ROBO : Une Mesure d'édition pour la comparaison de phrases Application au résumé automatique

Aurélien Bossard[1] Christophe Rodrigues[2]
(1) Université Paris 8, Laboratoire d'Informatique Avancée de Saint-Denis
(2) CNRS, UMR 7030, Laboratoire d'Informatique de Paris Nord
(1) aurelien.bossard@iut.univ-paris8.fr, (2) christophe.rodrigues@lipn.fr

Résumé. Dans cet article, nous proposons une mesure de distance entre phrases fondée sur la distance de Levenshtein, doublement pondérée par la fréquence des mots et par le type d'opération réalisée. Nous l'évaluons au sein d'un système de résumé automatique dont la méthode de calcul est volontairement limitée à une approche fondée sur la similarité entre phrases. Nous sommes donc ainsi en mesure d'évaluer indirectement la performance de cette nouvelle mesure de distance.

Abstract.

ROBO, an edit distance for sentence comparison – Application to automatic summarization
We here propose a sentence edit distance metric, ROBO, based on Levenshtein distance. This metric distance is weighted by words frequency and operation type. We apply ROBO on an automatic summarization system whose sentence selection metrics are on purpose restricted to sentence similarity approaches. ROBO performance can then be evaluated indirectly.

Mots-clés : résumé automatique, similarité sémantique, distance d'édition.

Keywords: automatic summarization, semantic similarity, edit distance.

1 Introduction

Dans cet article, nous nous intéressons à une nouvelle mesure de similarité entre phrases qui considère leur structure en utilisant un indice de surface – l'ordre des mots – et l'importance des mots. Le calcul de similarité entre phrases est un composant clé dans beaucoup de domaines du TAL, notamment la détection de plagiat et de paraphrase ou le résumé automatique. Cependant, dans ce dernier cas, la similarité entre phrases est souvent calculée à l'aide de méthodes dites de « sacs de mots », ou à l'aide de méthodes utilisant des traitements linguistiques complexes, syntaxiques et sémantiques. Alors que le premier type d'approches ne prend pas en compte la structure des phrases, la seconde est dépendante de la langue, requiert des ressources importantes et est coûteuse en temps de calcul. Nous proposons ici une mesure de similarité qui se situe à la croisée des mesures purement statistiques et de celles fondées sur une analyse linguistique profonde.

Étudier des mesures de similarité entre phrases pose le problème de leur évaluation. En effet, une mesure de similarité répond à un besoin particulier. Elles sont conçues pour mettre en avant des traits spécifiques des objets traitées pour un problème spécifique : Damerau-Levenshtein (Damerau, 1964) pour la vérification orthographique, (Smith & Waterman, 1981) pour l'alignement de séquences génétiques, la similarité cosinus (Salton & McGill, 1986) pour la classification de document ou la recherche d'information... L'évaluation d'une mesure de similarité ne peut donc pas être décorrélée de son application. Dans cet article, nous proposons d'appliquer et d'évaluer notre mesure dans le cadre du résumé automatique à base de graphe. Ce type de méthodes de résumé automatique a été introduite par (Salton *et al.*, 1997) et est très répandue parmi les résumeurs automatiques. Le calcul de la similarité entre phrases est au cœur de ces méthodes, qui sont donc un bon moyen d'évaluer notre mesure de similarité.

2 État de l'art

Afin de pouvoir comparer deux phrases entre elles, ou plus généralement deux séquences de symboles, différentes méthodes sont disponibles. Parmi les mesures les plus utilisées, nous pouvons distinguer deux principaux groupes : les méthodes dites de « sac de mots » ainsi que les distances d'édition. Dans cette section, pour étayer nos travaux, nous restreignons la présentation des distances ou mesures de similarité à leur application entre phrases, bien que celles-ci permettent de façon plus générale de comparer toute séquence de symboles. La liste des distances exposées ci-dessous n'est pas exhaustive et présente les mesures les plus utilisées. Ces mesures peuvent toutes être adaptées pour prendre en compte des n-grammes, moyennant un coût computationnel plus élevé (Damashek *et al.*, 1995).

