Consultation " floue " de grandes listes de formes lexicales simples et composées : un outil préparatoire pour l'analyse de grands corpus textuels.

Alain Lelu , Mohamed Hallab

Université Paris 8 / Département Hypermédias.
2 rue de la Liberté - 93200 Saint Denis

Abstract

The relevance of a textual data analysis depends heavily on the relevance of the lexical processing stage prior to this analysis. Full-text techniques, as well as linguistic or statistical compound terms detection, produce huge lists of terms, with a great deal of lexical variants or derivatives, or mispellings, needing semi-automatic revision. We present here a tool contributing to such a revision process, for the " fuzzy matching " of character strings, based upon N-gram inclusion and sequence indices. Unlike edit distance, computed using dynamic programming, our technique implements a real-time computation of the lexically closest terms to a given term, among lists of tens of thousand terms.

Résumé

La qualité d'une analyse de données textuelle repose principalement sur la qualité de la phase d'extraction de termes qui précède (indexation, au sens général). Les techniques de détection de termes composés produisent des listes immenses, dont l'accès est indispensable pour exercer un contrôle éditorial sur le vocabulaire d'indexation. Nous présentons ici un outil de détection d'"inclusion floue" d'une chaîne de caractères dans une chaîne plus grande, utile dans ce contexte, et basé sur un indice d'inclusion calculé à partir des fréquences de N-grammes, puis sur un indice de séquence. A la différence du calcul de la distance d'édition, qui utilise la programmation dynamique, notre technique réalise un calcul en temps réel des termes composés incluant de façon approchée la chaîne fournie en requête, parmi une liste de plusieurs dizaines de milliers de termes.

Mots-clés : Approximation lexicale, N-grammes, distance d' inclusion, distance d' édition, programmation dynamique, comparaison de chaînes.

1. Introduction

L'analyse des données textuelles repose sur un processus de réduction fondamental : représenter un texte par un ensemble, le plus souvent non ordonné, de mots, que ces mots soient l'ensemble des formes lexicales d'origine trouvées dans le texte (indexation en texte intégral), ou bien tout ou partie de ces formes ramenées à leurs lemmes, ou encore qu'ils soient constitués d'expressions composées – les véritables atomes sémantiques de base ; dans le cadre d'applications documentaires, ils peuvent aussi être attribués indépendemment du contenu formel du texte.

Quiconque veut aller au delà de l'accès ponctuel à une partie de texte que réalisent les traitements de texte classiques ne peut échapper à ce processus, traditionnellement dénommé indexation. Son "revers" est qu'il représente une véritable opération éditoriale, faite de choix implicites ou explicites en fonction des buts poursuivis. Par exemple, les mots à éliminer pour qui s'intéresse aux sujets traités, seront au contraire à conserver pour qui s'intéresse à la stylistique.

Tout comme les personnes "apolitiques" réalisent des choix politiques implicites, il n'existe pas d'indexation neutre, "objective". Les premières applications sur le World Wide Web de l'analyse des données textuelles (ex. : AltaVista / Refine), et les tentatives de commercialisation d'environnements d'analyse textuelle prétendus « tout automatiques » (U-Map, ThemeScape, Starlight, …) butent sur ce constat, qui semble bloquer jusqu'à leur essor commercial : sans contrôle éditorial de l'indexation, ces automates ne peuvent créér systématiquement le sens que l'on s'attend légitimement à reconnaître dans les résultats de leur travail.

Nous ne parlerons pas ici du nécessaire aller et retour entre résultats des analyses et paramétrage de l'automate, *leitmotiv* de l'analyse des données, ni de l'autre choix délicat, celui des unités sémantiques de discours au sein desquelles seront observées les associations entre mots, souvent traité dans les précédentes JADT. C'est au perfectionnement de l'environnement informatique de contrôle éditorial de l'indexation que nous nous attacherons dans le présent travail.

Il est à signaler que le monde de la recherche d'informations n'échappe pas non plus à cette problématique, car chaque utilisateur final a des attentes, porte un regard particulier, de façon la plus souvent inconsciente, sur les documents au sein desquels porte sa recherche : si cette remarque est particulièrement évidente pour la recherche iconographique (le regard d'un iconographe de presse « people » n'est pas le même que celui d'un illustrateur d'encyclopédie…) [Wertel-Fournier 99], elle vaut aussi pour la recherche textuelle, en particulier sur Internet.

