Modèle d'indexation dynamique à base d'ontologies

Gilles Hubert[1], Josiane Mothe[1,2], Bachelin Ralalason[1], Bertin Ramamonjisoa[3]

[1] *IRIT, 118 route de Narbonne, 31062 Toulouse Cedex 9,*
[2] *Institut Universitaire de Formation des Maitres, Av. de l'URSS, 31078 Toulouse*
[3] *Université de Fianarantsoa, Ecole Nationale d'informatique, BP 1487 Tanambao – Fianarantsoa 301, Madagascar*

{hubert/mothe/bachelin}@irit.fr

bertin@mail.univ-fianar.mg

RÉSUMÉ. *Cet article propose un modèle de données pour une indexation basée sur une ontologie de référence représentant la sémantique des termes d'indexation. Le modèle proposé vise à permettre une indexation en temps réel qui suit la dynamique du corpus tout en assurant la disponibilité des documents et de l'index. Ceci permet de garder la cohérence entre les documents de la collection, l'index et l'ontologie de référence. Notre modèle permet ainsi d'éviter la reconstruction de l'index lors de la modification du corpus de documents car il reste à jour en permanence. Ainsi, le modèle que nous proposons permet l'indexation sémantique dynamique d'un corpus. Le modèle est illustré par des algorithmes expliquant sa mise en œuvre.*

ABSTRACT. *This paper proposes a data model for semantic indexing based on an ontology. The ontology represents the semantic of the index terms. The proposed model aims at enabling real-time indexing according to the dynamic of the corpus while insuring the availability of documents and the index. So, it permits to keep the coherence between the documents, the index and the reference ontology. Using our model, there is no need to rebuild the index from scratch because the index is permanently up to date. Thus, the model we propose makes it possible a dynamic and semantic indexing of documents collection. The model is illustrated by some algorithms showing its implementation.*

MOTS-CLÉS : *Recherche d'information, Indexation sémantique, Indexation à base d'ontologie, Dynamique des corpus, Structures de donnés.*

KEYWORDS: *Information retrieval, Semantic indexing, Ontology-based indexing, Dynamics in corpus, Indexing data structure.*

1. Introduction

L'objectif des systèmes de recherche d'information (SRI) est de fournir aux utilisateurs les documents pertinents par rapport aux besoins qu'ils expriment.

Les SRI utilisent des listes inversées qui rassemblent les différents termes d'indexation choisis pour représenter les contenus des documents et les liens vers ces documents. En complément, à chaque couple (terme d'indexation, document) est associé un poids qui représente l'importance du terme dans un document. Lorsqu'une requête est soumise au système, les termes qu'elle contient sont mis en correspondance avec les termes d'indexation extraits des documents pour en déduire les documents à restituer à l'utilisateur. La phase d'indexation est donc une phase primordiale dans le processus de recherche. Lorsque la collection de documents est figée, l'indexation est réalisée une fois pour toutes. Cependant, ce cas n'intervient que dans le cadre des campagnes d'évaluation des moteurs où il s'agit de confronter différents SRI sur des mêmes collections. Dans l'usage réel, le SRI doit être capable de faire face à des collections dynamiques dans lesquelles des documents sont modifiés, ajoutés et supprimés.

Dans la littérature, diverses méthodes et stratégies ont été proposées pour permettre la mise à jour des index lorsque la collection de documents est modifiée. Il s'agit par exemple de l'utilisation des délimiteurs [Salton et al, 1993] et [Baeza-Yates et Navarro, 2000], la mise à jour incrémentale d'index [Lim et al., 2007], ainsi que la méthode *diff* [Ukkonen, 1985]. Ces méthodes considèrent une indexation de type « sac de mots », dans laquelle les termes issus des documents sont considérés comme indépendants. Cependant, il existe en réalité des relations (équivalence, subsomption, association, …) entre les termes. De nouvelles approches tentent de les prendre en compte de façon automatique lors de l'indexation, par exemple au travers de l'indexation sémantique [Hernandez et al. 2007]. L'indexation sémantique via des ontologies à laquelle nous nous intéressons dans nos travaux s'appuie sur les technologies du web sémantique. Dans ce type d'approche, la connaissance du domaine (terminologique en particulier) est représentée sous forme d'ontologies, c'est-à-dire en particulier de concepts, d'instances de ces concepts et de relations.

