Gismo : Mettez un tigre dans votre moteur

Marc-Olivier Buob, Fabien Mathieu

Gismo : Mettez un tigre dans votre moteur[†]

Marc-Olivier Buob[1] et Fabien Mathieu[1]

[1]*Nokia Bell-Labs France*

La recherche de documents (pages Web, courriels, fichiers Intranet...) est une tâche à laquelle tout un chacun est confronté régulièrement. Une recherche efficace repose sur un moteur de recherche précis et bien organisé. La majorité des techniques actuelles combinent une recherche par mots-clés à des informations structurelles (ontologies, relations entre éléments) afin de classer les documents d'un corpus par pertinence. Nous présentons dans cet article un nouveau moteur de navigation, appelé Gismo (*Generic Information Search with a Mind of its Own*). Gismo exploite uniquement le contenu textuel des documents et ne nécessite ni ontologie, ni méta-données, ni pré-apprentissage. Il est ainsi possible de l'utiliser sur n'importe quel corpus sans faire d'hypothèse sur le type ou la langue des documents. Le modèle choisi et les algorithmes utilisés permettent à Gismo d'être extrêmement rapide même sur des grands corpus. Gismo permet de trouver, trier et organiser les documents par thème et par pertinence, ce qui en fait un moteur de *navigation* et non un simple moteur de recherche.

Le plan de l'article est le suivant : après une présentation de l'état de l'art, nous décrivons successivement les trois principales contributions de Gismo : tout d'abord, la définition d'une nouvelle métrique, TF-IDTF, qui permet de représenter de manière symétrique un document par un vecteur de mots et un mot par un vecteur de documents ; ensuite, l'application d'un algorithme de diffusion pour extraire les mots et documents les plus pertinents relativement à une requête ; enfin, une organisation des résultats en arbre à l'aide d'un algorithme de regroupement hiérarchique.

Nous terminons en illustrant l'intérêt de Gismo à travers un bref exemple animalier.

Mots-clefs : Traitement automatique du langage naturel, Diffusion de pertinence, Clustering

1 État de l'art

TF-IDF (*Term Frequency - Inverse Document Frequency*) est une métrique communément utilisée en traitement du langage pour estimer l'importance d'un mot en fonction de sa fréquence dans un document et de sa rareté dans un corpus. Son principe est simple : plus un mot est rare, plus il est discriminant, et donc plus il est significatif au sens de la théorie de l'information [Aiz03].

PageRank [BP98], l'algorithme à l'origine du moteur de recherche Google, a déjà été utilisé en traitement du langage pour produire des résumés extractifs de documents [OER05]. L'algorithme de D-Iteration, utilisé ici, en est une variation [HHM15]. Comme PageRank, il s'applique à un graphe pondéré orienté. Intuitivement, la D-Iteration consiste à diffuser une quantité finie de fluide sur les sommets du graphe à partir d'une distribution initiale (correspondant à une recherche) et à estimer la pertinence de chaque sommet par la quantité de fluide qui le traverse. Cette approche par diffusion (*push*) limite en pratique les calculs «autour» de la recherche. Cela accélère la performance par rapport à un calcul traditionnel de PageRank (approche *pull*)

Plusieurs algorithmes permettent d'organiser des éléments dans une structure arborescente. On peut citer par exemple GAAC (*Group Average Agglomerative Cluster*) [RM05] ou Louvain [BGLL08], qui consistent à regrouper de manière gloutonne et récursive les éléments les plus proches. Ces algorithmes ont inspiré notre méthode de clustering, qui rajoute une notion de similarité locale caractéristique.

[†]Ce travail a été réalisé au LINCS (`https://www.lincs.fr/`)

2 Plongement réciproque des documents et des mots

L'approche TF-IDF classique postule qu'un document est défini par les mots qu'il contient et qu'un mot rare dans le corpus est plus spécifique. Nous proposons dans cette section d'enrichir cette approche par le postulat symétrique : un mot est défini par son contexte, c'est-à-dire par les documents qui le contiennent, et un document concis, c'est-à-dire qui contient peu de mots, est plus spécifique.

À partir de cette idée, nous construisons un graphe biparti pondéré G reliant les documents, supposés non vides, du corpus aux mots contenus dans au moins un document. On note X (resp. Y) l'ensemble des documents (resp. mots) et on pose $n = |X|$ (resp. $m = |Y|$). L'ensemble des sommets du graphe est $X \cup Y$, et une arête est créée si, et seulement si, elle relie un document x à un mot y présent dans x.

