Création de snippets : une application de la génération automatique de résumés

Liana Ermakova*, Nicolas Faessel**

*IRIT - UMR 5505, Université de Toulouse, France,
Université Nationale de Recherche d'État de Perm, Russie
liana.ermakova@irit.fr
**IRIT - UMR 5505, Université de Toulouse, France
nicolas.faessel@irit.fr

Résumé. Face à l'immense volume de documents renvoyé par les moteurs de recherche sur le Web, les utilisateurs sont aidés dans leur tâche de sélection de documents par de courts extraits d'une à deux phrases associés à l'URL de chaque document renvoyé par le moteur de recherche, nommés *snippets*. Dans cet article, nous considérons la génération automatique de snippets comme une tâche de résumé automatique de documents, dans le contexte d'un requête donnée. Selon notre approche, les phrases des documents susceptibles de résumer un document par rapport à une requête sont représentées par différents vecteurs prenant en compte leur contexte local et leurs caractéristiques linguistiques. Afin d'assurer la qualité de l'information fournie par les snippets en respectant la contrainte de faible longueur, nous proposons deux algorithmes pour la sélection des passages candidats. Cette approche a été testée sur la collection de la tâche de génération de snippets d'INEX 2011.

1 Introduction

Les moteurs de recherche sur le Web donnent souvent comme résultat un grand nombre de documents qu'il est impossible de lire dans leur totalité. Afin d'aider l'utilisateur à déterminer si une page Web est pertinente pour une requête sans avoir à cliquer sur le lien de cette page, les moteurs de recherche fournissent un *snippet*. Un snippet est un extrait textuel du document, provenant de son contenu ou de ses méta-données (Turpin et al., 2007) et apparaissant sous le lien pointant vers une page résultant d'une recherche. Idéalement, un snippet fournit à l'utilisateur l'information qu'il recherche. De bons snippets devraient se baser sur de petites unités d'information limitées en taille telles qu'une phrase ou encore un élément XML ou (X)HTML et ils devraient permettre de distinguer les documents qu'ils représentent des autres résultats fournis par le moteur de recherche (Huang et al., 2008).

Dans cet article, nous considérons la génération de snippets comme une tâche de résumé automatique mono-document. En effet, les snippets sont des extraits courts, généralement de moins de 300 caractères, qui doivent représenter le contenu du document, indépendamment ou non d'une requête d'utilisateur. Notre approche permet une génération contextuelle de snippets, qui dépend directement des requêtes de utilisateurs : pour une requête donnée, il s'agit

de fournir à l'utilisateur un snippet qui lui permet de savoir si le document peut répondre à son besoin d'information. Nous générons les snippets à partir de phrases ou de passages du document, en proposant une représentation multi-vectorielle prenant en compte : (1) le besoin d'information de l'utilisateur exprimé par une requête (contexte global) (2) les caractéristiques linguistiques du contenu du document, (3) la structure du document, (4) le contexte local des passages candidats à la génération de snippets, qui correspond à l'influence de leurs phrases voisines.

Compte tenu du fait que les snippets doivent être le plus informatif possible malgré leur petite taille, nous proposons, dans le cas de la sélection de passages, d'utiliser deux algorithmes : une approche dynamique pour résoudre le « problème du sac à dos » (Kellerer et al., 2004) et un algorithme de fenêtre glissante (MV). Le problème du sac à dos consiste à remplir un sac dont la capacité en terme de poids est limitée, avec des éléments qui ont une valeur et un poids. Le but est de maximiser la valeur des éléments du sac en respectant la limite de poids. Ce problème peut être appliqué dans notre contexte comme suit : sélectionner le sous-ensemble de phrases candidates dont la similarité à une requête donnée est maximale et le poids total (qui correspond ici au nombre de caractères des phrases) est inférieur ou égal à un seuil prédéfini. Toutefois, l'information la plus pertinente peut se trouver dans une phrase qui dépasse à elle seule le seuil. Pour résoudre ce problème, nous proposons un algorithme de fenêtre glissante, qui permet de rechercher des passages candidats dans des phrases. Ainsi, si les phrases qui contiennent une information pertinente sont trop grandes, l'algorithme de fenêtre glissante va permettre la sélection d'un passage pertinent, sans tenir compte de sa position de départ dans la phrase, et donc réduire le passage candidat pour la génération de snippets. Le problème de cette approche est qu'elle peut abaisser le niveau de lisibilité du snippet. Notre but est donc de combiner la capacité informative du snippet avec sa lisibilité.