2. Pour un environnement performant d'indexation contrôlée des données textuelles.

Le but est pour nous d'aider l'analyste en interaction avec le corpus 1) à faire les meilleurs choix, 2) à les faire rapidement, "au rythme de l'intellect". Non seulement doit-il pouvoir consulter vite et commodément de grandes listes de mots, mais encore doit-il pour chaque mot retenant son attention prendre des décisions "globales" (garder, éliminer, fusionner, corriger) ou "locales" (garder ou éliminer dans tel ou tel document.) *en connaissance de cause*, c'est à dire en faisant défiler les contextes individuels de ce mot sans opérations parasites risquant de lui faire perdre le fil de sa pensée.

Les listes de mots doivent pouvoir être ordonnées suivant tous les critères souhaitables : ordre alphabétique, nature grammaticale des mots, dates de modification, mais aussi selontous critères statistiques (fréquence, indicateur d'uniformité de répartition, …) permettant des opérations globales d'élimination ou fusion, avec possibilité à tout moment de revenir sur l'effet d'une décision [Rhissassi, Lelu 98].

Extraire des mots composés, par des méthodes statistiques ou linguistiques, s'avère indispensable pour les traitements ultérieurs, à même de mettre à jour les relations d'ensemble entre ces atomes de sens au sein de tout ou partie d'un corpus. Mais il faut savoir que plus la taille du corpus augmente, plus ces mots composés sont nombreux : ils n'obéissent pas à la loi de saturation des unitermes autour de la "limite du dictionnaire" (quelques dizaines de milliers), mais atteignent facilement quelques centaines de milliers, comme nous avons pu le constater avec un corpus d'un millier d'articles du Monde. Des méthodes beaucoup plus puissantes de consultation de listes, et pourquoi pas, de structuration du vocabulaire sur une base lexicale deviennent donc nécessaires. C'est au premier objectif évoqué que s'attaque le présent travail.

A partir du moment où des expressions composées sont en jeu, il faut prévoir un accès par "composant", quelle que soit soit leur position dans l'expression, dans les deux directions : à partir du mot *développement*, l'utilisateur doit accéder à *pays en voie de développement*, aussi bien qu'à *développement durable*. Mais aussi *moyens de développement* doit mener *à moyens humains*, aussi bien qu'à *société de développement*. Les variantes lexicales, les fautes de frappe ou d'orthographe, nombreuses dans les textes réels, … et le bruit inhérent à la phase de lemmatisation éventuelle devront être tolérés. En effet l' état de l' art actuel nous montre que toute procédure automatique ajoute du bruit, quelle que soit la valeur de son apport, et qu' il en sera toujours ainsi tant que de " vrais textes " produits par de " vrais auteurs " seront soumis à la machine, puisque les fautes de frappe ou de grammaire, les innovations stylistiques, les emprunts à d' autres langues, les néologismes sont inhérents au processus humain de pratique et de création du langage. Notre outil devra trouver trouver dans une grande liste de termes composés (l'extraction de mots composés lemmatisés s'apparente en fin de compte à l'extraction terminologique) ceux qui contiennent un mot ou groupe de mots donnés, pas nécessairement en première position, ou qui s' en dérivent par pré- (ou suf)fixation, ou qui ont une racine commune, ou plus simplement le contiennent avec une faute d' orthographe. Et cet outil devra fonctionner en temps réel, pour être vraiment utilisable.

En termes plus formels, nous avons besoin d'un indicateur d'inclusion "floue" d'une chaîne de caractères dans une autre.

3. Etat de l'art

- Consultations classiques de listes :

Signalons ici pour mémoire :

+ la consultation par ordre alphabétique, qui donne accès aux seuls termes commençant par la chaîne posée en requête.

+ le recherche d' une chaîne stricte incluse (fonction " search " des traitements de texte), qui par définition passe à côté de nombreuses variantes (lettres redoublées, fautes d' orthographe...).

+ la recherche après phonétisation, adaptée à un contexte d'application étroit (annuaires électroniques) et dépendante de la langue utilisée, ainsi que de la construction d'un modèle de fautes des utilisateurs.

- Comparaison de chaînes.

Une littérature considérable existe sur le problème de la comparaison de chaînes "voisines" ; en effet ce problème intervient dans plusieurs contextes d'applications importantes comme la correction orthographique (cf. [Kukish 92] pour une synthèse), ou la recherche d'information dans les systèmes documentaires indexés en texte intégral (donc indexés par des unitermes à multiples variantes). Dans ce dernier contexte, si [Baeza-Yates 92] fournit une revue des algorithmes de recherche de chaînes strictes, [Owolabi et McGregor 88] proposent un algorithme hybride de comparaison approchée. Plusieurs familles d'algorithmes sont à distinguer ; les plus récentes font appel aux automates finis [Oflazer 96] ou aux ensembles flous [Schneider et al. 92], et nous semblent difficilement extrapolables au problème de l'inclusion floue ; les plus répandues font appel à la programmation dynamique [Morgan 70] [Abe 82] [Du et Chang 92] .