Pour tous $x \in X, y \in x$, il s'agit à présent de préciser le poids de l'arête (x,y). On note $x.y$ le nombre d'occurrences de y dans x. Ce poids doit tenir compte de $x.y$, mais également de l'information apportée par y dans x. Nous rajoutons un troisième ingrédient, à savoir l'information que le document x apporte sur le mot y. Le produit de ces trois facteurs nous donne le poids TF-IDTF *Term Frequency - Inverse Document and Term Frequencies* de (x,y), noté $w(x,y)$ et défini par

$$w(x,y) = (1+\log(x.y))\log\left(\frac{1+n}{1+d(y)}\right)\log\left(\frac{1+m}{1+d(x)}\right), \text{ où } d(.) \text{ est le degré dans } G. \tag{1}$$

Dans (1), les deux premiers facteurs correspondent à un poids TF-IDF classique, et le troisième introduit une fréquence inverse sur les termes (ITF). La pondération logarithmique du premier facteur (fréquence) est introduite pour limiter des phénomènes de sur-représentation quand un mot est très présent dans un document. Le décalage d'une unité dans les deux derniers facteurs sert à adoucir la pondération. Il correspond à l'ajout d'un document fictif contenant tous les mots et d'un mot fictif apparaissant dans tous les documents.

En adoptant la convention $w(x,y) = 0$ si $y \notin x$, on construit alors la matrice $n \times m$ W induite par w telle que $W_{x,y} = w(x,y)$. Chaque ligne (resp. colonne) de W peut être interprétée comme représentant un document (resp. un mot) dans l'espace des mots (resp. des documents). W (resp W^t) est donc un plongement de X dans Y (resp. de Y dans X).

3 Recherche des mots et documents pertinents par diffusion

Nous allons à présent utiliser G pour extraire des mots et documents en réponse à une requête. Nous voulons sélectionner des éléments importants et proches de la requête. Nous souhaitons aussi que la recherche effectue des *associations d'idées* : si le tigre appartient au genre *Panthera* et si le lion appartient également au genre *Panthera*, la recherche doit pouvoir déterminer que le mot *lion* est pertinent par rapport à une requête *tigre* (à travers le mot *Panthera*), même si les mots *tigre* et *lion* n'apparaissent jamais ensemble dans un même document du corpus.

Tous ces éléments nous conduisent naturellement à appliquer une technique de type *PageRank*. En pratique, nous utilisons l'algorithme de diffusion appelé D-Iteration [HHM15] afin d'explorer les liens sémantiques représentés par W. L'algorithme de D-Iteration prend en paramètres une matrice stochastique A, un vecteur Z qui représente un a priori sur l'importance, et un coefficient d'atténuation $\alpha \in]0,1[$. Dans notre cas, A découle de W, Z découle des mots cherchés par l'utilisateur, et α peut être librement ajusté.

Dans le détail, A est une matrice de taille $(n+m)\times(n+m)$ définie par $A = \begin{pmatrix} 0_{n,n} & S(W) \\ S(W^t) & 0_{m,m} \end{pmatrix}$, où $S(M)$ désigne la renormalisation stochastique de M, qui consiste à diviser chaque ligne non nulle d'une matrice positive M par la somme de ses éléments. A définit une marche aléatoire sur G où chaque arête est choisie proportionnellement à son poids TF-IDTF. Le vecteur Z est construit à partir d'une requête z (un ensemble de mots) par transformation TF-IDF : pour tout document x, $Z_x = 0$, et pour tout mot y, $Z_y = (1+(\log(z.y))\log(\frac{1+n}{1+d(y)}))$ si $y \in z$, 0 sinon. L'algorithme de D-Iteration calcule à partir de ces paramètres un vecteur P qui associe à chaque sommet sa pertinence. P est défini par :

$$P = \sum_{k\geq 1} \alpha^k Z A^k \tag{2}$$

Chaque terme de la somme correspond à une marche aléatoire de longueur k issue de Z, pondérée par un poids α^k. Cela implique que la longueur moyenne des marches aléatoires que P agrège vaut $\frac{1}{1-\alpha}$. Autrement

dit, α contrôle le compromis entre similarité et importance dans le résultat : quand α est proche de 0, les marches sont courtes, et P reflète alors principalement la similarité par rapport à la requête ; quand α est proche de 1, les marches sont longues et P reflète avant tout l'importance structurelle des éléments, indépendamment de l'a priori de départ Z.