L'article est organisé comme suit : la section 2 présente un état de l'art sur les approches de résumé automatique et la génération de snippets. Puis, dans la section 3 nous décrivons notre approche pour la recherche de passages candidats et la génération de snippets. Cette approche est évaluée dans la section 4 sur la collection d'INEX 2011. Enfin la section 5 conclut cet article.

2 Travaux connexes

Un snippet provenant du moteur de recherche Google est défini comme suit : « Un snippet est la description ou un extrait d'une page Web, qui suit le titre et précède l'URL et le lien Cache » (Spencer, 2010).

Certains moteurs de recherche fournissent des informations détaillées pour certaines requêtes spécifiques (snippets enrichis). Par exemple, Google gère des snippets enrichis basés sur les Microdate, Microformats et les RDFa pour les types de contenu suivants : personnes, produits, organisations, recettes ...

Les descriptions contenues dans les méta-données sont couramment utilisées pour la génération de snippets. Toutefois, les moteurs de recherche peuvent pénaliser les descriptions de mauvaise qualité (ayant par exemple une mauvaise mise en forme, trop de mots clés, des informations redondantes dans la description, le titre et le contenu, etc.) (Spencer, 2010). Un moteur de recherche peut aussi générer un snippet, non pas à partir du contenu de la page, mais à partir de la description qu'il en a (Yahoo Dictory or DMOZ) (Slawski, 2009). De plus,

Yahoo se base aussi bien sur une pertinence dépendante des requêtes que sur la capacité d'un passage à résumer le document dont il provient, indépendamment de toute requête (Kanungo et Metzler, 2009). Cette capacité est estimée à partir de caractéristiques indépendantes de la requête telles que la position du passage dans le document, le nombre de termes communs entre le passage et le titre du document, etc. Lorsque la pertinence d'un passage est dite dépendante d'une requête, elle correspond bien souvent à la similarité entre la requête et le passage (généralement calculée à partir du nombre de termes que le passage et la requête ont en commun). En plus de leurs fréquences, la distance entre les termes semble être une caractéristique importante (Wang et al., 2012). Les techniques d'expansion de requête traditionnelles comme le retour de pertinence (Leal et al., 2012),(Wang et al., 2012), (Ko et al., 2008) ou l'analyse du contexte local (Sanderson, 1998) sont utilisées pour la génération de snippets.

La lisibilité est une des propriétés clés des snippets qui doit être prise en compte par les moteurs de recherche. Kanungo et Orr (2009) suggèrent de prédire la lisibilité des snippets en appliquant des arbres de décision augmentés par le gradient (gradient boosting decison trees) pour un ensemble de caractéristiques telles que la longueur moyenne des mots, la quantité des fragments de snippets, etc. Leur étude sur le comportement des utilisateurs à travers leur clics ont montré que la lisibilité influence l'utilisateur. Clarke et al. (2007) supposent que de simples caractéristiques telles que la présence de tous les termes de la requête, la lisibilité du snippet et la longueur de l'URL peuvent significativement influencer la façon dont vont cliquer les utilisateurs. Il existe deux façons d'améliorer la lisibilité des snippets : le filtrage et la pénalisation. Le filtrage signifie que les snippets candidats qui sont illisibles doivent être filtrés, alors que la pénalisation implique que le candidat doit être pénalisé dû à sa faible lisibilité. Contrairement aux tâches de génération de résumés ou de contextualisation, l'ordonnancement des phrases n'impacte pas la lisibilité des snippets, ces derniers étant trop courts.

3 Description de la méthode

Notre méthode de génération de snippets est une adaptation de la méthode proposée par Ermakova et Mothe (2012) pour la contextualisation de tweets. Nous avons modifié les techniques de pondération et développé différents algorithmes pour l'extraction des phrases candidates à la génération de snippets.