2.1 Sacs de mots

Indice de Jaccard Cet indice permet de dénombrer les mots communs aux deux phrases et de les pondérer par le nombre total de mots différents des deux phrases. Soient les phrases p_1 et p_2, l'indice de Jaccard est défini par :

$$J(p_1, p_2) = \frac{|p_1 \cap p_2|}{|p_1 \cup p_2|}$$

Cosinus Alors que l'indice de Jaccard repose sur une représentation ensembliste des phrases, la méthode cosinus utilise une représentation vectorielle. Une phrase devient alors un vecteur dont chaque dimension contient le poids d'un mot, généralement sa fréquence ou un *tf.idf*. Comparer deux phrases revient alors à comparer l'angle entre deux vecteurs.

$$Cosinus(p_1, p_2) = \sum_{w \in p_1 \cap p_2} V(w, p_1) \cdot V(w, p_2)$$

WordOrder (Li *et al.*, 2006) a développé une mesure fondée sur un modèle vectoriel. Chaque dimension des vecteurs qui représentent les phrases reçoit la position d'un mot dans la phrase. La norme de la différence entre deux vecteurs ainsi formés, normalisée par leur somme donne la similarité entre deux phrases. Des informations sémantiques issues de corpus et de ressources sémantiques affinent le calcul.

2.2 Distances d'édition

Jaro La distance de Jaro (Jaro, 1989) permet de prendre en compte dans une certaine mesure la position respective des mots identiques aux deux phrases (éloignement). Conceptuellement, un mot commun aux deux phrases mais situé en début d'une phrase et à la fin de l'autre sera fortement pénalisé (premier et deuxième terme de l'équation). De plus, les mots communs identiques se trouvant aux mêmes positions dans les deux phrases augmentent la similarité (troisième terme de l'équation). Plus la distance de Jaro est élevée (au maximum égale à un), plus les phrases sont similaires.

$$Jaro(p_1, p_2) = \frac{1}{3}\left(\frac{|m|}{|p_1|} + \frac{|m|}{|p_2|} + \frac{|m|-t}{|m|}\right)$$

- $|p_i|$, le nombre de mots de la phrase i ;
- $|m|$, le nombre de mots identiques (dont l'éloignement n'atteint pas la moitié de la longueur de la plus longue phrase) ;
- t, le nombre de transpositions : le nombre de fois où des mots différents sont présents à des positions identiques.

La distance de Jaro-Winkler (Winkler, 1999) est une adaptation de la distance de Jaro permettant de favoriser les phrases commençant par des mots identiques.

$$JW(p_1, p_2) = Jaro(p_1, p_2) + (lp(1 - Jaro(p_1, p_2))$$

- l, la longueur du plus grand préfixe commun aux deux phrases
- p, un coefficient fixé à la valeur 0.1.

Ainsi, plus les phrases sont similaires à leur début (l grand), plus la similarité obtenue est grande.

Levenshtein La distance de (Levenshtein, 1966) repose sur le calcul du nombre minimal d'opérations d'édition nécessaires pour transformer une phrase p_1 en une phrase p_2 : insérer ou supprimer un mot, substituer un mot par un autre. Le nombre minimal d'opérations nécessaires pour changer p_1 en p_2 représente alors la distance entre les deux phrases. Cette distance utilise une technique de programmation dynamique. Une solution optimale d'un problème est trouvée à partir de solutions optimales de sous-problèmes qui le composent.

Une matrice Lev de taille $(|p_1|+1) \times (|p_2|+1)$ est d'abord initialisée : chaque case i de la première ligne de Lev représente le coût nécessaire pour transformer les i premiers mots de p_1 en une phrase vide. De même, chaque case j de la première colonne de Lev représente le coût nécessaire pour transformer la phrase vide en les j premiers mots de p_2.