Cette technique séduisante permet de calculer le chemin le moins coûteux en terme de nombres de subtitutions, d' ajouts et suppressions de caractères, éventuellement pondérés, pour passer d' une chaîne à une autre. Son inconvénient tient :

1) à son inadéquation à la recherche de sous-chaînes dans une longue chaîne donnée. En effet elle repose sur l'hypothèse que l'optimisation d'une chaîne de décisions peut se décomposer en une suite de décisions partielles optimales, ce qui se traduit dans notre cas par la possibilité de "démarrer" de façon erronnée le calcul de distance quand la chaîne la plus longue comporte "trop tôt" la première lettre de la chaîne à comparer.
2) à son coût de calcul : son coût croît en O(k1, k2) dans le cas de la comparaison de deux chaînes de longueur k1 et k2, donc en O(k,K) en ce qui concerne la comparaison d' une chaîne de longueur k à une liste de chaînes contenant en tout K caractères.

Si cette technique est très utilisée dans certains logiciels et processeurs spécialisés dans la reconnaissance de la parole, aucun utilitaire de consultation en temps réel de listes de termes n' y fait appel, à notre connaissance.

4. Notre méthode basée sur les N-grammes

Pour nous chaque mot de la liste ainsi que le mot constituant la requête est caractérisé par le vecteur des fréquences de ses N-grammes, c' est à dire de toutes les chaînes de longueur fixe de N caractères qui s' y trouvent.

Elle comporte deux étapes:

i) une première, de préparation en différé, consiste à extraire les bigrammes et les trigrammes de chaque terme de la liste afin de construire le vecteur des fréquences de bigrammes et trigrammes correspondant, puis à construire l' ensemble des " listes inverses " de termes pour chaque bigramme et trigramme effectivement répertorié.

ii) une deuxième, interactive, s' exécute en deux temps : 1) calcul d' un indice d' inclusion du vecteur de départ avec les vecteurs construits dans l' étape précédente, 2) filtrage des réponses obtenues en fonction de l' indicateur d' ordre des N-grammes communs.

4.1. Comparaison entre vecteurs

Dès lors que nous caractérisons un mot simple ou composé par un vecteur de fréquences de N-grammes (nos meilleurs résultats proviennent des fréquences de leurs trigrammes concaténées à celles de leurs bigrammes), on peut définir un "indice d' inclusion", non rigoureux mais validé empiriquement, décrivant le degré d' inclusion de la chaîne A dans la chaîne B par la formule :

$$I(A/B) = < \mathbf{x}_A, \mathbf{x}_B > / \| \mathbf{x}_A \|$$

où $\mathbf{x}_A$ et $\mathbf{x}_B$ sont les vecteurs de fréquences de N-grammes de A et B, $< \mathbf{x}_A, \mathbf{x}_B >$ note leur produit scalaire, et ||.|| leur norme.

Cet indicateur, indépendant de toute notion de troncature, permet de repérer la présence stricte ou approchée d' une chaîne de longueur raisonnable au milieu, début ou fin d' une autre chaîne pouvant comporter des espaces (termes composés), et ce indépendamment de la langue ou de l' écriture utilisée. Cet indicateur a ses défauts, développés dans le paragraphe suivant, mais nous avons constaté que les listes de chaînes produisant un indice d' inclusion de plus de 0,62 par rapport à la chaîne posée en requête n' " oublient " aucun terme pertinent, et ne comportent

pas un bruit excessif (les termes non pertinents sont minoritaires, et plutôt rares dans le début de la liste).

4.2. Ordre des N-grammes communs

A l' issue de la première étape, on obtient un sous ensemble de mots triés par ordre de similarité décroissante. Cependant, une mesure basée seulement sur la distance vectorielle ne prend pas en compte l' ordre des N-grammes communs. En d' autres termes, une chaîne contenant tous les trigrammes et bigrammes de la chaîne de départ mais qui ne sont pas en séquence aura la même note de comparaison que cette chaîne vis à vis d' elle même. Par exemple l' indice d' inclusion de " peintre " dans " peinture " est du même ordre que celui de " peintre " dans " interentreprise "...

Pour pallier ce problème, on filtre ce sous ensemble de mots en tenant compte de l' ordre des N-grammes dans la chaîne de départ et dans la chaîne à comparer. Ainsi, si les N-grammes communs sont en séquence dans la chaîne réponse comme dans la chaîne de départ, alors la note de proximité sera maximale.

