Théorie de l'Information
Analyse et transmission de texte aléatoire

Gregory ANNE
M1SIS

Table des matières

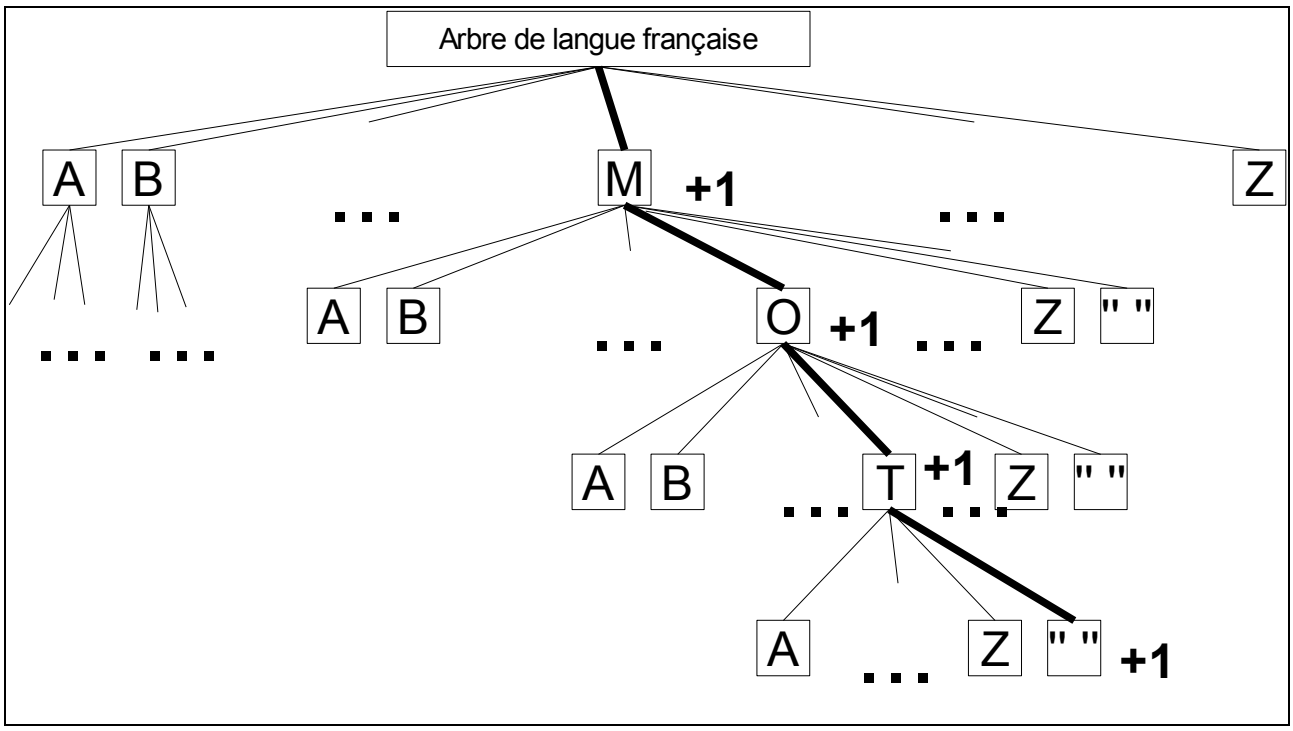
1 But de l'étude

Il s'agit d'élaborer un programme qui fonctionnera en 2 phases : la première est l'analyse d'une source de symboles, la seconde est la production de texte pseudo aléatoire au moyen des éléments statistiques accumulés lors de la première phase.

1.1 analyse

1.1.1 Mode général

L'analyse consiste à élaborer un arbre pondéré en lisant un texte ou un corpus textuel. Admettons qu'on considère que la source soit la langue française, ce corpus contiendrait alors des suites de mots les uns après les autres et ces mots sont eux mêmes des suites de lettres consécutives. A partir de ces données on peut retirer certaines informations, un classement d'occurences des lettres, mais encore mieux, on peut déterminer un classement d'occurences en fonction de la lettre précédente et c'est ainsi que l'on construit l'arbre pondéré en nombre d'occurences de la lettre avec au niveau supérieur sur l'arbre la lettre précédente dans le mot analysé. (voir sur figure 1 une illustration de l'analyse du mot « mot »)



<u>figure 1 :</u> arbre de la langue française et ajout à la pondération du mot « mot ».

Cet arbre est dessiné ici selon une représentation maximale : comme si à chaque niveau toute lettre suivante était possible, or ce n'est pas le cas. Prenons en exemple les mots commençant par la lettre Z, et bien dans le dictionnaire des noms communs, z peut être suivi de " ", a, e, i, l, o, u, y et rien d'autre. L'item Z en haut à droite sur la figure ne devrait donc posséder par exemple que 8 fils parmi les 27 symboles disponibles (on considère l'alphabet latin, sans distinction de minuscules ou majuscules)

1.1.2 Mode approximé

Ce qui a été proposé est d'approximer cette étude à un certain ordre et d'analyser les suites de lettres selon un certain ordre. Ceci limite la profondeur de l'arbre. On se sert ainsi d'une fenêtre pour visualiser ce n-uplet de lettres (voir figure2) :

- à l'ordre 1 on obtient la simple probabilité de la lettre dans la langue
- à l'ordre 2 on obtient probabilité d'apparence du couple dans la langue étudiée
- à l'ordre L, on obtient la probabilité du L-uplet de symboles.

