Brief Article

The Author

9 février 2017

1 Introduction

2 Architecture choisie

3 Alignement de mot

4 Interface

5 Classification morphologique

Après avoir effectué l'alignement des mots, nous avons voulu tenter de classifier les mots selon leur morphologie. Une première approche fut d'utiliser l'algorithme définit par John Goldsmith et al dans leur article de 2001 : http ://www.aclweb.org/anthology/J01-2001. Puis après des premiers résultats et un échange de mails avec Mr Goldsmith nous avons décidé de changer de méthode, en majeur partie à cause du temps qu'il nous restait pour trouver une solution.

5.1 Première méthode : Goldsmith

L'idée principale de cette méthode est de créer un ensemble de segmentation des mots en radicaux et en suffixes :

$$mot = \{(radical_i, suffixe_i)\} \tag{1}$$

Une fois cette segmentation effectué, on veut former ce qu'on appelle des signatures définit comme suit :

$$signature = \{radical_i, radical_j ...\} < - > \{suffixe_k, suffixe_m\} \tag{2}$$

tel que $\forall\ i,\ k\ radical_i + suffixe_k$ est un mot dans le texte. Chaque mot doit être encoder dans une seule signature.

Les premières segmentations nous permettent de créer un premier ensemble de signatures parmi lesquelles nous devons bien sûr en éliminer la plus part. Pour cela les auteurs de l'article utilisent la MDL ou Minimum Description Length définit par Rissanen. Ils définissent donc pour chacun des objets une longueur de compression, et vont chercher à minimiser la somme de toutes ces longueurs. Ne précisant pas comment ils effectue cette minimisation nous avons premièrement choisi de maximiser ce que les auteurs appellent l'information mutuelle pondéré d'un suffixe :

$$\frac{[suffixe]}{[kgram]} * log(\frac{[suffixe]}{\prod_{lettre \in suffixe}[letter]} \tag{3}$$

où $[suffixe]$ est le nombre de fois où le suffixe apparait dans le texte en tant que suffixe. En plus de cela nous avons cherché à maximiser pour chaque signature le produit suivant :

$$log(|radical|) * log(|suffixe|) \tag{4}$$

où $|suffixe|$ est le nombre de suffixe dans la signature. Ceci dans l'idée que nous voulions éviter au maximum les signatures ne contenant qu'un seul suffixe ou un seul radical puisque qu'elle n'apporte pas d'informations générales sur la langue du texte étudié.

Après un mois et demi sur cette méthode nos résultats n'était pas encourageant, nous avons décidé d'envoyer un mail à Mr Goldsmith pour plus de détails sur la méthode a employé. Il nous a grandement aidé, conforté dans nos résultats et nous a conseillé de changer d'estimateur et d'utiliser plutôt la MDL de Rissanen. Manquant de temps pour se plonger dans cette méthode et la mettre en place nous avons choisi de changer d'estimateurs et de méthode d'agrégation des signatures.

5.2 Seconde méthode : Patricia Trie et nouvelle heuristique

Ici nous avons commencé par encoder le texte dans un Patricia Trie.

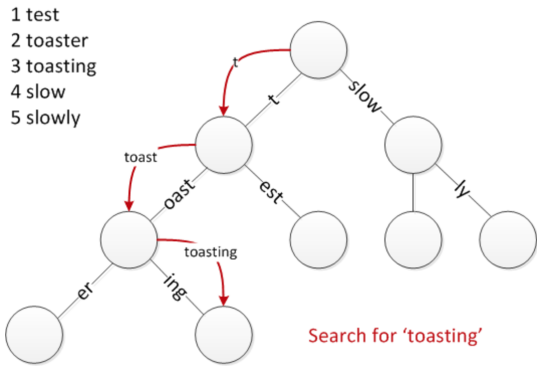

FIGURE 1 – Cette photo provient de la page wikipedia sur les patricia trie

Une fois cet arbre construit on peut avoir accès à ses feuilles dont on est sûr qu'elles constituent des suffixes. De plus en parcourant l'arbre et en s'arrêtant dès qu'on tombe sur un mot du texte on obtient une première décomposition du mot et de tous ceux qui suivent dans l'arbre. En bref, l'utilisation d'un patricia trie nous permet d'obtenir un premier sous ensembles de suffixes et une première segmentation pour chaque mot. Pour construire cet arbre nous avons bénéficié du code de Mr Clerc qui nous a permis de construire un premier trie (une lettre par noeud). Nous l'avons ensuite concaténé dès qu'il y avait deux lettres (noeuds) consécutifs n'ayant qu'un enfant.