3.1 Représentation multi-vectorielle de phrases

Un document est représenté par un ensemble de phrases. Nous modélisons une phrase comme un ensemble de vecteurs. Le premier vecteur représente les termes qui apparaissent dans une phrase. Cela correspond à une représentation unigramme. Le second vecteur correspond aux bigrammes. Les travaux de Ermakova et Mothe (2012) montrent que la comparaison d'entités nommées est efficace pour contextualiser des tweets portant sur des articles de journaux. Ainsi le troisième vecteur est composé des entités nommées identifiées dans la phrase. Au lieu de stocker les fréquences d'occurrence des composants dans chaque vecteur, nous nous contentons de stocker seulement les composants, considérant qu'un terme, un bigramme ou encore une entité nommée n'apparait rarement plus d'une fois dans une même phrase. Le fait d'effectuer une comparaison paire à paire des composants nous permet d'utiliser une représentation vectorielle creuse, c'est-à-dire ne comprenant que les composants retrouvés.

D'après Silber et Mccoy (2002), les noms fournissent l'information la plus importante. Nous proposons donc d'introduire différents coefficients permettant de faire la distinction entre l'impact des noms, des autres termes significatifs et des mots vides de sens. Ainsi, nous proposons de donner un coefficient à chaque étiquetage grammatical. Par exemple, les déterminants ont un poids nul, les noms propres ont le poids le plus fort, et les noms communs ont un poids plus fort que les verbes, adjectifs et adverbes. Les valeurs du vecteur d'unigrammes sont multipliées par le coefficient de leur valeur d'étiquetage grammatical.

La pondération des étiquettes grammaticales permet de pénaliser les anaphores pronominales non résolues et autres problèmes de lisibilité. Contrairement à la contextualisation et à la génération de résumés, nous ne pénalisons pas les phrases nominales telles que les titres, qui sont généralement très courtes et donnent une idée concise et condensée du document qui les contient. Toutefois, nous exploitons la structure des documents, en pondérant les phrases selon leur position dans les parties du document. Ainsi, les phrases provenant de résumés ont un poids plus fort que celles provenant des sections.

La sélection des phrases candidates s'effectue dans le contexte des requêtes des utilisateurs, que nous nommons ici le contexte global. Ainsi, nous générons une représentation multi-vectorielle pour chaque requête, de la même manière que pour les phrases candidates. L'appariement entre les requêtes et les phrases candidates est effectué au moyen de différentes mesures de similarité.

Pour les vecteurs unigrammes et bigrammes, nous calculons la similarité du cosinus entre une phrase et une requête (respectivement $\textit{similarity}_{\textit{unigram}}$ et $\textit{similarity}_{\textit{bigram}}$). Les vecteurs d'entités nommées sont traités différemment : pour chaque entité nommée dans une requête nous cherchons toutes les entités nommées correspondantes dans les phrases candidates. Si une requête ne contient pas d'entité nommée, toutes les phrases candidates sont considérées comme pertinentes à l'égard de ce type d'information. La similarité entre les entités nommées est notée comme suit :

$$NE_{COEF} = \frac{NE_{common} + 1}{NE_{query} + 1} \tag{1}$$

où NE_{common} est le nombre d'entités nommées apparaissant à la fois dans la phrase et dans la requête, NE_{query} est le nombre d'entités nommées de la requête. Nous lissons le résultat en ajoutant 1 au numérateur et au dénominateur : une phrase peut ne pas contenir d'entité nommée et être pourtant pertinente. Si le lissage n'était pas effectué, ce coefficient serait égal à zéro. En plus de prendre en considération les entités nommées présentes dans la phrase, nous prenons en compte les synonymes contextuels, déterminés au moyen d'un système de résolution d'anaphore et lors de la comparaison, nous choisissons le synonyme qui correspond le plus.

Cet ensemble de vecteurs nous permet de combiner les mesures de similarités obtenues pour différents types d'information. Le score final des phrases par rapport à une requête est donné par la somme pondérée de ces mesures de similarité.

3.2 Lissage en fonction du contexte local

Nous supposons que pour bien détecter une phrase candidate (appelée ici phrase cible) nous devons prendre en compte son contexte local, qui correspond aux phrases qui l'entourent.

Nous supposons que l'importance du contexte diminue au fur et à mesure que la distance augmente. Ainsi, les phrases les plus proches produisent plus d'effet sur la phrase cible que les autres phrases plus éloignées. Pour les phrases dont la distance est supérieure à k, le coefficient d'importance (c'est-à-dire le poids) est égal à zéro .

