Kodex ou comment organiser les résultats d'une recherche d'information par détection de communautés sur un graphe biparti ?

Emmanuel Navarro[*] **— Yannick Chudy**[**] **— Bruno Gaume**[*,**] **Guillaume Cabanac**[*] **— Karen Pinel-Sauvagnat**[*]

[*] *Université de Toulouse – IRIT UMR 5505 CNRS*

[**] *Université de Toulouse – CLLE-ERSS UMR 5263 CNRS*

{navarro, chudy, gaume, cabanac, sauvagnat}@irit.fr

RÉSUMÉ. *Les Systèmes de Recherche d'Information structurent en général leurs résultats sous la forme d'une liste de documents. Nous pensons qu'il existe une structure plus riche dans ces résultats. En effet, la plupart des graphes obtenus à partir de données réelles (entre autre, les graphes de documents) partagent certaines propriétés structurelles, en particulier une organisation en communautés que nous proposons d'exploiter afin de mieux organiser l'ensemble des documents restitués pour une requête. Pour ce faire, l'ensemble des documents restitués est modélisé par un graphe biparti (Documents ↔ Termes) sur lequel est appliqué notre algorithme Kodex de détection de communautés. Cet article présente Kodex et son évaluation : sur la mesure* F_1*, Kodex améliore significativement la* baseline *Okapi BM25 de 22 %.*

ABSTRACT. *Information Retrieval Systems (IRS) generally display results as a list of documents. One may think that a deeper structure exists within results. This hypothesis is reinforced by the fact that most of the graphs produced from real data (e.g., graphs of documents) share some structural properties, and in particular a community structure. We propose to use these properties to better organize the set of returned documents for a query from a given IRS. For this purpose, the retrieved document set is modeled as a bipartite graph (Documents ↔ Terms) on which the Kodex community detection algorithm is applied. This paper presents Kodex and its evaluation : regarding* F_1 *measure, Kodex overcomes* baseline *Okapi BM25 by 22%.*

MOTS-CLÉS : *Recherche d'information, graphe biparti, détection de communautés, évaluation*

KEYWORDS: *Information retrieval, bipartite graph, community detection, evaluation*

Ces travaux ont été en partie réalisés dans le cadre du programme QUAERO, financé par OSEO, agence française pour l'innovation.

1. Introduction

La liste de documents est incontestablement la forme de présentation des résultats la plus couramment utilisée par les Systèmes de Recherche d'Information (SRI). Cette liste est naturelle à la vue du paradigme usuel en recherche d'information : étant donné une requête donnée, le SRI mesure la pertinence de chacun des documents pour ne restituer que les meilleurs. Cette mesure induit donc un ordre complet entre les documents : deux documents sont toujours comparables en terme de pertinence pour une requête donnée. Toutefois, les requêtes sont souvent polysémiques car des utilisateurs différents peuvent entretenir des rapports différents avec elle, d'où des attentes différentes selon les intentions de l'utilisateur. Par exemple, pour la requête « orange », l'utilisateur peut souhaiter des documents concernant le fruit ou bien des documents concernant la ville, la couleur ou encore l'entreprise. Dès que la requête s'avère ambigüe, une liste ne permet que trois solutions possibles :

– une désambiguïsation tacite : « orange » en tant qu'entreprise est la requête la plus courante. Toutefois, est-ce donc ce que cherche l'utilisateur ?

– une désambiguïsation profilée : « orange » en tant que ville d'après le profil de l'utilisateur. Toutefois, encore faut-il connaître le profil de l'utilisateur,

– un assortiment : une liste panachée de documents concernant des interprétations très différentes de la requête. Toutefois, comment (avec une simple liste) informer l'utilisateur de cet assortiment ?

Une autre solution serait d'informer l'utilisateur de l'existence de ces multiples points de vue sur sa requête relativement à la base documentaire interrogée, l'utilisateur pouvant alors en toute connaissance de cause préciser son choix selon les informations recherchées.

Ce problème d'insuffisance de la liste de documents a déjà été de nombreuses fois soulevé en recherche d'information (Furnas *et al.*, 1987) et de nombreuses solutions ont été imaginées afin d'aider l'utilisateur à naviguer dans les résultats d'une recherche. C'est en particulier le but des techniques de clustering des résultats d'une recherche web (*web search clustering*) dans laquelle nous nous inscrivons (Carpineto *et al.*, 2009). Dans ce cadre, l'objectif principal de cet article est de présenter l'approche Kodex pour laquelle nous définissons une technique de *détection de communautés* sur un graphe biparti Documents $\leftrightarrow$ Termes afin de mettre en exergue la structure existante de l'ensemble des résultats d'un SRI pour une requête donnée.

Notre approche se nourrit d'avancées relativement récentes concernant les grands graphes de terrain. Il a été montré que la plupart des grands graphes construits à partir de données réelles (et donc les graphes modélisant des collections de documents) partagent des propriétés structurelles non triviales (Watts *et al.*, 1998; Newman, 2003), en particulier une organisation en communautés de sommets plus fortement connectés entre eux qu'au reste du graphe. Ces avancées ont engendré d'importants travaux concernant la définition formelle et la détection automatique de ces communautés, les

récents articles (Fortunato, 2010; Porter *et al.*, 2009) témoignent de l'effervescence des recherches autour de ce problème.

Cet article est organisé comme suit. Après avoir présenté les principes généraux de l'approche Kodex en section 2, nous dressons un état de l'art des systèmes de clustering de résultats en section 3, puis nous présentons en section 4 l'algorithme de détection de communautés dans un graphe uniparti que nous étendons en section 5 aux graphes bipartis Documents $\leftrightarrow$ Termes. Nous évaluons le système Kodex en section 6, puis envisageons les perspectives à ce travail en section 7.