Comme dans le cas de l'approche sac de mots, la collection de documents à indexer peut être dynamique et donc subir des modifications ; il est donc important de proposer des principes pour la mise à jour des index dans le cas d'une indexation sémantique. De plus, contrairement à l'approche sac de mots, le vocabulaire utilisé lors de l'indexation peut être amené à varier indépendamment des documents. Ainsi, l'ontologie qui sert de référence à l'indexation peut être modifiée. Dans ce dernier cas, il est important de considérer la mise à jour de l'indexation consécutive à une modification du vocabulaire de référence, cela afin de maintenir une cohérence entre les documents et le vocabulaire d'indexation.

Cet article présente les structures de données nécessaires ainsi que les stratégies utilisées pour permettre l'actualisation en temps réel des listes inversées issues de l'indexation sémantique basée sur une ontologie. Dans un premier temps, dans la section 2, nous présentons les différents travaux de la littérature liés à cet axe de recherche. Puis, dans la section 3, nous présentons notre modèle de données représentant les structures d'index. En section 4 nous discutons nos solutions en matière de structure de données et de stratégies des mises à jour d'index. Enfin, nous terminons, dans la section 5, par des conclusions et perspectives à notre travail de recherche.

2. Etat de l'art

La mise à jour des documents de la collection, l'arrivée de nouveaux documents ou la suppression de documents d'un corpus indexé nécessitent l'actualisation de l'index afin de garder la cohérence entre les documents et les index. Cette mise à jour est primordiale pour que le SRI puisse répondre au mieux aux besoins d'un utilisateur.

Dans cette section, nous présentons les travaux reliés relatifs à l'indexation dynamique des documents ainsi qu'à l'indexation sémantique à partir d'ontologies dans la mesure où notre approche s'intègre dans ce type d'indexation.

2.1. Indexation dynamique

L'indexation dynamique consiste à mettre à jour l'index après modification de la collection (ajout, modification et suppression de documents).

La toile correspond à ce cadre de collections hautement dynamiques. Ainsi des travaux dans le domaine de la recherche et de la collecte incrémentale des pages web visent à permettre aux collections d'un moteur de recherche d'être plus synchronisées avec le web réel. Dans [Cho et Garcia-Molina, 2000] la collecte vise à télécharger toutes les pages web relatives à un URL de départ. Ensuite, la collecte incrémentale ne télécharge que les pages qui ont été modifiées à la source. Dans cette méthode, la maintenance de la collection ne nécessite pas de télécharger les pages qui n'ont subi aucune modification. Cependant, la collection synchronisée, qui est l'image exacte des documents sources, ne peut pas être recherchée immédiatement car la reconstruction (à partir de zéro) de l'index de mots-clés est moins fréquente que la mise à jour de la collection. Il n'est donc possible de rechercher les nouveaux documents collectés qu'après la prochaine reconstruction de l'index. Le décalage entre la mise à jour de la collection et celle de l'index s'explique par le fait que la reconstruction complète de l'index est très coûteuse en termes de temps de traitement.