Le calcul de notre indicateur de proximité entre la chaîne de départ et chaque terme de la liste est une somme de notes partielles, et se fait comme suit :

Pour chaque catégorie de N-grammes (bigrammes et trigrammes) :

i) pour le premier N-gramme commun, on donne par défaut la note 3 (le nombre 3 a été choisi empiriquement comme note de concordance maximale entre un N-gramme et un autre),

ii) pour les N-grammes communs suivants dans l' ordre de la chaîne de départ, la note est fonction de la distance entre un N-gramme commun et le N-gramme commun précédent dans la chaîne d' arrivée :

- si l' écart est égal à +1 (ils se suivent), alors la note globale sera incrémentée de 3,

- si l' écart est égal à +2, alors la note globale sera incrémentée de 2,

- si l' écart est égal à +3, alors la note globale sera incrémentée de 1, et

- au delà de 3 ou si l' écart est négatif, la note globale reste inchangée.

L'idée de cet indicateur est de permettre des retours en arrière dans la recherche de séquences maximales de N-grammes (ex. : *postcombustion* sera considéré comme fortement inclus dans *processus de combustion postérieure* aussi bien que dans *moteur à post-combustion*), évitant ainsi le défaut principal de la programmation dynamique.

Une fois l' indicateur de séquentialité calculé, on utilise un seuil pour supprimer les chaînes ayant des indicateurs faibles. Les expériences ont montré 1) que les chaînes produisant un indice de séquentialité de plus de 75% de la note maximale possible par rapport à la chaîne posée en requête nous sont apparues jusqu' à ce jour comme valides, 2) que cette méthode permet la "détection floue" d' un motif court au milieu d' une longue chaîne, même si ce motif se répète partiellement dans la chaîne.

4.3. Exemple :

Dans le mode : requête = chaîne incluse dans les réponses, nous avons appliqué notre algorithme à la chaîne de caractères erronnée *conaissance* sur une liste de mots composés lemmatisés issue de l'indexation automatique, peu retravaillée, de 8000 pages Web

francophones par le logiciel Nomino (cf. <www.ling.uqam.ca/nomino>), d'où la liste ordonnée suivante :

connaissance
reconnaissance vocal
connaissance militaire
connaissance technique
connaissance fondamental
connaissance stratégique
connaissance scientifique
connaissance auteur
données/base de connaissance auteur
connaissance
méconnaissance
base de connaissance
arbre de connaissance
données/base de connaissance
base de données/base de connaissance auteur
base de données/base de connaissance
naissance d Ariane

Nous n'avons pas encore implanté le mode : requête = chaîne incluant les chaînes en réponse, qui nous a été suggéré par la pratique du premier mode de requête. La requête *base de connaissances* devrait donner dans ce cas : *base de connaissance ; base de donnée ; arbre de connaissance ; connaissance de base ; connaissance technique ;* …

5. Evaluation :

Nous n'avons pas trouvé dans nos recherches bibliographiques d'algorithme s'attaquant spécifiquement à notre problème d'inclusion floue de chaîne, ni a fortiori de jeu d'essai permettant d'établir des comparaisons.

Rappelons que la méthodologie généralement admise pour l'évaluation des systèmes documentaires consiste à mesurer les taux de rappel (nombre de réponses pertinentes / nombre d'unités documentaires pertinentes) et de précision (nombre de réponses pertinentes / nombre total d'unités documentaires) pour une base documentaire donnée et un jeu de requête donné. Elle suppose de connaître "les bonnes réponses" pour chaque requête, et ce de façon exhaustive, travail considérable qui n'est réalisé que très ponctuellement dans le cadre de programmes d'évaluation comme TREC aux Etats-Unis ou Amaryllis en France. Sur le fond, l'existence de "bonnes réponses" dans l'absolu, et la question de la limite entre "bonnes" et "mauvaises" réponses est déjà un sujet de discussions dans le milieu documentaire. Les considérations développées au paragraphe 1 sur la multiplicité des points de vue éditoriaux possibles en analyse de données textuelles nous font émettre des doutes sur la possibilité de trouver un consensus dans un milieu suffisamment large pour qu'on puisse s'attaquer à la construction d'une "base d'expressions" de test et d'un jeu de questions-réponses exhaustives. Tout comme pour l'évaluation de l'utilisation des résultats d'analyse multidimensionnelle dans notre domaine, c'est encore l'impression - à la fois subjective et partagée par un milieu de recherche – de *sens* dégagé qui primera sans doute encore longtemps.