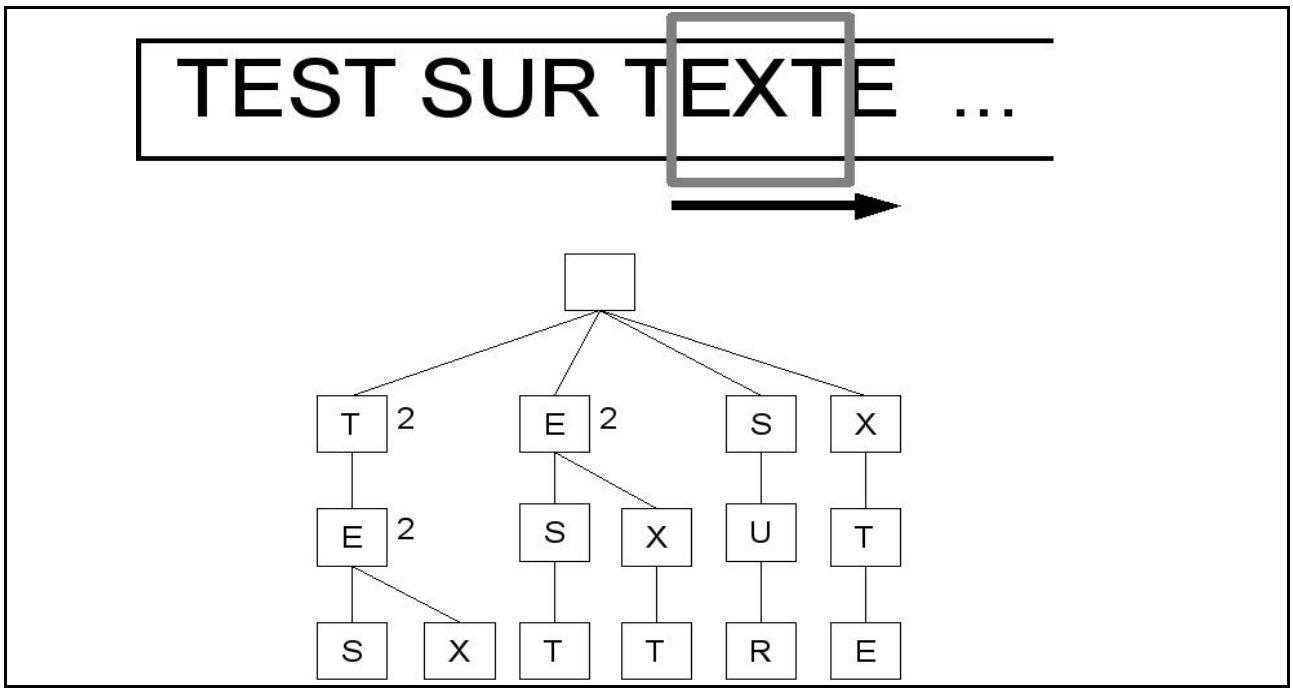


figure 2 : construction de l'arbre avec une fenêtre de largeur 3 se déplaçant de symbole en symbole

1.2 production

La production de texte est simple : il s'agit de parcourir l'arbre produit lors de l'analyse et de tirer au sort à chaque étape la lettre suivante.

Ce tirage au sort se produit selon les pondérations enregistrées sur l'arbre. Par exemple imaginons que la lettre A aie trois fils B(3), C(4), D(2), et E(1). On aura alors pour symbole suivant 3 chances 10 d'avoir un B, 4 chances sur 10 d'avoir un C, 2 chances 10 pour D, et 1 chance sur 10 pour E. L'algorithme de tirage au sort des lettres devra donc traduire sur cet exemple p(B)=3/10, p(C)=2/5, p(D)=1/5, p(E)=1/10.

Lorsque l'on arrive aux feuilles des arbres, on reprend au niveau 1 les noeuds contenant la même valeur pour poursuivre.

2 Principes mis en jeu.

2.1 Chaînes de Markov

Aussi bien lors de l'analyse que lors de la production, on considère chaque mot (version 1.1.1) ou le texte (version 1.1.2) comme une chaîne de Markov.

(La méthode adoptée est la version 1.1.2.)

En effet, chaque lettre utilisée dans le texte est une variable aléatoire Xk. Le texte s'écrit X1X2X3X4...Xn, n étant la taille du texte.

Il y a 26 symboles possibles : les 26 lettres de l'alphabet. Donc, pour tout $r \in L$ $(Xk = r)=1/26$ (L est l'alphabet utilisé).