Face à ce problème, [Lim et al., 2007] propose une méthode de mise à jour incrémentale d'index inversé pour les documents qui ont changé sur le web. Cette méthode qui s'appelle *délimiteur-diff* [Landmark-diff en anglais] combine la technique de *délimiteur* [Landmark en anglais] [Salton et al, 1993], [Baeza-Yates et Navarro, 2000] avec la méthode *diff* [Ukkonen, 1985]. La technique de délimiteur consiste à subdiviser les documents en plusieurs blocs et à mémoriser les positions relatives des mots du document par rapport aux délimiteurs du bloc dans lequel les mots se trouvent. La méthode *diff* par contre mémorise la liste des modifications apportées dans un document pour obtenir sa nouvelle version. A chaque document est donc associé un annuaire de délimiteurs qui liste les différents délimiteurs et leurs positions absolues dans le document. La mise à jour d'un document entraine celle de l'annuaire des délimiteurs associé au document. Ce nouvel annuaire des délimiteurs associés à la *transcription des modifications* [edit transcript en anglais] permet de mettre à jour l'index inversé. La transcription des modifications correspond à la liste des modifications qui amènent vers la nouvelle version d'un document à partir de l'ancienne version). Une entrée dans l'index inversé est composée de l'identifiant de chacun des mots (wordID), la liste des documents contenant un mot donné (docID), l'identifiant du délimiteur (landmarkID) auquel le mot est rattaché, ainsi que la position relative (offset) du mot par rapport au délimiteur. La méthode Délimiteur-diff présente une vitesse de mise à jour de l'index inversé (pour les documents qui ont changé) trois fois plus rapide que la méthode *Premier Index* [forward index en anglais] utilisée par Google [Page et Brin, 1998]. De son côté, [Büttcher S. et Clarke C., 2006] propose une approche hybride qui combine les deux méthodes *In-Place* (mise à jour directe sur place) et *Merge-based* (mise à jour d'index basée sur la technique de fusion) qui sont deux techniques largement utilisées pour la mise à jour d'index dans un SRI dynamique basés sur les listes inversées. La stratégie *In-Place* consiste non seulement à transformer les structures de données en d'autres structures plus petites mais aussi à écraser les anciennes versions de documents. La technique *Merge-based* [Cutting et Pedersen, 1990], quant-à elle, consiste à minimiser le déplacement de la tête de disque pour la maintenance de l'index. La mise à jour des index sur disque se fait dès que l'espace mémoire alloué aux index commence à être saturé. L'inconvénient de la technique *Merge-based* est que la totalité de l'index inversé est lu et écrit sur le disque à chaque mise à jour, même si une petite partie seulement de l'index est affecté. Les accès au disque sont donc importants. La stratégie *In-place* essaie de résoudre ce problème en laissant assez de place à la fin des index inversés. Dès lors, [Büttcher S. et Clarke C., 2006] utilisent la méthode *In-place* (respectivement *Merge-based*) pour une longue liste inversée (respectivement une courte liste inversée). Cette approche hybride combinant l'utilisation de la méthode *In-place* et *Merge-based* donne une meilleure performance en termes de temps d'indexation que l'une ou l'autre de ces méthodes, tout en gardant la même performance au niveau de traitement de la requête.

La plupart des algorithmes d'actualisation d'index ne permet pas l'ajout de nouveaux documents pendant le processus de mise à jour de l'index. De plus, ce processus peut demander plusieurs heures pour les corpus de grande taille. Ainsi, [Galambos L., 2006] a développé un algorithme de mise à jour dynamique d'index. Cet algorithme est basé sur la mise à jour incrémentale d'index en utilisant une liste inversée de type *Citerne (Tanker).* Dans ce modèle, une *citerne* est un index composé de *Barils* (*barrels*) ; où un baril est un index réalisé sur un sous-ensemble de documents du corpus. L'actualisation d'index est réalisée à chaque arrivée de nouveaux documents et à chaque modification de documents. La modification d'un document est considérée comme une opération de suppression suivie d'un ajout d'un nouveau document.