2. Présentation générale de l'approche Kodex

Afin de permettre à l'utilisateur de préciser incrémentalement et interactivement sa requête, nous proposons l'algorithme Kodex. Ce dernier prend en entrée les n_D documents $[D_1, \ldots, D_{n_D}]$ les mieux classés par un moteur M pour une requête R sur une collection Ω, puis fournit en sortie $\{(C_1, W_1), (C_2, W_2), \ldots, (C_l, W_l)\}$, où les $C_i \subseteq \{D_1, \ldots, D_{n_D}\}$ sont des ensembles de documents et les W_i sont des ensembles d'étiquettes (*labels*) représentant la thématique de C_i. Les ensembles d'étiquettes $\{W_1, W_2, \ldots, W_l\}$ sont alors présentés à l'utilisateur sous forme de l nuages de mots, chaque nuage W_i représentant l'ensemble de documents C_i. S'il existe un cluster de documents $C_a \subseteq \{D_1, \ldots, D_{n_D}\}$ dont la pertinence pour l'utilisateur est identifiable par ses étiquettes associées W_a, alors l'utilisateur peut examiner directement les documents du cluster C_a, ou bien ré-interroger le moteur M avec une requête et le focus $(R, (C_a, W_a))$ identifié, le moteur pouvant ainsi favoriser les documents relativement au focus (C_a, W_a) dans son calcul de pertinence des documents.

Nous nous concentrons dans cet article sur la qualité des clusters de documents C_i. Il faut en effet qu'il existe au moins un cluster pertinent C_a contenant un maximum de documents pertinents pour l'utilisateur. Cette hypothèse est nommée *Cluster hypothesis* dans l'état de l'art (Jardine *et al.*, 1971), et a servi de base à de nombreux travaux. L'ideal est que le cluster C_a soit constitué de tous (et seulement tous) les documents de $\{D_1, \ldots, D_{n_D}\}$ qui sont pertinents pour l'utilisateur, les autres clusters ne contenant aucun document pertinent pour l'utilisateur. Nous cherchons donc dans cet article à évaluer la pertinence des clusters de documents, l'évaluation des étiquettes constituant une future étape à ce travail.

3. État de l'art du clustering appliqué à la recherche d'information

L'intention de cette section est de replacer l'algorithme de détection de communautés qui est au cœur de notre approche par rapport aux techniques employées dans l'état de l'art. Cela nous permet de mettre en exergue les difficultés spécifiques au clustering des résultats d'une recherche de documents (et recherche web en particulier)

par rapport au problème général de clustering (ou de détection de communautés). Pour un état de l'art plus complet, nous renvoyons à l'article de Carpineto *et al.* (2009)[1].

Le clustering des résultats d'un SRI a été introduit pour la première fois par Cutting *et al.* (1992). Les clusters y étaient construits en utilisant *Fractionation*, algorithme de clustering hiérarchique basée sur la modélisation vectorielle des documents (Salton *et al.*, 1975), introduit avec le système *Scatter/Gather*. Son principal avantage résulte de sa complexité linéaire. En revanche, il ne permet pas de traiter le problème de la polysémie, chaque document étant associé à un seul cluster. De plus, le nombre de clusters produits est un paramètre fixé par l'utilisateur, ce qui nous semble peu réaliste. Ce nombre de clusters ne doit-il pas plutôt dépendre de la polysémie de la requête ? N'existe-t-il pas des requêtes pour lesquelles une organisation des résultats en plusieurs groupes n'a aucun sens ? Nous pensons que ce sont là des points importants qui ont souvent été mis de côté dans les techniques proposées dans l'état de l'art. Enfin c'est un algorithme centré sur les données – *data-centric*[2], pour reprendre la classification de Carpineto *et al.* (2009) – et l'étiquetage des clusters n'est pas forcément pertinent et interprétable par l'utilisateur (pour chaque cluster, les termes les plus fréquents sont utilisés comme étiquettes).

Une approche très différente a ensuite été proposée avec *Grouper* (Zamir *et al.*, 1999; Zamir *et al.*, 1998). Ce système repose sur l'algorithme *STC* (*Suffix Tree Clustering*) qui rassemble les documents sur la base d'une seule propriété bien choisie : *STC* regroupe les documents qui partagent des expressions suffisamment fréquentes (dans l'ensemble des extraits de documents). L'astuce principale de *STC* repose sur l'utilisation d'un arbre de suffixe généralisé pour rechercher les séquences de termes les plus fréquentes en un temps linéaire. L'intérêt d'une telle approche est que les clusters ont une justification clairement lisible pour l'utilisateur final – *description-aware* selon Carpineto *et al.* (2009). En revanche, nous doutons de la robustesse de telles méthodes, il suffit en effet de changer l'ordre de deux mots ou de remplacer un mot par un synonyme pour que les séquences les plus fréquentes changent complètement.

Depuis ces deux travaux de référence, de nombreuses autres contributions ont été proposées, dont : *SHOC* (Zhang *et al.*, 2004), *SnakeT* (Ferragina *et al.*, 2005), *Lingo* (Osinski *et al.*, 2005) et *Noodles* (Mecca *et al.*, 2007). Il est assez remarquable que la plupart de ces méthodes (toutes celles citées ici sauf *SnakeT*) se basent sur la technique de *Latent Sementic Indexing* (*LSI*) introduite par Deerwester *et al.* (1990). C'est compréhensible car, d'une part, en prenant en considération la co-occurence des termes, la *LSI* répond au problème de la synonymie (Manning *et al.*, 2008, page 378) ; d'autre part, la complexité importante de cette méthode est limitée par la taille des données réduites au sous-ensemble de documents restitués par un SRI. Notre approche

1. Notons tout de même que les travaux (Boley *et al.*, 1999; Dhillon, 2001; Mecca *et al.*, 2007; Chen *et al.*, 2008) ne sont pas cités dans cette revue.
2. Selon la dénomination de Carpineto *et al.* (2009), les approches de clustering *data-centric* sont centrées sur la qualité des clusters ne se préoccupant pas de l'interprétabilité des clusters, alors qu'au contraire les méthodes *description-aware* sont centrées sur l'interprétabilité des clusters par l'utilisateur.

répond aussi au problème de la synonymie[3] pour une complexité en revanche plus faible que la *LSI*.