Ce système permet de prendre en compte les k phrases voisines avec un poids dépendant de leur distance de la phrase cible (équation 2). Dans ce cas, le score total R_t de la phrase cible correspond à la somme pondérée des scores des phrases voisines r_i et de la phrase cible r_0 :

$$R_t = \sum_{i=-k}^{k} w_i \times r_i \tag{2}$$

$$w_i = \begin{cases} \frac{1-w_t}{k+1} \times \frac{k-|i|}{k} & \text{si } 0 < |i| \leq k \\ w_t & \text{si } i = 0 \\ 0 & \text{si } |i| > k \end{cases} \tag{3}$$

$$\sum_{i=-k}^{k} w_i = 1 \tag{4}$$

où w_t est le poids de la phrase cible défini par l'utilisateur, w_i sont les poids des phrases du contexte k. Les poids diminuent au fur et à mesure que la distance augmente. Si la distance de la phrase dans le contexte droit ou gauche est inférieure à k, son poids est ajouté au poids w_t de la phrase cible. La contrainte exprimée dans l'équation 4 nous permet de garder la somme des poids égale à 1.

3.3 Sélection de passage

3.3.1 Problème du sac à dos

Un snippet est généralement limité à une ou deux phrases (150-300 caractères). Toutefois, il doit fournir le plus d'information possible à propos du document qu'il représente. On peut ainsi considérer la génération de snippets comme la sélection de passages d'une importance maximale et dont le poids total (c'est-à-dire la longueur) ne dépasse pas un seul prédéfini. Cela nous donne un problème classique en optimisation combinatoire : le problème du sac à dos. Ce problème est défini comme suit : étant donné un ensemble d'éléments possédant un poids et une valeur, trouver le sous-ensemble de cet ensemble permettant de remplir le sac à dos de telle façon que le poids total soit inférieur ou égal à la capacité du sac, et la valeur totale soit la plus grande possible (Kellerer et al., 2004). Nous considérons le poids comme le nombre de caractères d'une phrase, et sa similarité à la requête représente la valeur. Nous ne traitons que le problème du sac à dos 0-1 (0-1 KP), qui restreint le nombre de chaque type d'élément à zéro ou un, afin que les snippets n'aient pas de redondance d'information. Nous résolvons ce problème par la programmation de l'algorithme DP-1 avec une complexité d'exécution $o(nc)$ où n est le nombre d'éléments et c est la capacité du sac à dos (Kellerer et al., 2004).

3.3.2 Fenêtre glissante

Deux problèmes majeurs se posent lors de l'utilisation du problème du sac à dos :

1. Si chaque phrase d'un document est plus grande que le seuil prédéfini (la capacité du sac à dos), alors le snippet sera vide.
2. Cet algorithme a un temps d'exécution pseudo-polynomial.

Nous utilisons donc une fenêtre glissante pour choisir le passage ayant le meilleur score. Pour cela nous générons un nouveau passage en respectant les étapes suivantes :

1. le premier terme est enlevé du passage candidat ;
2. le terme suivant le passage candidat est ajouté tant que la taille totale du nouveau passage ne dépasse pas le seuil prédéfini ;
3. le score du nouveau passage est calculé ;
4. si le score est plus grand que le score maximal courant, il devient le nouveau score maximal.

Bien que cela permette d'améliorer le score du passage candidat, le fait que ce passage puisse commencer en milieu de phrase risque de dégrader la lisibilité. Pour éviter cela, nous proposons de pénaliser les snippets qui ne commencent pas en début de phrase.

4 Évaluation

4.1 Corpus

L'évaluation de notre système a été effectuée sur le corpus de la tâche de recherche de snippets d'INEX 2011. La collection de documents correspond à une version XML de plus de deux millions de pages provenant de Wikipédia anglais. Les expressions d'un besoin d'information sont au nombre de 50. Chaque besoin d'information contient une requête textuelle courte (titre), une requête portant sur la structure et le contenu (titre cas), une phrase de titre, une description du besoin, et une partie narrative expliquant le besoin d'information (Trappet et al., 2012).

4.2 Mesures d'évaluation

L'évaluation a été réalisée manuellement. Nous avons utilisé les mêmes techniques d'évaluation que celles utilisées dans la tâche de recherche de snippets lors de la campagne INEX 2011 (Trappet et al., 2012). Pour chaque besoin d'information, le but des évaluateurs était de déterminer si les snippets fournissaient une information suffisante concernant les documents, afin que l'utilisateur décide de la pertinence d'un document à la simple lecture du snippet correspondant. Pour cela, les évaluateurs devaient évaluer les résultats de deux manières :

– évaluation de la pertinence des documents,
– évaluation de la pertinence des snippets.