En ce qui concerne Google [Page et Brin, 1998] qui a été conçu pour rechercher sur le www, il emploie plusieurs techniques (importance des pages ou PageRank, structure des liens, texte des liens, polices de caractères, position des mots dans les documents, etc…) pour améliorer la qualité de recherche. L'analyse des structures des liens à partir de la popularité des pages permet à Google d'évaluer la qualité des pages web [Page et Brin, 1998]. Pour atteindre ses objectifs, les structures de données utilisées par Google sont :

Liste d'importance (*Hit list*) : il s'agit d'une liste des occurrences d'un mot d'un document particulier, comprenant les informations de la position, la police et la taille (visuelle) du mot.

Baril (*Barrel*) : Index obtenu par la méthode *Premier Index* (*Forward-Index*) présentée plus haut, partiellement trié par le docID. Chaque baril stocke des listes d'importance pour un ensemble d'identifiant de mots (*wordID)*. Il y a deux types de barils : Les barils courts qui contiennent les listes d'importance incluant le titre ou les ancres et les barils longs pour toutes les listes d'importance.

Entrepôt (*Repository*) : Contient le code HTML des pages Web. Chaque document est préfixé par son identifiant docID, sa longueur, et son URL.

Index Doc: Garde les informations concernant les documents telles qu'un pointeur vers le dépôt, le nombre de liens provenant d'autres pages, le nombre de liens sortant qui pointent vers d'autres pages, l'état de chaque document ainsi que son URL.

Lexique : Table de hashage gérée en mémoire vive qui garantit l'appariement entre un mot et son identifiant (wordID)

Ancre : Stocke les destinations et les étiquettes des liens.

Dans nos travaux, nous nous inspirons de ces différentes techniques qui visent à indexer les documents sur le web et les adaptons à des collections indexées par des ontologies. Ainsi, la méthode proposée par Google associée avec la méthode *Délimiteur-Diff* va nous permettre de gérer dynamiquement l'évolution de l'indexation des documents d'une collection. Du point de vue sémantique,

l'indexation avec une ontologie permet d'exprimer les relations entre des expressions du document à l'aide de celles des concepts auxquels les expressions sont associées dans l'ontologie. De plus, l'utilisation des concepts d'ontologie comme vocabulaire de référence d'index sert à bien préciser les sens accordés aux termes d'un document.

2.2. Indexation sémantique

Suite au développement du web sémantique et de ses technologies [Berners-Lee, 2001], différents travaux s'intéressent à son application en RI. L'utilisation d'une ontologie lors de la phase d'indexation permettrait de lever les ambiguïtés des sens des termes utilisés et de mieux représenter les connaissances inhérentes dans les documents. En termes d'indexation sémantique, des concepts de l'ontologie sont associés à chaque document selon les sémantiques qui y sont véhiculées. Dans cette section, nous présentons des méthodes et structures de données utilisées pour manipuler les index et ontologies en vue de la RI sémantique.

Différents travaux ont montré l'intérêt d'utiliser une indexation sémantique à base d'ontologie. Dans le domaine de l'apprentissage en ligne, [Chang et al., 2007] propose une indexation basée à la fois sur une ontologie du domaine de l'apprentissage et sur une ontologie dérivée de LOM (Learning Object Metadata), qui représente les métadonnées décrivant les ressources pédagogiques. Dans le cadre des recherches d'objets pédagogiques relatifs aux mathématiques en secondaire, les résultats montrent une meilleure efficacité en termes de rappel et de précision par rapport aux mêmes recherches basées sur mots-clés. De même, [Hernandez et al., 2008] utilise les termes d'une ontologie de domaine, associée à une ontologie de tâche et de scenario d'apprentissage comme valeurs des métadonnées de LOM. Afin d'accéder aux instances d'ontologie d'une part et aux index associés aux documents d'autre part, [Hernandez et al., 2007] propose de les stocker dans une base de données relationnelles.

Par ailleurs, [Song et al., 2005] propose un modèle de RI basé sur des ontologies de domaine, définies avec OWL lite. Les différentes ontologies de domaines sont intégrées pour former une ontologie unique. Les termes définis dans l'ontologie sont alors utilisés d'une part comme métadonnée pour annoter les contenus du web et d'autre part comme termes d'indexation de la collection.