La majeure partie des approches – toutes sauf *Scatter/Gather* (Hearst *et al.*, 1996) et *Noodles* (Mecca *et al.*, 2007) – sont basées sur les extraits de documents plutôt que sur le texte complet. C'est un intérêt évident au regard de la complexité, et cela permet de faire facilement fonctionner les systèmes au-dessus d'un SRI existant en analysant uniquement (et en ligne) les extraits restitués par celui-ci. Par ailleurs, Zamir *et al.* (1998) n'ont pas observé d'augmentation de la qualité des clusters en utilisant les documents complets plutôt que les simples extraits. On peut cependant penser que ce résultat est dû a un comportement particulier de l'algorithme *STC*. En effet, Mecca *et al.* (2007) montrent qu'avec une méthode basée sur la *LSI*, les résultats sont significativement meilleurs quand les documents complets sont utilisés. Ayant à notre disposition des documents complets, nous utilisons ceux-ci plutôt que leurs extraits (la modélisation des documents est décrite en début de section 5).

Alors que les approches précédentes reposent principalement sur des modélisations vectorielles des documents, quelques auteurs ont traité le problème de clustering des résultats d'un SRI comme un problème de partitionnement de graphe (Boley *et al.*, 1999; Dhillon, 2001). Le partitionnement d'un graphe consiste à trouver une partition de l'ensemble des sommets qui minimalise la taille de la coupe induite (c'est-à-dire le nombre total d'arêtes entre deux groupes). Notre approche adopte une modélisation similaire à base de graphe, en revanche nous ne cherchons pas à minimiser la taille de la coupe induite ; Newman (2006) a en effet montré que cela ne conduit pas forcément à un découpage sémantiquement intéressant du graphe.

Le travail récent de Chen *et al.* (2008) exploite une technique de détection de communautés (par optimisation de la modularité) sur un graphe de termes construit à partir des co-occurrences de ceux-ci dans les extraits de documents restitués. Le but est de trouver les *word sense communities* qui permettent ensuite de construire des clusters de documents. Comme le signalent les auteurs, la principale originalité de cette approche par détection de communautés est que le nombre de clusters n'est pas un paramètre fixé mais dépend des données. Nous conservons cet avantage, même si notre algorithme est *in fine* très différent, puisque nous nous basons sur un graphe biparti et que nous cherchons des communautés comportant à la fois des termes et des documents. C'est en effet la dernière caractéristique particulière de notre approche par rapport à celles de l'état de l'art. Les méthodes citées soit construisent des clusters de documents puis les étiquettent, soit construisent des clusters d'étiquettes puis leur attribuent des documents. Seules (Dhillon, 2001) et l'approche (non encore citée) par une analyse formelle de concept (Carpineto *et al.*, 2004) ne séparent pas le clustering de l'étiquetage (ou le clustering de « l'assignation »).

3. Notons que Pons *et al.* (2006) montrent le lien entre l'approche que nous utilisons et les méthodes « spectrales » telles la *LSI* : grossièrement, les marches aléatoires courtes (section 4) renforcent les fortes valeurs propres de la matrice Termes $\leftrightarrow$ Documents, alors que la *LSI* ne garde qu'un certain nombre des plus fortes de ces valeurs propres.

4. Détection de communautés sur un graphe uniparti

Notre approche de détection de communautés est une adaptation aux cas des graphes bipartis de la méthode *Walktrap* proposée par Pons *et al.* (2006), que nous présentons dans cette section. Elle consiste à ramener le problème de détection de communautés dans un graphe à un problème de clustering hiérarchique d'un ensemble de points dans un espace vectoriel. Cette transformation (nous parlons de *géométrisation*) de la topologie du graphe dans un espace vectoriel pertinent est opérée par la méthode stochastique *Prox* initialement introduite par Gaume (2004).

4.1. *Notations préliminaires*

Un graphe $G = (V, E)$ est la donnée d'un ensemble non vide fini V de sommets, et d'un ensemble $E \subseteq V \times V$ de couples de sommets formant des arêtes :

– $n = |V|$ est l'*ordre* de G (son nombre de sommets),

– $m = |E|$ est la *taille* de G (son nombre d'arêtes),

– le graphe est *biparti* lorsqu'il existe deux ensembles $\top \subset V$ et $\bot \subset V$ tels que :

- $\top \cup \bot = V$ et $\top \cap \bot = \emptyset$: V est l'union des ensembles d'intersection vide ;

- $E \subseteq (\top \times \bot) \cup (\bot \times \top)$: il n'existe pas d'arête entre les sommets de $\bot$ ni entre les sommets de $\top$.

On notera alors un tel graphe biparti : $G = (\top, \bot, E)$. Par ailleurs, un graphe $G = (V, E)$ est dit *pondéré* lorsque chaque arête $(r, s) \in E$ est valuée par un poids $w(r, s) \in \mathbb{R}$. On notera alors un tel graphe pondéré $G = (V, E, w)$.

4.2. *Géométrisation de la topologie d'un graphe uniparti*

Soit $G = (V, E, w)$ un graphe pondéré de n sommets et m arêtes où chaque arête $(i, j) \in E$ est pondérée par un poids $w(i, j)$.

Pour géométriser le graphe G, on attribue à chaque sommet du graphe un vecteur de coordonnées dans $\mathbb{R}^n$ qui représente la « vision » qu'a le sommet en question sur le reste du graphe : plus deux sommets ont deux « visions » semblables, plus ces deux sommets seront proches dans $\mathbb{R}^n$. Cette approche permet d'appliquer alors au graphe ainsi plongé dans $\mathbb{R}^n$ tous les outils de la géométrie. On peut ainsi visualiser[4] le graphe (Gaume, 2008) ou bien appliquer des techniques de clustering hiérarchique comme présentées en section 4.3 pour la détection de communautés.