Une fois la matrice Lev initialisée, on reporte dans l'ordre des indices i et j pour chaque case vide de la matrice, la valeur minimale parmi les cases adjacentes antérieures :
– la valeur de la case de gauche, de coordonnées $(i-1, j)$ incrémentée de un, correspondant à une suppression ;
– la valeur de la case du dessus, de coordonnées $(i, j-1)$ incrémentée de un, correspondant à une insertion ;
– et la valeur de la case en diagonale, de coordonnée $(i-1, j-1)$ correspondant à une substitution, incrémentée de un seulement si le $(i-1)^{\text{ème}}$ mot de p_1 est différent du $(j-1)^{\text{ème}}$ mot de p_2.
Le processus est réitéré pour chaque case restante du tableau jusqu'à parvenir à la dernière case contenant le nombre d'opérations minimal pour changer p_1 en p_2. Comme toute la matrice doit être parcourue entièrement mais seulement une fois, l'algorithme introduit par (Levenshtein, 1966) possède une complexité quadratique relative à la taille des phrases.

Des variantes de la distance de Levenshtein comme Smith-Waterman (Smith & Waterman, 1981) ou Smith-Waterman-Gotoh (Gotoh, 1982) existent, issues de travaux sur l'alignement de séquences biologiques. Elles ont pour objectif principal d'identifier efficacement les sous-séquences communes.

3 La mesure ROBO

Distance générale, la distance de Levenshtein peut être utilisée sur différents types de problèmes. Celle-ci permet de prendre en compte des éléments de structure des séquences comme par exemple l'ordre des symboles, contrairement à des méthodes « sac de mots ». De plus, elle possède l'avantage de distinguer explicitement les différentes opérations permettant de passer d'une séquence à une autre. Enfin, la fonction de coût associée à la substitution de deux symboles peut être directement adaptée au problème à traiter. Néanmoins, dans certains contextes, il peut être judicieux d'adapter son comportement afin de prendre en compte des caractéristiques particulières au problème à traiter.

Par exemple, pour le système de résumé automatique sur lequel repose l'évaluation de notre distance et que nous présentons en §4.1, il est important de rendre compte le plus fidèlement possible à la fois des similarités structurelles de deux phrases et de l'importance, relative au contexte des documents à résumer, des mots qui les composent. C'est pourquoi nous proposons d'adapter la distance de Levenshtein afin de prendre en compte une pondération des mots qui reflète leur importance au sein des textes, ainsi que la pondération des opérations d'édition afin de rendre compte le plus fidèlement possible de la structure des phrases.

L'idée d'adapter la distance de levenshtein à des problèmes spécifiques a déjà été abordée dans la littérature comme par exemple pour la reconnaissance vocale en prenant en compte des poids pour chaque transcription phonétique (Ziolko *et al.*, 2010) ou encore (Barat *et al.*, 2010) pour la classification d'images.

L'algorithme 1 illustre les modifications apportées à la distance de Levensthein. Dès l'initialisation, le poids de chaque mot est pris en compte en place d'une valeur unitaire constante. La fonction de coût utilisée demeure une fonction affine usuelle ; ici, la fonction moyenne permet lors de la substitution de deux mots de prendre en compte à part égale leur poids dans leur phrase respective. Enfin un facteur k est ajouté aux substitutions et permet de modifier l'importance de la fonction de coût des substitutions. Intuitivement, il peut être considéré qu'une substitution est l'équivalent d'une suppression suivie d'une insertion, ce qui peut être pris en compte en fixant $k = 2$. Si tous les poids sont affectés à une valeur unitaire, alors la distance entre deux phrases est identique à celle obtenue par la distance de Levenshtein d'origine. De même, la complexité de la méthode reste quadratique en le nombre de mots des phrases, dans la mesure où toute la matrice Lev doit être parcourue une fois.

4 Évaluation

Nous avons intégré la mesure ROBO à un système de résumé automatique afin de pouvoir l'évaluer indirectement, grâce à son impact sur les résultats du système de résumé par rapport à d'autres mesures de similarité. Le système de résumé a été simplifié au maximum, afin d'isoler les effets de la mesure ROBO sur les résultats obtenus. Nous utilisons pour toutes les mesures évaluées le *tf.idf* comme poids des mots.