Dans ces différents travaux, les structures de données utilisées permettent d'associer des concepts issus d'une ontologie aux documents de la collection. Cependant, le suivi de la dynamique (ajout, suppression, modification) des documents de la collection ainsi que l'impact de ces évolutions au niveau de l'index sémantique n'a pas été traité. Dans la section suivante, nous présentons le modèle de données que nous avons défini et qui permet une indexation sémantique dynamique.

3. Modèle pour une indexation dynamique basée sur une ontologie

Dans cette section, nous proposons les structures de données et les stratégies d'indexation nécessaires en vue de la mise en place d'un SRI basé sur une indexation sémantique par ontologies. Ces structures sont conçues pour permettre la mise à jour dynamique des index tout en permettant la recherche d'information sémantique.

3.1. *Paramètres et contraintes*

Chaque SRI est jugé au travers de ses performances en termes de satisfaction utilisateur sur les pertinences des documents retrouvés, du temps de réponse aux requêtes utilisateurs ainsi que la disponibilité du système. Ainsi, afin de définir les structures de données à utiliser dans les listes inversées, plusieurs paramètres qui entrent en jeu dans la performance des SRI doivent être pris en compte. Parmi ces paramètres, nous pouvons citer : la taille du corpus (nombre de documents constituant la collection), la fréquence de mise à jour de la collection, et le format des documents. En effet, la mise à jour de la collection demande la ré-indexation des documents. La taille du corpus affecte la durée de ré-indexation, cela peut engendrer une lenteur du système, voire son indisponibilité pendant un certain temps. De même, la fréquence de mise à jour de la collection a un impact sur la disponibilité de l'index. En effet, plus la fréquence de mise à jour de la collection est élevée, moins l'index est disponible pour la recherche d'information car il est à tout moment en cours de modification. Enfin, le format des documents affecte le temps d'indexation car les durées d'extraction des termes et expressions d'un document ne sont pas les mêmes pour tous les formats.

Ces paramètres liés à la collection se combinent avec les exigences des utilisateurs qui souhaitent obtenir des documents pertinents dans un meilleur délai et les principes utilisés en RI au moment de l'indexation. Ainsi, notre modèle prend en compte les contraintes suivantes:

- Minimisation du délai de mise à jour dynamique d'index. Dans les SRI actuels, suivant la taille du corpus, la mise à jour de l'index peut prendre des heures de traitement. Ceci limite donc les ajouts de nouveaux documents ou modification de documents dans le corpus et leur prise en compte dans les index. Selon notre approche, à chaque arrivée de nouveaux documents (qui peut emmener de nouvelles instances d'expressions), nous mettons à jour seulement les données statistiques du nouveau document et de ses expressions (fréquences d'apparition suivant le type de texte). Le calcul des poids des expressions et concepts se fait au moment de l'évaluation de la requête. De plus, la mise à jour des données suite à une modification de la collection ne s'effectue qu'au niveau des documents concernés.

- Minimisation du temps de réponse (traitement des requêtes). Le temps qui s'écoule entre la saisie de la requête et l'affichage de résultat doit se situer dans la limite acceptable par l'utilisateur. Le fait de traiter dynamiquement l'indexation des documents ne doit pas pénaliser de trop le temps de réponse du système lors d'une recherche de documents. La minimisation du temps d'indexation assure la disponibilité permanente de l'index. A son tour, la disponibilité de l'index permet d'évaluer les requêtes utilisateurs à tout moment. Dans notre modèle, toutes les informations statistiques et sémantiques qui servent à l'évaluation des requêtes sont accessibles dans la base.