Pour modéliser la « vision » qu'a un sommet sur le reste du graphe, nous considérons un marcheur se baladant aléatoirement en suivant les arêtes du graphe. La distribution de probabilité de la position de ce marcheur est donnée par la chaîne de Markov associée

4. voir par exemple `http://prox.irit.fr`.

au graphe. Cette chaîne de Markov est définie par la matrice de transition P (équation 1) où $W(i)$ est la somme des poids des arêtes partant de i, soit $W(i) = \sum_{j \in V} w(i,j)$.

$$[P]_{ij} = \begin{cases} \frac{w(i,j)}{W(i)} & \text{si } (i,j) \in E \\ 0 & \text{sinon} \end{cases} \quad [1]$$

Si P_0 est la distribution de probabilité initiale du marcheur (c'est-à-dire sa probabilité de présence sur chacun des sommets de G au temps $t = 0$) alors la distribution de probabilités du marcheur est $P_0 P^t$ après t pas.

Pour modéliser la « vision » qu'a un sommet $u \in V$ à un instant t donné sur le reste du graphe, on définit le vecteur $\vartheta(u,t) = \delta_u P^t$ comme la distribution de probabilité d'un marcheur ayant effectué t pas depuis u, où δ_u est la certitude d'être sur le sommet u (δ_u est un vecteur-ligne de dimension $|V|$ contenant la valeur 0 sur toutes ses coordonnées, exceptée celle correspondant au sommet u qui vaut 1).

Si le graphe est apériodique, ce vecteur $\vartheta(u,t)$ converge quand $t \to \infty$. Cette limite correspond en fait à la version la plus simple du PageRank (Brin *et al.*, 1998; Manning *et al.*, 2008). Notons que cette limite ne dépend plus du sommet de départ, c'est-à-dire $\forall u, r \in V, \lim_{t\to\infty} \vartheta(u,t) = \lim_{t\to\infty} \vartheta(r,t)$ et donne une information globale sur le graphe (quels sont les sommets les plus « importants »).

À l'inverse, pour $t = 1$, $\vartheta(u,1)$ correspond à une version normalisée du vecteur d'adjacence du sommet u. Cette information est alors complètement locale, puisque ce vecteur ne dépend que du strict voisinage de u (u ne voit que ses voisins). Il est possible d'utiliser ce vecteur comme modèle, on a alors une modélisation vectorielle classique. Cependant cette modélisation ne prend en compte qu'une vision extrêmement locale de la topologie du graphe depuis u.

En revanche, lorsqu'on effectue des balades de temps courts ($3 \leqslant t \leqslant 8$), $\vartheta(u,t)$ dépend d'un voisinage plus large. Dans ce cas, même si deux sommets n'ont aucun voisin immédiat en commun, la ressemblance potentielle des voisins de leurs voisins peut amener à rapprocher ces deux sommets. Une « vision » un peu plus large de u se trouve « injectée » dans $\vartheta(u,t)$. Ces balades en temps courts sont un compromis entre une information trop locale ($t = 1$) et une information trop globale ($t \to \infty$).

4.3. *Clustering hiérarchique et coupe du dendrogramme d'un graphe uniparti*

La seconde phase de Kodex consiste en un clustering hiérarchique des sommets à partir de leurs vecteurs de coordonnés $\vartheta(u,t)$ dans $\mathbb{R}^n$ précédemment calculés pour un t donné. N'importe quelle méthode de clustering hiérarchique agglomératif pourrait être utilisée. Tout comme Pons et Latapy, nous utiliserons une adaptation de la méthode de Ward qui est intéressante tant du point de vue de la complexité que du point de vue de la qualité. Nous renvoyons à l'article (Pons *et al.*, 2006) pour une description complète de la méthode. Le principe est de partir d'un découpage pour lequel chaque

sommet est seul dans sa propre communauté puis de fusionner à chaque étape les deux communautés les plus proches en utilisant une distance sur les vecteurs calculés précédemment.

Cette phase de clustering hiérarchique produit un dendrogramme, qui représente $n-1$ partitionnements possibles des nœuds (n étant le nombre de sommets du graphe). La dernière étape de la méthode consiste donc à choisir une coupe particulière du dendrogramme. Pour cela on sélectionne celle qui maximise une fonction de qualité du partitionnement : la modularité. La modularité a justement été introduite par Newman *et al.* (2004) pour fournir un argument mathématique permettant de choisir une coupe privilégiée dans un dendrogramme sur les sommets d'un graphe. Lorsque le graphe est uniparti et non dirigé, la modularité $Q(Z)$ d'une partition Z des sommets du graphe est définie par :

$$Q(Z) = \frac{1}{S} \cdot \sum_{C \in Z} \sum_{i,j \in C} \left(A_{i,j} - \frac{k_i \cdot k_j}{S} \right) \quad [2]$$

où A est la matrice d'adjacence du graphe, $S = \sum_{i,j \in V} A_{i,j}$ et k_i est le degré du sommet i. La modularité $Q(Z)$ de la partition Z est la différence entre le nombre d'arêtes appartenant à une même communauté dans le graphe réel G et ce même nombre attendu dans un graphe aléatoire de même ordre et de même taille dans lequel les degrés de tous les sommets restent inchangés par rapport à G.

Pons et Latapy montrent que la complexité de cette méthode de clustering des sommets d'un graphe est dans tous les cas inférieure à $O(m \cdot n^2)$ et que si le dendrogramme est équilibré (ce qui semble être une hypothèse réaliste), elle est inférieure à $O(m \cdot n \cdot \log(n))$.