4 Organisation hiérarchique des documents

Les plus grandes valeurs de P définissent les mots et documents les plus pertinents pour une requête donnée. Dans cette section, nous montrons comment, en utilisant W et P, il est possible d'organiser structurellement les documents en arbre (l'approche vaut aussi pour les mots, *mutadis mutandis*). Nous employons une approche agrégative gloutonne inspirée des algorithmes de clustering de graphe [RM05, BGLL08] : partant d'une partition atomique (les feuilles), nous regroupons ensemble les parties similaires et itérons jusqu'à obtenir la partition triviale (la racine). La question est de savoir comment effectuer les regroupements.

Soit $X' \subset X$ un ensemble de documents à organiser, par exemple l'ensemble des k documents les plus pertinents selon P. Nous commençons par contextualiser le plongement des documents en associant à x le vecteur $C(x) = (W_{x,y}P_y)_{y\in Y}$. L'importance intrinsèque des mots de x, estimée par la ligne de x dans W, est ainsi modulée par leur pertinence relativement à la requête, qui est donnée par P. Nous utilisons le cosinus pour définir la similarité entre documents : pour tous $x,x' \in X', \mathrm{sim}(x,x') = \frac{C(x).(C(x'))^t}{||C(x)||_2||C(x')||_2}$.

Pour un document x donné, nous définissons sa similarité caractéristique $s(x)$ comme la similarité entre $x' = \mathrm{argmax}_{x'\in X'\setminus x}\mathrm{sim}(x,x')$ et $x'' = \mathrm{argmax}_{x''\in X'\setminus x'}\mathrm{sim}(x',x'')$[‡]. $s(x)$ estime l'ordre de grandeur des similarités au voisinage de x en comparant plus proche et plus proche du plus proche.

L'agrégation, contrôlée par un paramètre de résolution $r \in [0,1]$, se déroule ainsi : deux documents x et x' sont agrégés si, et seulement si, $\mathrm{sim}(x,x') \geq r\min(s(x),s(x'))$. On agrège ainsi tous les documents qui peuvent l'être, puis on remplace les documents d'un agrégat $A \subset X'$ par un nouvel élément de représentant $C(A) = \sum_{x\in A} C(x)$. Il ne reste plus qu'à recommencer jusqu'à n'avoir qu'un unique agrégat.

La résolution r permet de contrôler la finesse de l'arbre obtenu. Lorsque $r = 1$, seules les paires d'éléments mutuellement plus proches sont agrégées, et le résultat obtenu, aux cas d'égalité près, est un dendrogramme, e.g. un arbre binaire. À l'inverse, lorsque $r = 0$, tous les éléments sont agrégés en une étape et l'on obtient une étoile. Les valeurs intermédiaires permettent de limiter la profondeur de l'arbre tout en préservant les parties caractéristiques.

5 Évaluation expérimentale

Les approches proposées sont implantées dans le logiciel libre Gismo. Instructions d'installation, documentation et tutoriels sont disponibles via l'adresse `https://gismo.readthedocs.io/`. Sur un ordinateur standard, le plongement préliminaire d'un corpus de plusieurs centaines de milliers de documents se calcule en quelques minutes, et le traitement d'une requête quelques secondes.

Pour illustrer notre travail, nous utilisons Gismo sur un corpus minimaliste de $n = 260$ animaux, obtenu à partir de la page `https://en.wikipedia.org/wiki/List_of_animal_names`[§]. Nous injectons la requête *Tigre* avec un coefficient $\alpha = 0,5$. Les 20 documents les plus pertinents obtenus sont, dans l'ordre : Tigre, Siamang, Cuon d'Asie, Requin, Sanglier sauvage, Puma, Dingo, Lion, Jaguar, Léopard, Loup, Guépard, Lynx roux, Orang-outan, Singe, Éléphant, Koala, Bouquetin, Coyote, Homme. Seuls les 7 premiers éléments contiennent *tigre* dans leur description (le requin-tigre est une espèce de requin, le siamang et le tigre sont deux espèces affectées par la déforestation...). Notons que le lion est l'animal le plus pertinent dont la description ne contient pas *tigre* (de même que la description du tigre ne contient pas *lion*). La Figure 1 présente ces documents organisés avec une résolution $r = 0,8$. Nous observons que Gismo, sans aucune autre information que les mots contenus dans les documents, arrive à distinguer animaux terrestres et marins, carnivores et herbivores, les hominidés, et en particulier qu'il isole ensemble félidés et canidés Nous avons observés des résultats similaires pour les autres animaux que nous avons testés.

[‡] Quite à utiliser une règle de départage, nous supposons la fonction argmax non ambiguë.