Mais les pondérations de l'arbre indiquent cette fois des probabilités (approximatives puisqu'elles sont le résultat de l'analyse d'un échantillon de la langue française, cette analyse ne peut être exhaustive) selon lesquelles on obtient une lettre suivant la précédente : on a donc une valeur $P(Xn+1 = r|Xn)$. Ceci identifie la probabilité markovienne (probabilité de transition d'un pas).

On peut ainsi recalculer la probabilité de chaque symbole à une position k :

P(Xn+1 = r) = ∫ P(Xn+1 = r|Xn) P(Xn) dXn

2.2 Entropie de Shannon

L'article de Shannon « A mathematical theory of communication » introduit la notion d'entropie depuis une modélisation Markovienne de la langue anglaise ; il y démontre à quel point la langue anglaise est prédictable en utilisant une approximation d'ordre 1 seulement.

L'entropie est une fonction correspondant à la quantité d'information contenue dans une source. L'énoncé de Shannon est le suivant : plus la source est redondante, moins elle contient d'informations : l'entropie est maximale pour une source dont tous les symboles sont équiprobables. Plus l'entropie est faible plus il est décrire un comportement systématique de la source.

$$H(X) = -\mathbf{E}[\log_2 p(i)] = \sum_{i=1}^{n} p(i) \log_2 \left(\frac{1}{p(i)}\right) = -\sum_{i=1}^{n} p(i) \log_2 p(i).$$

et p(i) = ∫ P(i | i-1) P(i-1) d(i-1) : p(i) n'est que peu aléatoire sachant i-1 (exemple de Z qui ne peut être suivique de 8 symboles uniquement) et sachant que l'alphabet ne contient que peu de symboles.

L'entropie de la langue française devrait être suffisament faible pour que l'on puisse générer des mots en français correct de manière plutôt fréquente.

2.3 Turbocode

Le turbocode devenu un standard pour de nombreux systèmes comme le GSM-3G par exemple est né ainsi : il se base se le système d'entropie. Il adopte comme tout correcteur d'erreur la redondance dans la source, il établit des statistiques et corrige ainsi les transmissions érronées.

3 Algorithmique et codage

3.1 choix

Malgré plusieurs tests en C, Le java semblait être le plus apte à l'implémentation de ce problème. Il permet la lecture dans les fichiers, ainsi que l'écriture pour la seconde phase, et il permet aussi la génération d'arbres de façon simple. Tout l'intérêt du code étant dans la constation des résultats, le java était tout aussi fiable et efficace que le C.

Question algorithmique, il a été beaucoup plus judicieux de présenter tous les fils d'un noeud comme des freres, plutot que d'avoir un arbre n-aire dont l'optimisation du code aurait été peu avantageuse. Ainsi l'arbre possède une racine (variable theRoot) et cette racine n'a pas de frères, mais un fils et tous les frères de ce fils sont les premières lettres possibles du corps de texte entré en paramètre (la fenêtre). Chacun de ses frères ont eux mêmes un fils qui a lui aussi n frères, ces frères étant toutes les lettres possibles pouvant suivre la première lettre. et ainsi de suite.

3.2 code

en annexe.

4 Resultats

Voici le résultat d'une execution à l'ordre 4, avec une production de 500 mots lus ensuite sur l'arbre. Le texte fourni en source est le livre (intégrale du premier tome) de Jean-Jacques Rousseau « le contrat social » récupéré sur wikisource.org et passé à travers un script pour eliminer tout espace ponctuation accent ou majuscule. La limite du script étant qu'il n'a pas éliminé les cotes, il en reste comme on peut l'observer ici.

```
poti dmet huma jete rter unan etab harg quoi quoi
 ticu dues deme amod quoi font aien mbla guer publ
 ayem adic dmet denc jusq hacu quoi hacu jete dues
 tait hilo sses pist prou mett quoi scen etab etab
 mens ndam quic roup juri jete hine quoi prud rmin
 publ tyra ayem rise reur aibl rbit appe quoi  hui
 roup null doxe revi ppel quoi arda teur pron hete
(...)
vage qu  o imes adox jete etab duit vrai coup vain
 rect juri ppel ourr mell publ scen pire occu ourr
 jama rava vati quab poli phiq mett prou publ  hui
 qu  e roup mpir oute  hui nser denc sque croi huma
 scen sfor etab agem hant haqu emis croi assi appe
 nais jete nexp gali gato tude buti prud caus clop
 rume hacu publ tabl vote iqu  ssac publ d  hu prud
```

5 Conclusions

on remarque sur le resultat que l'on retrouve logiquement uniquement des ensembles de lettres que l'on peut rencontrer dans la langue française. Mais de plus, on remarque que l'on retrouve parmi les mots formés des mots eux mêmes de langue française et ce de façon très itérative. L'entropie de la langue française est donc faible, elle est grandement prévisible et descriptible par un ensemble de règles.