- Pondération des termes d'indexation pour rendre compte de leur pouvoir discriminant. La pondération des expressions dans les index tient compte de la mesure tf*idf. Ce poids associé aux termes d'indexation [Robertson, 1976] est utilisé lors de l'étape d'appariement d'une requête avec les documents. Tf (Term Frequency) est la fréquence d'apparition d'un terme dans le document et Idf (Inverse Document Frequency) est la valeur de l'importance du terme dans l'ensemble de la collection. Dans notre modèle, la pondération tient également compte de certains éléments associés aux expressions dans le document comme leur mise en forme.

- Recherche sémantique sur les contenus. Le système recherche dans l'ontologie les concepts qui correspondent aux termes de la requête puis restitue les documents qui sont indexés par ces concepts.

Les structures de données qui sont présentées dans la section 3.2 ont été conçues pour répondre à ces contraintes et objectifs. Elles contiendront donc les éléments de l'index et aussi de l'ontologie qui sert de référence à l'indexation. Notons que le format des documents de la collection n'affecte pas les structures de données.

3.2. Modèle

Nous avons défini un modèle de données servant de socle à la mise en place d'un SRI basé sur une indexation sémantique par ontologie. Ce modèle présenté dans la figure 1 prend notamment en compte le double objectif d'actualisation dynamique des listes inversées et d'utilisation d'ontologies lors de l'indexation.

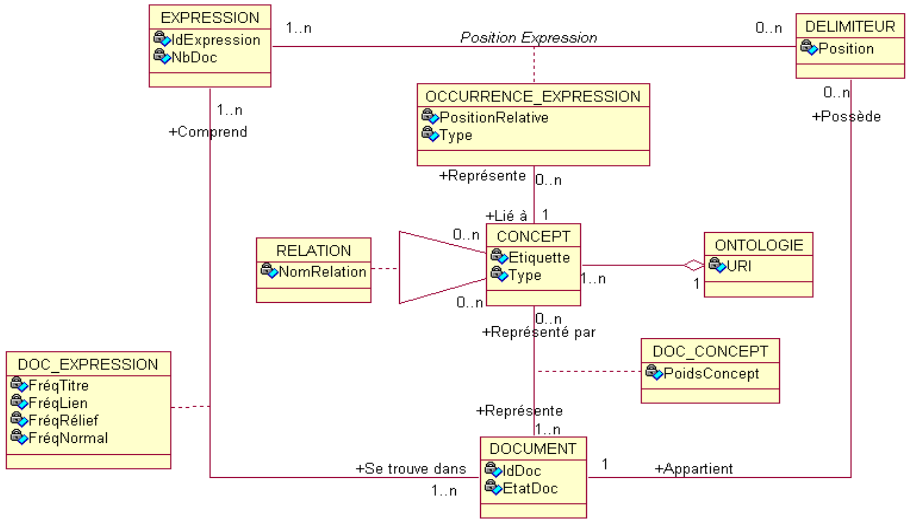

Figure 1 : Diagramme de classes représentant les données utilisées pour l'indexation.

La classe *DOCUMENT* représente les unités d'indexation. A chaque document est associé son identifiant (*IdDoc d'un document correspondant à son URI - Uniform Resource Identifier*), unique référence dans l'ensemble du système. Trois états possibles, impliquant des traitements différents, sont distingués pour un document. L'état d'un document peut être *normal* (cas des documents intégrés dans le système avec indexation à jour), *mis à jour* (cas des documents dont le contenu a été modifié depuis la précédente indexation) ou bien *effacé* (le document est retiré logiquement du système depuis la précédente indexation). Chaque document est subdivisé en plusieurs sections séparées par des marqueurs logiques. La classe *DELIMITEUR* représente ces marqueurs en précisant leur *Position* absolue dans le document.