Algorithm 1 retourne la distance de Levenshtein pondérée entre les phrases p_1 et p_2

procedure LEVENSHTEIN PONDÉRÉ(p_1, p_2)
  $Lev[0][0] \leftarrow 0$
  for $i \leftarrow 1$ to $|p_1|$ **do**
    $Lev[i, 0] \leftarrow Lev[i-1, 0] + w_{i-1}$
  end for
  for $j \leftarrow 1$ to $|p_2|$ **do**
    $Lev[0, j] \leftarrow Lev[0, j-1] + w_{j-1}$
  end for
  for $i \leftarrow 1$ to $|p_1|$ **do**
    for $j \leftarrow 1$ to $|p_2|$ **do**
      if $p_1[i-1] = p_2[j-1]$ **then** coût $\leftarrow 0$
      else coût $\leftarrow \frac{w_{i-1}+w_{j-1}}{2}$
      end if
      $suppr \leftarrow Lev[i-1, j] + w_{i-1}$
      $insert \leftarrow Lev[i, j-1] + w_{j-1}$
      $subst \leftarrow Lev[i-1, j-1] + k\times$ coût
      $Lev[i, j] \leftarrow minimum(suppr, insert, subst)$
    end for
  end for
  Return $Lev[|p_1|, |p_2|]$
end procedure

4.1 Système de résumé pour l'évaluation

Un résumé doit contenir les informations centrales des documents, mais aussi éviter la redondance et préserver la diversité informationnelle. On obtient souvent cela en appliquant un algorithme de sélection des phrases centrales suivi d'un algorithme d'élimination de la redondance. Nous appliquons ici les algorithmes bien connus LexRank (Erkan & Radev, 2004) puis MMR (Carbonell & Goldstein, 1998). L'algorithme LexRank consiste à simuler une marche aléatoire au sein d'un graphe où les nœuds sont les phrases et les arêtes les similarités entre elles. Il affecte itérativement à chaque nœud un score qui dépend du score des nœuds adjacents et de leur similarité. Le calcul des scores suivant est appliqué jusqu'à convergence :

$s(u) = \frac{d}{N} + (1-d) \sum_{v \in adj(u)} \frac{s(v)}{sim(u,v)}$

où s est le score LexRank, u un nœud, adj(u) les nœuds adjacents à u, sim(u,v) la similarité du nœud u avec le nœud v, et N le nombre total de nœuds.

LexRank nécessite donc deux paramètres : le *damping* (probabilité de saut aléatoire), et ϵ (seuil de convergence).

L'algorithme MMR consiste à donner un score à chaque phrase itérativement, en fonction d'un score de centralité et de la similarité aux phrases sélectionnées dans les étapes précédentes. La phrase qui maximise le score MMR est ajoutée au résumé, tant qu'elle n'amène pas le résumé à dépasser la taille maximum autorisée. L'algorithme est répété jusqu'à ce qu'aucune phrase ne vérifie cette dernière contrainte.

$$MMR = \underset{P_i \in D \setminus S}{\operatorname{argmax}} \left[\lambda score(P_i) - (1-\lambda) \underset{P_j \in S}{\operatorname{argmax}}\ sim(P_i, P_j) \right]$$

où D l'ensemble des phrases à résumer, S l'ensemble des phrases déjà sélectionnées, λ le facteur de nouveauté, $score$ l'évaluation de la pertinence d'une phrase et sim une similarité entre deux phrases.

A chaque lancement du système de résumé, les algorithmes LexRank et MMR se fondent sur la même mesure de similarité entre phrases. La mesure ROBO est une mesure de distance, et est comprise entre 0 et la somme des poids des mots de chacune des deux phrases. Nous avons donc normalisé la distance ROBO d'après la formule suivante :

$$sim_{ROBO}(P_1, P_2) = 1 - (ROBO(P_1, P_2) / \sum_{m \in P_1, P_2} poids(m))$$

4.2 Comparaison avec d'autres mesures de similarité

Nous comparons ici l'impact des mesures de similarités suivantes, toutes testées en pondérant les mots par leur *tfidf* : Cosinus, Jaccard, Jaro, Levenshtein, WordOrder (pas d'ajout d'informations sémantiques dans notre cas).