5. Détection de communautés dans un graphe biparti Documents ↔ Termes

Afin d'organiser les résultats d'un SRI pour une requête R donnée, nous commençons par collecter les n_D premiers documents restitués par le SRI (se sont les « documents à clusteriser ») ainsi que l'ensemble des termes indexés par ce SRI pour ces n_D documents. Nous filtrons les termes ayant un score BM25 (Robertson *et al.*, 1994) sur les documents à clusteriser[5] inférieur à un certain seuil (f_{BM25}). Ce premier filtre permet de supprimer les termes peu « discriminants » pour le clustering, typiquement les termes apparaissant dans « trop » de documents. De plus, afin d'éliminer les termes trop rares, nous filtrons ceux appartenant à moins d'un certain nombre (f_{df}) de documents dans la collection complète (notion de *document frequency*). Notons que les mots trop rares sont inconnus de la plupart des utilisateurs et engendrent une surcharge cognitive parasite car ils ne sont donc d'aucune utilité dans les étiquettes (*labels*) W_i

5. Pour chaque terme, le « score BM25 sur les documents à clusteriser » est le score maximal obtenu avec l'un des documents à clusteriser.

(représentant la thématique des clusters). Ensuite est calculé, pour chaque terme, le ratio entre la *document frequency* du terme dans l'ensemble des documents restitués et cette même *document frequency* calculée dans la collection complète. Seul les n_T termes ayant les plus fort ratios sont gardés. Mis à part le rôle « d'anti-parasitage cognitif » de f_{df}, le réglage des trois seuils f_{BM25}, f_{df} et n_T doit se faire dans le but de diminuer la complexité pour la suite des calculs, sans dégrader les résultats.

Nous pondérons chaque arête entre un terme et un document par une mesure de $tf \cdot idf$ calculée sur ce sous-ensemble de n_D documents. Pour une requête R donnée, nous obtenons ainsi $G_R = (D, T, w)$ un graphe biparti pondéré Documents $\leftrightarrow$ Termes de $n = n_D + n_T$ sommets et m arêtes. Notons que ce graphe biparti correspond à un sous-ensemble de la matrice $Documents \times Termes$ complète.

5.1. *Géométrisation de la topologie d'un graphe biparti*

La méthode de géométrisation décrite en section 4.2 fonctionne bien pour les graphes unipartis (Gaume, 2004). Toutefois, nous avons affaire ici à un graphe $G_R = (D, T, w)$ biparti et la convergence de $\vartheta(u, t)$ n'est plus assurée – en effet un graphe biparti est un graphe périodique de période 2. Plus intuitivement, en fonction du type du sommet de départ (document ou terme) et de la parité de t, les coordonnées non nulles de $\vartheta(u, t)$ seront soit complètement sur l'espace engendré par les documents, soit complètement sur l'espace engendré par les termes (cf. Figure 1). Cela ne permet pas de comparer dans un même espace des documents et des termes.

Pour répondre à ce problème nous proposons une adaptation simple : elle consiste à effectuer des balades de longueur $2 \cdot t$ pour les documents, et des balades de longueur $2 \cdot t + 1$ pour les termes. On définit alors le vecteur $\vartheta_b(u, t)$ tel que :

$$\vartheta_b(u,t) = \begin{cases} \vartheta(u, 2t) & \text{si } u \in Documents \\ \vartheta(u, 2t+1) & \text{si } u \in Termes. \end{cases} \quad [3]$$

Il est alors clair que pour tout $t \geqslant 1$:

– si u est un document, alors les coordonnées non nulles de $\vartheta_b(u, t) = \vartheta(u, 2t)$ sont toutes sur l'espace engendré par les documents,

– si u est un terme, alors les coordonnées non nulles de $\vartheta_b(u, t) = \vartheta(u, 2t + 1)$ sont toutes sur l'espace engendré par les documents.

En fait, l'ensemble des n sommets (les n_D documents et les n_T termes) de G_R sont alors plongés dans le même espace $\mathbb{R}^{n_D}$ engendré par les n_D premiers documents restitués par le SRI.

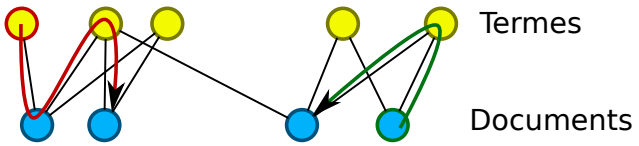

Figure 1. *Exemple de deux marches aléatoires sur le graphe biparti Documents ↔ Termes. En rouge (à gauche) une marche de temps impaire (trois pas) partant d'un terme ; en vert (à droite) une marche de temps paire (deux pas) partant d'un document. On observe qu'en fonction du type (document ou terme) du sommet de départ et de la parité de la longueur de la balade, un marcheur aléatoire se retrouve obligatoirement sur un document, ou obligatoirement sur un terme.*

5.2. *Clustering hiérarchique et coupe du dendrogramme d'un graphe biparti*

Comme présentée en section 4.3, la seconde phase de notre proposition consiste en un clustering hiérarchique des sommets (documents et termes) à partir de leurs vecteurs de coordonnés $\vartheta_b(u,t)$ dans $\mathbb{R}^{n_D}$ précédemment calculés pour un t donné.

Nous commençons par utiliser ici la même adaptation de la méthode de Ward : à partir d'un découpage où chaque sommet (document et terme) est seul dans sa propre communauté, on fusionne à chaque étape les deux communautés les plus proches en utilisant une distance sur les vecteurs $\vartheta_b(u,t)$ calculés précédemment.

Cette phase de clustering hiérarchique construit un dendrogramme, qui représente $n-1$ partitionnements possibles des n sommets. Pour choisir une coupe particulière de ce dendrogramme, nous ne pouvons plus ici utiliser – comme en section 4.3 – la modularité introduite par Newman *et al.* (2004) qui n'est définie que pour les graphes unipartis. Aussi, nous utilisons une extension de cette mesure pour les graphes bipartis proposée par Barber (2007). Nous adaptons à notre tour cette mesure au cas des graphes bipartis pondérés :

$$Q_b(Z) = \frac{1}{S} \cdot \sum_{C \in Z} \sum_{i \in D \cap C} \sum_{j \in T \cap C} \left(w(i,j) - \frac{W(i) \cdot W(j)}{S} \right) \qquad [4]$$

où $S = \sum_{i,j \in V} w(i,j)$ est la somme des poids de tous les arêtes du graphe, et $W(i) = \sum_{j \in V} w(i,j)$ est la somme des poids des arêtes partant de i.