Par ailleurs, chaque document est considéré comme un ensemble d'expressions décrites au travers de la classe *EXPRESSION*. Une expression est un groupe de termes représentant un concept du domaine. Le nombre de documents (*NbDoc*) où apparaît chaque expression est conservé. Chaque occurrence d'une expression apparaissant dans un document est repérée par sa position relative (*PositionRelative*) par rapport à un délimiteur du document. Cette idée est modélisée par l'association *Position Expression* et la classe d'association *OCCURRENCE_EXPRESSION*. La propriété *Type* conserve l'importance du texte incluant l'occurrence de l'expression (c'est-à-dire si celle-ci apparaît dans un titre, dans un lien hypertexte, dans un texte mis en relief ou dans un texte normal). La mémorisation des positions des

occurrences des expressions (*PositionRelative*) dans les documents offre des possibilités de localisation précise des index pour l'utilisateur.

De plus, chaque occurrence d'une expression peut être associée à un concept d'une ontologie de référence. La classe CONCEPT décrit les instances des concepts identifiées par leur *Etiquette* telle que définie dans l'ontologie. Cela permet de faire le lien avec la définition de l'ontologie repérée par son *URI* (classe *ONTOLOGIE*).

Une instance de concept peut être liée à d'autres par des instances de relations (ces relations étant définies entre concepts dans l'ontologie). Plusieurs relations pouvant exister entre deux concepts, il est nécessaire de conserver le nom de la relation concernée (*NomRelation*). La classe d'associations *DOC_EXPRESSION* rassemble les données statistiques concernant les expressions et leur apparition dans les documents (fréquence d'apparition dans les titres, dans des liens, dans des textes en relief, dans des textes normaux).

Enfin, la classe d'association *DOC_CONCEPT* précise les poids des concepts associés à un document. Un concept n'est pas forcément associé à une expression d'un document, il peut être affecté explicitement au document.

4. Exploitation du modèle

Le modèle de données sur lequel se base notre approche est axé sur l'utilisation d'ontologies lors de l'indexation, et principalement sur l'actualisation dynamique des listes inversées consécutive à l'évolution de la collection de documents et de l'ontologie. L'utilisation de la structure de données présentée dans la figure 1 est détaillée, d'une part, du point de vue de l'évolution de l'index, et d'autre part, de l'évolution de l'ontologie. Les données seront stockées dans une base de données Oracle pour faciliter leurs accès.

4.1 Ajout d'un nouveau document :

A chaque arrivée de nouveau document, l'utilisation de la structure de données suit l'algorithme 1 :

```
Début
   Créer une instance de DOCUMENT à l'état (EtatDoc) normal.
   Délimiter le nouveau document en blocs de paragraphes
   Pour chaque bloc lié à un délimiteur Faire    /* Intégration de
nouveau bloc  */
      Créer une instance dans DELIMITEUR
      Extraire les expressions décrivant le bloc
      Pour chaque expression Faire  /* Ajout d'expression */
         Si expression n'existe pas dans la classe EXPRESSION Alors
```

```
            Créer une nouvelle instance d'expression
            Créer une nouvelle instance de DOC_EXPRESSION
         FinSi
         Si instance de DOC_EXPRESSION n'existe pas Alors
            Créer une nouvelle instance de DOC_EXPRESSION liée à
         l'expression
            Mettre à jour la valeur de NbDoc dans EXPRESSION
         Sinon
            Mettre à jour les propriétés (fréquences d'apparition) pour
         l'instance de DOC_EXPRESSION
         FinSi
         Créer une instance de la classe d'association
         OCCURRENCE_EXPRESSION liée aux instances d'expression et de
         délimiteur en cours
         Moyennant d'une part des relations entre Termes et Concepts,
         et d'autre part les relations entre Concepts dans l'ontologie
         de domaine, identifier l'instance de CONCEPT correspondant à
         l'occurrence d'expression et les éventuelles relations
         auxquelles l'instance de CONCEPT participe.
      Fin Pour
      Identifier les éventuelles relations entre instances de CONCEPT
   Fin Pour
Fin
```

Algorithme 1 : Prise en compte de l'ajout d'un nouveau document

La suppression ou la modification d'un document implique dans un premier temps uniquement la mise à jour de son statut dans la classe *DOCUMENT* avant de mettre à jour toutes les autres classes.