Corpus d'évaluation Nous avons évalué nos mesures sur le corpus RPM2, qui est à notre connaissance le seul corpus libre (licence GPL) en français disposant de résumés de références réalisés manuellement (de Loupy *et al.*, 2010). Il est constitué, à l'instar des corpus des campagnes DUC 2007 et TAC de 2008 à 2010, d'une partie dédiée au résumé classique et d'une partie consacrée au résumé de mise à jour, tâche plus complexe qui nécessite notamment de gérer la temporalité. Nous nous somme intéressé uniquement à la partie dédiée au résumé classique afin d'évaluer le plus précisément possible l'apport de notre mesure de distance. Cette partie est constituée de 200 dépêches de presse regroupées en 20 thèmes de

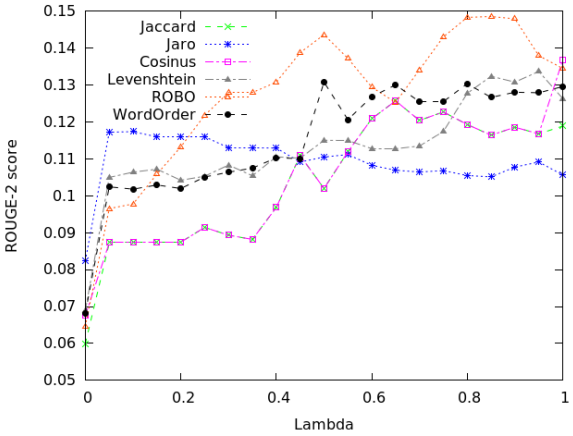

FIGURE 1: Résultats pour différentes mesures de similarités

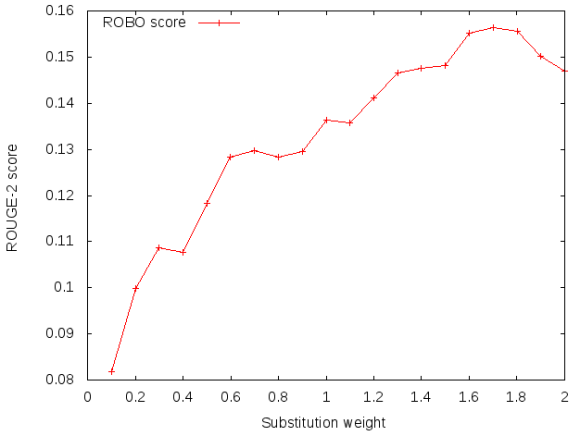

FIGURE 2: Scores ROBO selon le paramètre k, λ fixé à 0.8

dix documents. Chaque thème est pourvu de quatre résumés de référence de 100 mots maximum écrits manuellement. Les documents sont composés en moyenne de 230 mots. Nous avons utilisé les cinq premiers groupes de documents comme corpus de test, et les quinze derniers pour l'évaluation.

Paramètres expérimentaux Nous avons employé les paramètres suivants, conseillés dans l'article d'origine (Erkan & Radev, 2004). *dampling* : 0.15 ; ϵ : 0.000001 ; poids des substitutions (ROBO) : 1.5. La taille maximum des résumés a été fixée à 100 mots afin de les évaluer par rapport aux résumés de référence de RPM2, qui sont également de 100 mots maximum.

Mesures d'évaluation Les mesures ROUGE, entièrement automatiques, permettent de discriminer efficacement des systèmes procédant uniquement à de l'extraction de phrases, sans traitement supplémentaire risquant d'altérer la lisibilité et la cohérence des résumés générés (Lin, 2004) [1]. Nous utilisons donc ici ROUGE qui répondent bien à nos besoins de rapidité d'évaluation et de corrélation aux jugements humains de pertinence des résumés (corrélation de Pearson de 0.96 sur une tâche similaire (Lin, 2004)).Les mesures ROUGE procèdent par comparaison de n-grammes entre résumés de référence et résumés automatiques. Nous évaluons les résumés d'après la variante ROUGE-2 (comparaison de bigrammes), la plus corrélée aux évaluations humaines pour notre cas (génération de résumés de 100 mots) d'après (Lin, 2004). L'outil d'évaluation ROUGE est « orienté rappel », puisque d'une part les résumés sont limités en nombre de mots et d'autre part, malgré la multiplicité des références, une information n'apparaissant pas dans une des références n'est pas forcément incorrecte.