Nous avons indiqué en section 4.3 que la complexité de cette méthode de clustering des sommets est dans tous les cas inférieure à $O(m \cdot n^2)$ et que si le dendrogramme est équilibré – ce qui semble être une hypothèse réaliste – elle est inférieure à $O(m \cdot n \cdot \log(n))$. Dans ces notations, m correspond au nombre d'arêtes du graphe et n correspond au nombre de sommets du graphe. Or, dans ce calcul de complexité, la composante en $O(n)$ correspond au calcul d'une distance entre deux vecteurs de taille n dans le cas des graphes unipartis. Toutefois, dans notre cas $n = n_D + n_T$, et les

vecteurs ont des coordonnées nulles sur l'ensemble des n_T termes et donc ce calcul de distance se fait en $O(n_D)$. En supposant que le dendrogramme est équilibré, notre méthode a donc une complexité en $O(m \cdot n_D \cdot \log(n_D + n_T))$. De plus, en général dans un graphe réel biparti Documents $\leftrightarrow$ Termes, m est $O(n_D + n_T)$, la complexité de la méthode est donc en $O((n_D^2 + n_D \cdot n_T) \cdot \log(n_D + n_T))$.

6. Évaluation et résultats

Notre évaluation a pour but de valider l'hypothèse suivante : à partir d'un ensemble de documents restitués par un moteur de recherche pour une requête, Kodex est capable de construire un cluster contenant un maximum de documents pertinents. Pour ce faire, nous exploitons une collection de test présentée en section 6.1 et réalisons des analyses détaillées en section 6.2.

6.1. *Collection de test et système Kodex*

Nous avons constitué, dans le cadre du projet Quaero, une collection de test comprenant les trois éléments suivants :

– un *corpus* composé de 2,6 millions de pages Web issues du domaine français (.fr) et aspirées par le moteur de recherche Exalead en 2008,

– 25 *topics* comprenant une requête textuelle et un texte explicitant le besoin en information. Ces derniers sont des besoins réels qui ont été soumis par des utilisateurs d'Exalead (extraits du log de ce moteur de recherche),

– des *jugements de pertinence* recueillis suivant une procédure similaire à celle utilisée dans TREC (Harman, 2005). Nous avons constitué un *pool* de résultats issus de 144 configurations différentes du moteur de recherche Terrier (Ounis *et al.*, 2005). Ces configurations ont été construites en utilisant différentes formes d'indexation, différents modèles de recherche, et en réalisant ou non l'expansion de requêtes. Ce *pool* a ensuite été évalué manuellement pour identifier les documents pertinents de chaque requête.

Pour les expérimentations de cette section, nous avons utilisé la configuration suivante de Kodex :

– le SRI utilisé est Terrier (indexation avec élimination des mots vides, modèle Okapi BM25 avec paramètres par défaut, reflétant la configuration optimale du système dans la majorité des situations),

– nombre de documents à clusteriser (pour chaque requête) : $n_D = 100$,

– marches de longueur 4 pour les documents et 5 pour les termes ($t = 2$ dans l'équation 3),

– seuil $f_{BM25} = 5$ sur le score BM25 maximum des termes par rapport aux documents à clusteriser,

– seuil $f_{df} = 100$ sur la *document frequency* des termes,

– nombre de termes à clusteriser : $n_T = 1000$.

6.2. *Expérimentations : protocole et résultats*

Afin d'évaluer l'intérêt du clustering de Kodex par rapport à une organisation sous forme de liste de documents, nous nous basons sur la procédure MK1-k décrite dans Tombros *et al.* (2002). Le principe est basé sur les deux ensembles de documents e_k et e_t provenant respectivement de Kodex et Terrier. L'ensemble e_k contient les documents du « meilleur cluster » de Kodex (contenant k documents) et e_t contient les k premiers documents de la liste de Terrier. Les ensembles e_k et e_t sont alors comparables grâce aux mesures rappel (R), précision (P) et F_1 définies comme suit, pour un système s et un topic t (Jardine *et al.*, 1971).

$$R(s,t) = \frac{\text{Nombre de documents pertinents dans } e_s \text{ pour } t}{\text{Nombre de documents pertinents dans le top-100 de Terrier pour } t} \quad [5]$$

$$P(s,t) = \frac{\text{Nombre de documents pertinents dans } e_s \text{ pour } t}{\text{Nombre de documents dans } e_s \text{ pour } t} \quad [6]$$

$$F_1(s,t) = \frac{2 \cdot R(s,t) \cdot P(s,t)}{R(s,t) + P(s,t)} \quad [7]$$

Pour identifier le « meilleur » cluster de Kodex (e_k) une possibilité serait de montrer les clusters résultats à des utilisateurs pour qu'ils le sélectionnent ; on retiendrait alors le cluster le plus consensuel. Pour ne pas introduire un facteur (humain) externe à l'évaluation et afin de garantir sa reproductibilité, nous avons plutôt opté pour un critère objectif de sélection de ce meilleur cluster. La mesure de précision nous est apparue la plus adaptée car elle indique la plus forte concentration de documents pertinents dans un cluster. Le tableau 1 présente les résultats comparatifs pour Kodex et Terrier, calculés sur les mesures R, P et F_1. Les résultats montrent clairement l'amélioration matérielle ($\geqslant 22\,\%$) des trois mesures apportée par Kodex sur les top-k documents de la liste Terrier.