6. Comparaison de complexité avec la programmation dynamique :

Bien que notre méthode présente une complexité égale ou supérieure à celle de la programmation dynamique, elle est est plus adaptée pour une utilisation en temps réel.

En effet une grande partie de nos calculs est faite en différé dans la première phase (constitution des vecteurs, listes inverses pour les N-grammes) en O(K log(K)), alors que la deuxième phase interactive (extraction d' un sous-ensemble de termes proches, puis filtrage par indicateur de séquentialité) ne porte :

1) que sur k listes inverses de N-grammes (même définition de k et K vue plus haut),

2) que sur un sous ensemble T' de vecteurs, inférieur d' au moins un ou deux ordres de grandeur à leur nombre total.

Cette phase s' effectue donc en O(T' , k, k') où k' est le nombre moyen de caractères des chaînes à comparer.

En pratique, nous avons constaté que le temps de réponse obtenu dans le cas d' une requête adressée à une liste de 50 000 termes, dont la moitié de termes composés, était de l' ordre de quelques dixièmes de secondes sur un ordinateur de bureau standard (Pentium 350).

7. Conclusions et perspectives

Cet outil trouve naturellement sa place dans notre environnement de révision d' indexation et de cartographie textuelle Hypermap, en prélude aux phases d' analyse proprement dites.

Mais il peut être utile dans tous les cas où l' on a besoin de rechercher de façon floue et en temps réel une chaîne de caractères au sein d' une liste importante de "mots" longs répertoriés dans un système informatisé : adresses, glossaires, noms propres, listes terminologiques, annuaires, dictionnaires, moteurs de recherche...

Dans le futur proche où des listes de centaines de milliers d'expressions devront être contrôlées, l'outil de requête ponctuelle présenté ici trouvera à coup sûr ses limites ; il deviendra alors nécessaire de structurer l'ensemble du vocabulaire sur une base lexicale. L'indice d'inclusion présenté pourrait alors servir à constituer une, ou peut-être deux cartographies (une par sens d'inclusion) qui présenteraient simultanément à l'analyste une vue d'ensemble du vocabulaire, et des vues locales des familles de termes - un terme pouvant nécessairement appartenir à plusieurs familles.

Remerciements

- Au Ministère de l'Education Nationale, de la Recherche et de la Technologie pour son aide, dans le cadre de l'action "Recherche et filtrage d'information sur les réseaux".
- Aux lecteurs-évalueurs anonymes de JADT'2000, dont les remarques et suggestions nous ont amenés à préciser à la fois notre problème et l'originalité de notre solution par rapport à l'état de l'art.

Références

Abe K., Sugita N. (1982.). Distances between strings of symbols, review and remarks. In: Proc. of the 6th International Conference on Pattern Recognition, Munich.

Baeza-Yates R.A. (1992). String Searching Algorithms. In: Frakes W.B., Baeza-Yates R.A. editors, *Information Retrieval: data structures and algorithms*. Prentice-Hall.

Du M.W., Chang S.C. (1992). A Model and a Fast Algorithm for Multiple Errors Spelling correction. In: Acta Informatica, vol.(29): 281-302.

Galil Z., Giancarlo R. (1988). Data structures and algorithms for approximate string matching. In: Journal of Complexity, vol.(4): 33-72.

Halleb M., Lelu A.. (1997). Hypertextualisation automatique multilingue à partir des fréquences de N-grammes. In Balpe J.P; Lelu A., Saleh I. editors, *Hypertextes et Hypermédias, réalisations, outils et méthodes*, Hermès, Paris.

Kukish K. (1992). Techniques for automatically correcting words in texts. In: ACM Computing Surveys, vol.(24): 377-439.

Morgan H.L.(1970) Spelling corrections in system' s programs. In: Comm. ACM, vol.(13),2:90-94.

Oflazer K. (1996). Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction. In: Computationnal Linguistics, vol.(22): 73-90.

Owolabi O., McGregor D.R. (1988). Fast approximate string matching. In: Software – Practice and Experience, vol.(18), N°4: 387-393.

Rhissassi H., Lelu A. (1998). Indexation assistée et cartographie sémantique pour la génération automatique d' hypertexte. In Mojahid M. editor,*Proc. of CIDE'98,* INPT/Europia Productions, Rabat, pp.131-139.

Schneider M., Lim H., Shoaff W. (1992). The utilization of fuzzy sets in the recognition of imperfect strings. In: Fuzzy Sets and Systems, vol.(49): 331-337.

Wertel-Fournier I. (1999). Pour un imagier numérique comme espace cartographié de l'iconothèque. Thèse de l'université Paris 8.