5 Résultats

La figure 2 montre l'évolution des scores de résumé avec la mesure ROBO selon le poids donné aux opérations de substitution. Pour cette expérience, le paramètre λ de MMR a été fixé à 0.8. Passée la valeur 1, qui correspond à la mesure $\text{Levenshtein}_{\text{tf.idf}}$, les scores des résumés montrent une amélioration notable jusqu'à atteindre un palier aux alentours de la valeur 1.5. Cela correspond à l'idée intuitive suivante : les substitutions doivent être pénalisées moins qu'un couple (insertion, suppression), mais plus qu'une insertion ou suppression simple.

La figure 1 présente les scores ROUGE obtenus par toutes les mesures de similarité citées en §4.2 en faisant varier le paramètre λ de MMR. Ce paramètre gère l'élimination de la redondance ; plus il est proche de 1, moins la redondance compte dans la constitution du résumé. A 0, le score LexRank n'est plus pris en compte et la génération d'un résumé dépend uniquement de l'élimination de la redondance. Les résultats montrent que la mesure Cosinus permet d'obtenir de meilleurs résultats pour λ égal à 1. En revanche, et Levenshtein et ROBO améliorent les résultats lorsque le système procède à l'élimination de la redondance. Le score ROUGE pour λ fixé à 0.8 (paramètre conseillé dans (Carbonell & Goldstein, 1998)) montre une amélioration de 14% par rapport au score obtenu avec la mesure Cosinus, et de 9% par

1. Il est actuellement impossible d'évaluer la qualité linguistique et la cohérence d'un résumé sans expertise humaine.

ROBO, score ROUGE-2 : 0.22581
" Ca arrive tous les jours " Jérôme Kerviel, qui serait à l'origine de cette fraude, est aujourd'hui introuvable.
Après des perquisitions au domicile de Jérôme Kerviel et à la Société générale, le jeune homme a été mis en garde à vue.
La Société générale a révélé jeudi avoir été victime d' une fraude colossale, portant sur 40 à 50 milliards d'euros de positions, entraînant une perte de 4,9 milliards.
A la question Yahoo " Que pensez -vous de Jérôme Kerviel, le trader qui à fait perdre des milliards à la Société Générale ?
La Société Générale a déposé plainte contre Jérôme Kerviel.

Cosinus$_{\text{tfidf}}$, score ROUGE-2 : 0.19523
Une perquisition a été effectuée mercredi au domicile parisien du frère de Jérôme Kerviel, trader de la Société générale accusé d'être à l'origine d'une perte record de la banque de près de cinq milliards d'euros, a constaté une journaliste de l'AFP.
Des perquisitions ont eu lieu vendredi soir au siège de la Société Générale à La Défense, suite à la "fraude" massive qui a fait perdre à la banque près de 5 milliards d'euros, a indiqué samedi à l'AFP une porte-parole de la banque.
La Société Générale a déposé plainte contre Jérôme Kerviel.

FIGURE 3: Résumés comparés du Cluster 7 de RPM2 avec les mesures ROBO et Cosinus$_{\text{tfidf}}$

rapport au score obtenu avec la mesure Levenshtein pondérée par le *tf.idf*, et ce dans les mêmes conditions ($\lambda = 0.8$). La figure 3 donne en exemple deux résumés, l'un généré d'après la mesure de similarité Cosinus$_{\text{tfidf}}$, l'autre d'après la mesure ROBO, du cluster 7 du corpus RPM2 relatif aux débuts de l'affaire Kerviel.