Run	R	P	F_1
Kodex	**0,4461**	**0,4728**	**0,3210**
Terrier	0,3295	0,3709	0,2628
Gain de Kodex *versus* Terrier	35 %*	27 %	22 %*

Tableau 1. *Moyenne de R, P et F_1 pour Terrier et Kodex, calculée sur 25 topics. Une astérisque indique que l'amélioration est statistiquement significative selon le test t de Student pairé et bilatéral avec $p < 0,05$*

Afin de compléter notre évaluation, nous avons ensuite comparé Kodex à une approche d'état de l'art en clustering : l'algorithme *Lingo* implémenté dans Carrot[2] (Osinski *et al.*, 2005) avec les paramètres par défaut, traduisant une configuration optimale selon les concepteurs du système. Afin que les deux approches soient

directement comparables, l'entrée du système Carrot^2 est la même que celle du système Kodex, à savoir le top-100 des documents restitués par Terrier (modèle Okapi BM25). Dans le tableau 2, nous donnons des éléments de comparaison quant au nombre de clusters générés et à leur taille. Comparativement à Carrot^2, Kodex produit de plus gros clusters. Contrairement à Kodex, Carrot^2 produit des clusters avec recouvrement (un document peut faire partie de plusieurs clusters), ce qui concourt à expliquer le grand nombre de clusters produits.

Système	Nombre moyen de clusters par requête	Taille moyenne des clusters	Taille moyenne du meilleur cluster
Kodex	4,6	21,2	20,6
Carrot^2	28,9	7,3	5,4

Tableau 2. *Statistiques relatives aux clusters produits par Kodex et Carrot^2*

Puisque Carrot^2 repose sur du clustering avec recouvrement, il nous est impossible de comparer les deux systèmes avec les mesures usuelles d'entropie, d'information mutuelle et de pureté (Crabtree *et al.*, 2007). Le tableau 3 rapporte les performances comparatives de Kodex et Carrot^2 pour les mesures R, P et F_1.

Run	R	P	F_1
Kodex	**0,4461**	0,4728	**0,3210**
Carrot^2	0.2161	**0,6978**	0,2651
Gain de Kodex *versus* Carrot^2	106 %*	-32 %*	21 %

Tableau 3. *Moyenne de R, P et F_1 pour les meilleurs clusters selon P de Terrier et Kodex, calculée sur 25 topics. Une astérisque indique que l'amélioration est statistiquement significative selon le test t de Student pairé et bilatéral avec $p < 0,05$*

Nous constatons que Kodex améliore le rappel et la mesure F_1. La baisse observée sur la précision peut s'expliquer par le fait qu'obtenir une bonne précision sur des petits clusters (cas de Carrot^2) est plus facile que sur des gros (cas de Kodex). Toutefois, la présentation de petits clusters peut frustrer l'utilisateur et pénalise Carrot^2 sur le rappel. Globalement, la mesure F_1, qui synthétise rappel et précision de façon équilibrée, est en faveur de Kodex.

Pour résumer les observations de cette évaluation, considérant la mesure F_1, Kodex est significativement meilleur que le système d'état de l'art « liste » Terrier BM25. Kodex est également meilleur que le système d'état de l'art de clustering Carrot^2. Afin de s'assurer de la robustesse de notre approche, ces conclusions doivent être confirmées par une évaluation complémentaire sur d'autres collections (TREC notamment).

7. Conclusion et perspectives

Nous avons présenté dans cet article une méthode de présentation des résultats d'un moteur de recherche par détection de communautés sur le graphe biparti Documents $\leftrightarrow$ Termes. La détection des communautés est basée sur la géométrisation du graphe par une approche Markovienne, puis sur un clustering hiérarchique agglomératif. Nous obtenons ainsi des co-clusters contenant à la fois des documents et des termes.

Il est essentiel que les clusters de documents soient le plus pertinents possibles, c'est à dire qu'il existe un cluster regroupant les documents pertinents pour l'utilisateur. C'est ce que nous avons évalué en section 6 en montrant que Kodex améliore de manière significative les résultats proposés par un système de l'état de l'art. Il serait envisageable pour améliorer à nouveau ces résultats de se baser sur un bi-graphe Documents $\leftrightarrow$ Termes plus complexe, où les termes ne sont plus simplement les mots présents dans chaque document mais le résultat d'un étiquetage plus précis des documents. La prise en compte des hyperliens entre les pages pourrait aussi aider à définir les clusters. De plus, la méthode de clustering de Kodex procède par partitionnement : chaque terme (et chaque document) n'appartient qu'à un cluster, constituant une limite certaine. Un récent parallèle entre le clustering de graphes bi-partis et l'Analyse Formelle des Concepts (AFC) (Gaume *et al.*, 2010) donne des pistes pour remplacer l'algorithme de clustering hiérarchique par une méthode (basée sur une relaxation des approches AFC) permettant un recouvrement entre clusters.

Dans cet article nous n'avons pas abordé le problème de l'identification du meilleur cluster par l'utilisateur. Nous avons, pour le moment, supposé que l'utilisateur identifiait toujours le meilleur cluster selon son besoin (ce cluster étant celui de meilleure précision). Nous devrons donc évaluer par la suite la capacité d'un utilisateur à identifier le meilleur cluster afin de focaliser son attention sur les documents qui y sont circonscrits. Ce sont les clusters de termes (les étiquettes) qui informeront l'utilisateur quant à la thématique des clusters. Grâce à cette information, l'utilisateur pourra reclasser les documents en plaçant en tête de liste les documents du cluster identifié, ou préciser sa requête à partir des étiquettes du cluster.

8. Bibliographie

Barber M. J., "Modularity and community detection in bipartite networks", *Phys. Rev. E*, vol. 76, n° 6, p. 066102, December, 2007.

Boley D., Gini M., Gross R., Han E. S., Hastings K., Karypis G., Kumar V., Mobasher B., Moore J., "Partitioning-based clustering for Web document categorization", *Decision Support Systems*, vol. 27, n° 3, p. 329–341, December, 1999.

Brin S., Page L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine", *Computer Networks*, vol. 30, n° 1-7, p. 107–117, 1998.

Carpineto C., Osiński S., Romano G., Weiss D., "A survey of Web clustering engines", *ACM Comput. Surv.*, vol. 41, n° 3, p. 1–38, 2009.

Carpineto C., Romano G., “Exploiting the potential of concept lattices for information retrieval with CREDO”, *Journal of Universal Computer Science*, vol. 10, n° 8, p. 985–1013, August, 2004.