[§] Les documents sont les résumés descriptifs fourni par Wikipedia. Nous exposons l'exemple en français, mais tous les calculs sont traités dans la langue d'origine (anglais).

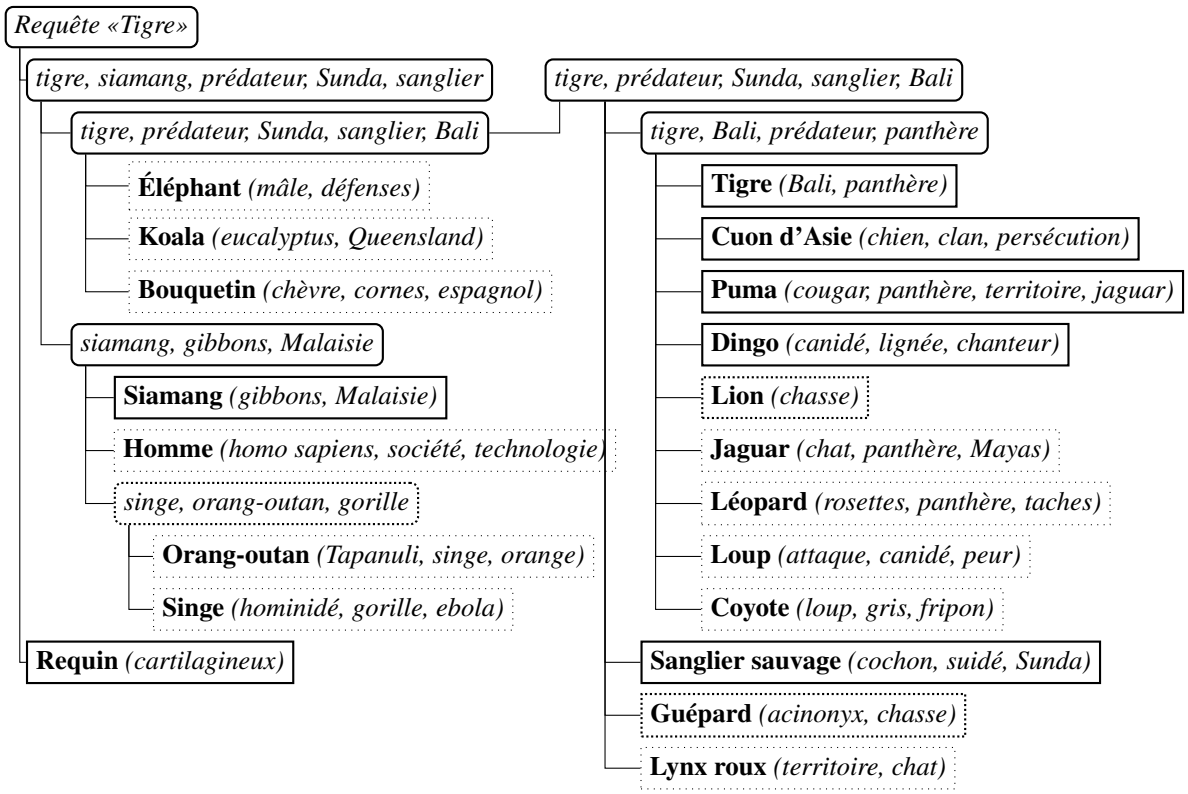

Figure 1: Représentation des résultats pour la requête «Tigre». Les animaux correspondant aux documents sont écrits en gras. Les principaux mots caractéristiques sont écrits en italique. La densité du bord des sommets reflète leur pertinence selon Gismo. Les traits pleins indiquent la présence explicite du mot «tigre».

6 Conclusion

Dans cet article, nous avons présenté un moteur de navigation appelé Gismo. Étant donné une recherche, Gismo peut trouver les documents les plus pertinents et les organiser par thème dans un arbre. Gismo ne fait aucune d'hypothèse sur le langage et la sémantique des documents et peut traiter de grands corpus. Toutes ces forces font de Gismo un moteur de navigation à la fois générique et performant.

References

[Aiz03] A. Aizawa. An information-theoretic perspective of TF-IDF measures. *Information Processing & Management*, 39(1):45–65, 2003.

[BGLL08] V. D Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, (10), 2008.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[HHM15] D. Hong, T. D. Huynh, and F. Mathieu. D-Iteration: diffusion approach for solving PageRank. *CoRR*, abs/1501.06350, 2015.

[OER05] J. Otterbacher, G. Erkan, and D. Radev. Using random walks for question-focused sentence retrieval. In *HLT/EMNLP*, pages 915–922, 2005.

[RM05] L. Rokach and O. Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.