Le titre des requêtes, leur description et leur intention donnent une idée du besoin d'information de l'utilisateur. Les évaluateurs doivent parcourir les snippets et décider si le document auquel il correspond semble pertinent pour la requête seulement en lisant le snippet. La valeur de pertinence est binaire : 1 s'il semble pertinent, 0 sinon. Après cela, ils doivent lire le document entier pour juger de sa pertinence. Le jugement des documents sert de vérité terrain, permettant de comparer les jugements de pertinence des snippets.

Tout comme pour la tâche d'INEX 2011, nous avons utilisé les mesures suivantes :

– Prédiction d'exactitude moyenne (Mean accuracy prediction - MPA), qui correspond au pourcentage moyen des résultats que les évaluateurs ont correctement évalués par rapport à la vérité terrain (c'est à dire par rapport à la pertinence des documents) :

$$MPA = \frac{(TP + TN)}{(TP + FN + TN + FP)} \tag{5}$$

où TP correspond aux vrais positifs (c'est-à-dire, les snippets évalués comme pertinents et dont la vérité terrain du document correspondant l'indique comme pertinent), TN correspond aux vrais négatifs et FP et FN respectivement aux faux positifs et faux négatifs.

– Prédiction normalisée d'exactitude moyenne (MNPA), qui est le pourcentage moyen des résultats pertinents que les évaluateurs ont correctement évalués comme pertinents ainsi que les résultats non pertinents correctement évalués comme tel :

$$MNPA = 0,5 * \frac{(TP)}{(TP + FN)} + 0,5\frac{(TN)}{(TN + FP)} \tag{6}$$

– Rappel (R), qui est le pourcentage moyen des snippets pertinents évalués correctement comme tel :

$$R = \frac{TP}{(TP + FN)} \tag{7}$$

– Rappel négatif (NR), qui est le pourcentage moyen des snippets non pertinents correctement évalués comme tel :

$$NR = \frac{TN}{(TN + FP)} \tag{8}$$

– Accord positif (PA) qui est la probabilité conditionnelle d'un accord entre l'évaluateur d'un snippet et l'évaluateur d'un document, étant donné que l'un des deux a déclaré un document pertinent :

$$PA = 2 * \frac{TP}{(2 \cdot TP + FP + FN)} \tag{9}$$

– Accord négatif (NA) qui est la probabilité conditionnelle d'un accord entre l'évaluateur d'un snippet et l'évaluateur d'un document, étant donné que l'un des deux a déclaré un document non pertinent :

$$PA = 2 * \frac{TN}{(2 \cdot TN + FP + FN)} \tag{10}$$

– Moyenne géométrique (GM) du rappel et du rappel négatif :

$$GM = \sqrt{R \times NR} \tag{11}$$

4.3 Résultats

Pour chaque besoin d'information, nous avons produit une liste ordonnée de 10 documents ainsi que les snippets correspondants. Nous avons évalué deux exécutions obtenues en appliquant le problème du sac à dos (knapsack) et l'algorithme de fenêtre glissante pour la sélection

	MPA	MNPA	R	NR	PA	NA	GM
knapsack	0.81	0.81	0.76	0.86	0.80	0.83	0.81
MV	0.76	0.75	0.63	0.87	0.72	0.79	0.74

TAB. 1 – *Résultats*

de passages (MV). Les résultats présentés dans le tableau 1 montrent que l'application du problème du sac à dos donne un score bien plus élevé que la sélection par fenêtre glissante, ce malgré sa complexité de calcul.

Afin de mesurer la corrélation entre les résultats de nos deux exécutions, nous avons calculé le coefficient de contingence phi qui montre une forte corrélation entre ces variables (Everitt et Skrondal, 2010) :

$$\varphi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 1}n_{\cdot 0}}} = 0.68 \tag{12}$$

5 Conclusion

Dans cet article, nous considérons la génération de snippets comme une tâche de résumé automatique mono-document. Nous nous appuyons sur l'approche proposée par Ermakova et Mothe (2012), qui se base sur une représentation multi-vectorielle des phrases utilisant le contexte et un ensemble de paramètres de pondération dépendant de critères aussi bien linguistiques que structurels. Par ailleurs, nous proposons deux algorithmes spécifiques à la sélection de passages candidats : une approche dynamique de résolution du problème du sac à dos, et un algorithme de fenêtre glissante. Nous avons évalué nos résultats sur les données de la tâche de recherche de snippets provenant de la campagne d'évaluation d'INEX 2011. Nos résultats montrent que l'algorithme de résolution du problème du sac à dos offre de bonnes performances malgré sa complexité.