Chen J., Zaïane O. R., Goebel R., “An Unsupervised Approach to Cluster Web Search Results Based on Word Sense Communities”, *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, p. 725–729, 2008.

Crabtree D., Andreae P., Gao X., “QC4 - A Clustering Evaluation Method”, *Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and data mining*, Springer-Verlag, Nanjing, China, p. 59–70, 2007.

Cutting D. R., Karger D. R., Pedersen J. O., Tukey J. W., “Scatter/Gather: a cluster-based approach to browsing large document collections”, *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, Copenhagen, Denmark, p. 318–329, 1992.

Deerwester S. C., Dumais S. T., Landauer T. K., Furnas G. W., Harshman R. A., “Indexing by Latent Semantic Analysis”, *Journal of the American Society for Information Science*, vol. 41, n° 6, p. 391–407, 1990.

Dhillon I. S., “Co-clustering documents and words using bipartite spectral graph partitioning”, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, San Francisco, California, p. 269–274, 2001.

Ferragina P., Gulli A., “A personalized search engine based on web-snippet hierarchical clustering”, *Special interest tracks and posters of the 14th international conference on World Wide Web*, ACM, Chiba, Japan, p. 801–810, 2005.

Fortunato S., “Community detection in graphs”, *Physics Reports*, vol. 486, n° 3-5, p. 75–174, 2010.

Furnas G. W., Landauer T. K., Gomez L. M., Dumais S. T., “The Vocabulary Problem in Human-System Communication”, *Commun. ACM*, vol. 30, n° 11, p. 964–971, 1987.

Gaume B., “Balades aléatoires dans les petits mondes lexicaux”, *I3 Information Interaction Intelligence*, vol. 4, n° 2, p. 31–90, 2004.

Gaume B., “Mapping the forms of meaning in small worlds”, *International Journal of Intelligent Systems*, vol. 23, n° 7, p. 848–862, 2008.

Gaume B., Navarro E., Prade H., “A Parallel between Extended Formal Concept Analysis and Bipartite Graphs Analysis”, *in* E. Hüllermeier, R. Kruse, F. Hoffmann (eds), *IPMU’10: Proceedings of the 13th International Conference on Information Processing and Management of Uncertainty*, vol. 6178 of *LNCS*, Springer, p. 270–280, 2010.

Harman D. K., “The TREC Test Collections”, *in* E. M. Voorhees, D. K. Harman (eds), *TREC: Experiment and Evaluation in Information Retrieval*, MIT Press, Cambridge, MA, USA, chapter 2, p. 21–53, 2005.

Hearst M. A., Pedersen J. O., “Reexamining the cluster hypothesis: scatter/gather on retrieval results”, *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, Zurich, Switzerland, p. 76–84, 1996.

Jardine N., van Rijsbergen C. J., “The Use of Hierarchic Clustering in Information Retrieval”, *Inform. Stor. Retr.*, vol. 7, n° 5, p. 217–240, 1971.

Manning C. D., Raghavan P., Schütze H., *Introduction to Information Retrieval*, Cambridge University Press, July, 2008.

Mecca G., Raunich S., Pappalardo A., "A new algorithm for clustering search results", *Data & Knowledge Engineering*, vol. 62, n° 3, p. 504–522, September, 2007.

Newman M. E. J., "The structure and function of complex networks", *SIAM Review*, vol. 45, p. 167–256, March, 2003.

Newman M. E. J., "Finding community structure in networks using the eigenvectors of matrices", *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 74, n° 3, p. 036104–19, 2006.

Newman M. E. J., Girvan M., "Finding and evaluating community structure in networks", *Phys. Rev. E*, vol. 69, n° 2, p. 026113, February, 2004.

Osinski S., Weiss D., "A Concept-Driven Algorithm for Clustering Search Results", *IEEE Intelligent Systems*, vol. 20, n° 3, p. 48–54, 2005.

Ounis I., Amati G., Plachouras V., He B., Macdonald C., Johnson D., "Terrier Information Retrieval Platform", *ECIR'05: Proceedings of the 27th European Conference on IR Research*, vol. 3408 of *LNCS*, Springer, p. 517–519, 2005.

Pons P., Latapy M., "Computing communities in large networks using random walks (long version)", *Journal of Graph Algorithms and Applications (JGAA)*, vol. 10, n° 2, p. 191–218, 2006.

Porter M. A., Onnela J., Mucha P. J., "Communities in Networks", *Notices of the American Mathematical Society*, vol. 56, n° 9, p. 1082–1097, 2009.

Robertson S. E., Walker S., Jones S., Hancock-Beaulieu M., Gatford M., "Okapi at TREC-3", *TREC-3: Proceedings of the 3rd Text REtrieval Conference*, p. 109–126, 1994.

Salton G., Wong A., Yang C. S., "A Vector Space Model for Automatic Indexing", *Commun. ACM*, vol. 18, n° 11, p. 613–620, November, 1975.

Tombros A., Villa R., Rijsbergen C. V., "The effectiveness of query-specific hierarchic clustering in information retrieval", *Information Processing & Management*, vol. 38, n° 4, p. 559–582, 2002.

Watts D., Strogatz S., "Collective dynamics of 'small-world' networks", *Nature*, vol. 393, n° 6684, p. 440–442, 1998.

Zamir O., Etzioni O., "Web document clustering: a feasibility demonstration", *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, Melbourne, Australia, p. 46–54, 1998.

Zamir O., Etzioni O., "Grouper: a dynamic clustering interface to Web search results", *Proceedings of the eighth international conference on World Wide Web*, Toronto, Canada, p. 1361–1374, 1999.

Zhang D., Dong Y., "Semantic, Hierarchical, Online Clustering of Web Search Results", *in* J. X. Yu, X. Lin, H. Lu, Y. Zhang (eds), *APWeb'04: Proceedings of the 6th Asia-Pacific Web Conference on Advanced Web Technologies and Applications*, vol. 3007 of *LNCS*, Springer, p. 69–78, 2004.