Nous avons de plus garder l'heuristique développé dans l'article de Mr Goldsmith qui consiste à créer un ensemble de segmentations pour chaque mot. Puis pour caractériser la vraisemblance d'une segmentation $mot = \{(radical_i, suffixe_i)\}$ nous utilisons l'estimateur suivant :

$$(-1) * log(\frac{[radical_i + suffixe_k]}{[radical_i] * [suffixe_k]}) \tag{5}$$

Pour qualifier la vraisemblance d'un suffixe et d'un radical nous n'utilisons plus l'information mutuelle pondéré mais les estimateurs suivants :

$$suffixe : (-1) * \sum_{segmentation=(radical,suffixe)} log(\frac{[radical + suffixe]}{[radical] * [suffixe]}) \tag{6}$$

$$radical : (-1) * \sum_{segmentation=(radical,suffixe)} log(\frac{[radical + suffixe]}{[radical] * [suffixe]}) \tag{7}$$

Enfin on obtient donc pour chaque segmentation trois indices de sa potentielle pertinence : la vraisemblance de la segmentation, la vraisemblance du suffixe et celle du radical. Ces informations sont sauvegardées dans le fichier "$leftseg.txt$" pour le texte de gauche et "$rightseg.txt$" pour le texte de droite. Par exemple pour le premier Harry Potter en français on obtient pour le mot 'approchèrent' :

```
approchèrent : approchère | 11.78135971352466 :+: nt | 24449.13054101507 : = : 35.34407914057398
 approch | 93.97512396327853 :+: èrent | 4044.19535115923 : = : 28.84964653133697
 approchèren | 12.976743106734322 :+: t | 22156.839601702086 : = : 38.93022932020297
 approchèr | 11.113742166049189 :+: ent | 24343.19253010545 : = : 33.34122649814756
 approchè | 8.46352437327118 :+: rent | 6030.423909676321 : = : 25.390573119813542
```

FIGURE 2 – Exemple de triplet d'information pour plusieurs segmentation du mot 'approchèrent'

En regardant ses résultats on se rend compte qu'on ne peut pas choisir une segmentation en ne triant que sur un seul indice. Pour parvenir à une unique segmentation nous appliquons l'arbre de décision suivant :

Si pour les maximums pour chacune des informations nous donnent une unique segmentation on utilise celle-là.
Si on obtient deux maximum menant à une segmentation alors on garde celle-ci.
Si toutes les segmentations sont différentes alors on prends celle donnée par le maximum d'information provenant des radicaux.

Dans le cas de l'exemple ci-dessus on a :

— Le maximum au sens des segmentations est : 'approchèren' + 't'

— Le maximum au sens des suffixes est : 'approchèr' + 'ent'

— Le maximum au sens des radicaux est : 'approch' + 'èrent'

Nous avons trois segmentations, on garde celle au sens des radicaux donc : 'approch' + 'èrent'. Nous avons désormais une unique segmentation pour chaque mot, on créer les signatures et on les sauvegarde dans "$leftsig.txt$" et "$rightsig.txt$".

Par exemple on obtient :

```
('a', 'e', 'er') :
 confirma | confirm
 répliqua | répliqu
 accepta | accept
 accepter | accept
 réplique | répliqu
 confirmer | confirm
 accepte | accept
 remua | remu
 remue | remu
 remuer | remu
 confirme | confirm
 répliquer | répliqu


('a', 'ant', 'e', 'er') :
 plaquant | plaqu
 déchirant | déchir
 caresse | caress
 plaquer | plaqu
 tapoter | tapot
 caressant | caress
 tapota | tapot
 tapote | tapot
 déchire | déchir
 déchirer | déchir
 plaqua | plaqu
 plaque | plaqu
 caresser | caress
 déchira | déchir
 caressa | caress
 tapotant | tapot


('ient', 'cait') :
 menaient | mena
 menaçait | mena


('ait', 'u', 'urent') :
 parcoururent | parcour
 parcourait | parcour
 parcouru | parcour


('a', 'aient', 'ait', 'ant', 'e', 'er', 'ez', 'èrent', 'é') :
 tournez | tourn
 tournait | tourn
 tourné | tourn
 tournaient | tourn
 tourna | tourn
 tournant | tourn
 tourner | tourn
 tourne | tourn
 tournèrent | tourn
```

```
('i',) :
 rebondi | rebond
 malpoli | malpol
 ainsi | ains
 mistigri | mistigr
 lundi | lund
 pourquoi | pourquo
 ressenti | ressent
 assombri | assombr
 reviendrai | reviendra
 celuici | celuic
 chéri | chér
 accompli | accompl
 endormi | endorm
 celui | celu
 rendormi | rendorm
 aussi | auss
 viendrai | viendra
 mersssssi | mersssss
 ralenti | ralent
 étourdi | étourd
 enverrai | enverra
 fourni | fourn
 appellerai | appellera
 raccourci | raccourc
```