Les perspectives de travail futur concerne la recherche des critères indépendants des requêtes et d'identification de l'intention de l'utilisateur. De plus, dans notre approche, nous avons supposé que certains paramètres utilisés dans le cas du résumé multi-document doivent être adaptés au cas de la génération de snippets. En effet, la longueur très courte des snippets laisse supposer que l'ordre des phrases n'a pas d'importance pour la génération, que les phrases nominales ne doivent pas être pénalisées, que celles provenant des titres sont utiles à la création de snippets alors qu'elles ne le sont à priori pas dans une tâche de contextualisation de tweets. Nous souhaitons valider ces suppositions dans de futures expérimentations.

Références

Clarke, C. L. A., E. Agichtein, S. Dumais, et R. W. White (2007). The influence of caption features on clickthrough patterns in web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, pp. 135–142. ACM.

Ermakova, L. et J. Mothe (2012). IRIT at INEX : question answering task. In *Proceeding of the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, Volume 7424 of *LNCS*, pp. 219–227. Springer.

Everitt, B. S. et A. Skrondal (2010). *The Cambridge Dictionary of Statistics*. Cambridge University Press.

Huang, Y., Z. Liu, et Y. Chen (2008). eXtract : a snippet generation system for XML search. *VLDB 1*(2), 1392–1395.

Kanungo, T. et D. Metzler (2009). System and method for automatically ranking lines of text. Patent Application. US 2009/0292683 A1.

Kanungo, T. et D. Orr (2009). Predicting the readability of short web summaries. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, Barcelona, Spain, pp. 202–211. ACM.

Kellerer, H., U. Pferschy, et D. Pisinger (2004). *Knapsack problems*. Springer-Verlag, Berlin.

Ko, Y., H. An, et J. Seo (2008). Pseudo-relevance feedback and statistical query expansion for web snippet generation. *Inf. Process. Lett. 109*(1), 18–22.

Leal, L., F. Scholer, et J. Thom (2012). RMIT at INEX 2011 snippet retrieval track. In *Proceeding of the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, Volume 7424 of *LNCS*. Springer.

Sanderson, M. (1998). Accurate user directed summarization from existing tools. In *Proceedings of the seventh international conference on Information and knowledge management*, Bethesda, Maryland, United States, pp. 45–51. ACM.

Silber, H. G. et K. F. Mccoy (2002). Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics 28*(4), 487–496.

Slawski, B. (2009). How a search engine may choose search snippets - SEO by the sea. http://www.seobythesea.com/2009/12/how-a-search-engine-may-choose-search-snippets/.

Spencer, S. (2010). Anatomy of a google snippet. http://searchengineland.com/anatomy-of-a-google-snippet-38357.

Trappet, M., S. Geva, A. Trotman, F. Scholer, et M. Sanderson (2012). Overview of the INEX 2011 snippet retrieval track. In *Proceeding of the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, Volume 7424 of *LNCS*. Springer.

Turpin, A., Y. Tsegay, D. Hawking, et H. E. Williams (2007). Fast generation of result snippets in web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, pp. 127–134. ACM.

Wang, S., Y. Hong, et J. Yang (2012). PKU at INEX 2011 XML snippet track. In *Proceeding of the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, Volume 7424 of *LNCS*. Springer.

Génération de snippets

Summary

A search engine returns to a user a large volume of results associated to a query that it is impossible to read. Therefore, to define whether a web page is relevant or not to a query without clicking a link, a search engine provides a user with a small text passage appearing under a search result called "snippet". In this paper, we consider snippet generation as a single-document summarization task. We propose an approach to select sentences for a snippet based on multi-vector sentence representation, taking into account the influence of neighboring sentences (smoothing from local context) and linguistic features. Snippets should be as informative as possible despite they often consist of 1-2 sentences. We propose two algorithms for the candidate passage selection: dynamic programming approach to solve the knapsack problem and the moving window algorithm. The approach was tested on the collection of INEX 2011 Snippet Retrieval Track.