contenu de l'article.

Pour effectuer cette identification, nous nous basons sur l'article **A Fast Method for Parallel Document Identification** par *Jessica Enright* and *Grzegorz Kondrak*.

2 La méthode

La méthode est très simple et se résume en une phrase. On considère qu'un texte est parallèle à un autre s'ils partagent le plus d'**hapax** communs. Ainsi, pour chaque document on récupère uniquement les hapax. Réduisant considérablement les données qu'on manipule. Et là, on est content.

3 Implémentation

Bien qu'extrêmement simple, la difficulté de l'implémentation, pour notre cas, est que l'on va travailler avec beaucoup de fichiers (115k en français et 315k en anglais). Et même si on ne joue qu'avec les hapax, on en retrouve facilement plus d'une centaine par fichier. Cela peut donc prendre beaucoup de places en mémoire et surtout beaucoup de temps si on est pas malin.

La première étape consiste à extraire les hapax pour chaque fichiers de chaque langue. Rien de bien compliquer, surtout en *Scala* (voir HapaxExtractor.scala, méthode *apply*). A noter qu'on fait attention à ne pas avoir d'hapax plus petit que 4 caractères et moins grand que 20 caractères (pour éviter de prendre les liens par exemple). Une fois qu'on a les hapax, on les sauvegarde sur le système de fichier pour éviter de tout recalculer à chaque fois. Surtout que ça prendre du temps pour l'anglais (un peu plus de 10 minutes sur ma machine).

La deuxième étape va nous permettre de gagner **énormément** de temps dans la partie finale, à savoir l'identification. Vu qu'on est complètement dans de la RI, on ne peut se passer d'un bel index

inversé. En effet, on va construire un **index inversé des hapax anglais**. Ainsi, pour un hapax on saura directement dans quels articles anglais il se trouve, sans avoir à parcourir TOUS les articles anglais et perdre du temps bêtement. La construction d'un index inversé c'est simple, surtout en *Scala* (voir InvertedIndex.scala). Chez moi ça prend 15 minutes (en tirant bien sur les 8Go de RAM et en swappant).

Et enfin l'étape finale. Pour chaque article français, on parcourt tous ses hapax et pour chaque hapax on récupère d'un coup (merci index) tous les articles anglais qui l'ont aussi. Il n'y a alors plus qu'à trouver l'article le plus présent dans ceux récupérés. Et hop, c'est aligné (voir Main.scala, ligne 53-63). Cette dernière partie prend **2 heures** sur ma machine. C'est long, mais c'est bien mieux que plusieurs jours, surtout lorsqu'on oublie de sauvegarder les résultats (sans vouloir offenser un autre groupe #trollface).

4 Résultat

En utilisant *trec_eval* on obtient une *MAP* de **31,54%**. Ce qui était le résultat espéré avec cette méthode.