6 Discussion et perspectives

Dans cet article, nous avons montré l'intérêt de l'intégration d'une double pondération dans une mesure de distance d'édition pour une application de résumé automatique. Testée sur un ensemble reconnu de données libres (le corpus RPM2), la mesure ROBO a permis d'améliorer nettement la qualité des résumés générés, par rapport aux mesures « sacs de mots » ou à des distances d'édition non pondérées ou pondérées par l'importance des mots uniquement. Contrairement à des mesures de similarité qui exploiteraient des analyses syntaxiques lourdes, la mesure ROBO conserve un coût quadratique pour rendre compte uniquement de la structure de surface des phrases. Elle est générique et peut donc être directement adaptée à n'importe quelle langue ou corpus.

Etant donné les performances de la mesure ROBO sur l'évaluation de similarité inter-phrastique pour du résumé automatique, surtout en ce qui concerne l'élimination de la redondance, celle-ci pourrait être évaluée sur d'autres domaines du TAL, notamment la détection de paraphrases, ou l'évaluation automatique de résumé. Elle pourrait également être intégrée à des systèmes de résumé automatique plus complexes car ce système dépendant uniquement des similarités entre phrases nous a montré la pertinence de ROBO dans ce cadre.

Les travaux présentés dans cet article sont préliminaires, et de nombreuses pistes d'amélioration et d'évaluations complémentaires restent à explorer.

La mesure ROBO s'est avérée performante sur un corpus d'évaluation en français ; une évaluation sur un corpus multilingue permettra de valider l'approche sur d'autres langues qui arborent des structures différentes. Les comparaisons proposées dans l'article ne sont pas exhaustives et peuvent être étendues.

La mesure ROBO prend en compte la structure de surface des phrases et l'importance des mots, mais pas leur sémantique. Intégrer à la mesure ROBO des similarités sémantiques entre mots est une piste d'évolution qui nous semble naturelle.

Références

BARAT C., DUCOTTET C., FROMONT E., LEGRAND A.-C. & SEBBAN M. (2010). Weighted symbols-based edit distance for string-structured image classification. In *Machine Learning and Knowledge Discovery in Databases*, volume 6321 of *Lecture Notes in Computer Science*, p. 72–86. Springer Berlin Heidelberg.

CARBONELL J. & GOLDSTEIN J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98 : Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, p. 335–336, New York, NY, USA : ACM.

DAMASHEK M. *et al.* (1995). Gauging similarity with n-grams : Language-independent categorization of text. *Science*, **267**(5199), 843–848.

DAMERAU F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, **7**(3), 171–176.

DE LOUPY C., GUÉGAN M., AYACHE C., SENG S. & TORRES MORENO J.-M. (2010). A french human reference corpus for multi-document summarization and sentence compression. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

ERKAN G. & RADEV D. R. (2004). Lexrank : Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*.

GOTOH O. (1982). An improved algorithm for matching biological sequences. *Journal of molecular biology*, **162**(3), 705–708.

JARO M. A. (1989). Advances in record linking methodology as applied to the 1985 census of tampa florida. In *Journal of the American Statistical Society*, volume 64, p. 1183–1210.

LEVENSHTEIN V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, **10**, 707.

LI Y., MCLEAN D., BANDAR Z., O'SHEA J. & CROCKETT K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, **18**(8), 1138–1150.

LIN C.-Y. (2004). Rouge : A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, p.1̃0.

SALTON G. & MCGILL M. J. (1986). *Introduction to Modern Information Retrieval*. New York, NY, USA : McGraw-Hill, Inc.

SALTON G., SINGHAL A., MITRA M. & BUCKLEY C. (1997). Automatic text structuring and summarization. *Inf. Process. Manage.*, **33**(2), 193–207.

SMITH T. F. & WATERMAN M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, **147**(1), 195–197.

WINKLER W. E. (1999). *The state of record linkage and current research problems*. Rapport interne RR/1999/04, Statistics Research Division, U.S. Bureau of the Census.

ZIOLKO B., GALKA J. & SKURZOK D. (2010). Speech modelling using phoneme segmentation and modified weighted levenshtein distance. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, p. 743–746.