4.2. Suppression d'un document :

Pour un document supprimé, son statut est changé en « Effacé » avant de mettre à jour les différentes informations relatives au document dans la base. Ainsi, ce document ne sera plus pris en compte par les requêtes. Après les mises à jour des données du document, le document sera supprimé physiquement de la collection. L'algorithme 2 correspond à la suppression d'un document.

```
Début
   Changer l'état du DOCUMENT à l'état (EtatDoc) Effacé. /* Le
   document sera ignoré par toutes les requêtes */
   Pour chaque délimiteur du document dans DELIMITEUR Faire
      Supprimer toutes les occurrences de OCCURRENCE_EXPRESSION liées
au délimiteur
      Supprimer le délimiteur
   Fin Pour
   Pour chaque instance de DOC_EXPRESSION liée au document Faire
```

```
      Décrémenter NbDoc de l'expression correspondante dans EXPRESSION
      Si NbDoc = 0 Alors Supprimer l'expression dans EXPRESSION
      Supprimer l'occurrence de DOC_EXPRESSION
   Fin Pour
   Supprimer les occurrences de DOC_CONCEPT relatif au document
   Supprimer le document dans DOCUMENT
   Supprimer physiquement le document de la collection
Fin
```

Algorithme 2 : Suppression d'un document

4.3 Modification d'un document :

A chaque modification d'un document, détectée à l'aide de la méthode *diff* [Ukkonen, 1985], son statut passe à l'état « Modifié » avant la phase de mise à jour des données relatives aux expressions du document. Une légère modification dans un document peut apporter ou supprimer des termes importants. L'importance de la modification en termes d'impact sur l'indexation ne sera connue qu'au moment de l'appariement du document avec une requête. Les éventuelles requêtes faisant appel au document en cours de modification utilisent les anciennes valeurs dans la base en attendant la fin des mises à jour des données relatives au document modifié. L'état du document redevient « Normal » après les mises à jour des données. Une simple réindexation du document modifié n'est pas viable sur un gros document. Les mises à jour qui peuvent être réalisées dans un document sont à l'origine des opérations suivantes :

- Ajout de nouvelles expressions
- Suppression d'expressions
- Modification d'une occurrence d'expression

Ajout de nouvelles expressions :

L'ajout d'expressions dans un document déjà indexé a pour conséquence de modifier l'indexation comme le montre l'algorithme 3.

```
Début
    Changer l'état du DOCUMENT à l'état  Modifié.
    /* L'affectation de cette nouvelle expression au bloc dans lequel
    il est inséré */
    Créer une occurrence dans OCCURRENCE_EXPRESSION
    Si instance de DOC_EXPRESSION n'existe pas Alors
       Créer une nouvelle instance de DOC_EXPRESSION liée à l'expression
       Incrémenter la valeur de NbDoc dans EXPRESSION
```

```
    Sinon
       Mettre à jour les propriétés (fréquences d'apparition) pour
       l'instance de DOC_EXPRESSION
    FinSi
    Mettre à jour les positions des délimiteurs (du document) dont la
    valeur de la propriété position est supérieure à celle du
    délimiteur du bloc contenant la nouvelle expression.

    Si la taille du bloc devient trop importante (le double de la
    taille normale d'un bloc) Alors /* le bloc courant est éclaté en
    deux. */
        Créer une nouvelle occurrence de délimiteur dans DELIMITEUR
        Pour chaque expression de la deuxième partie du bloc faire
               Affecter l'occurrence d'expression au nouveau délimiteur
               Mettre à jour la position relative par rapport au
        nouveau délimiteur
        FinPour
    FinSi
    Changer l'état du DOCUMENT à l'état  Normal
Fin
```

Algorithme 3 : Modification d'un document – ajout d'une expression

Suite à des successions de plusieurs ajouts d'expression, la taille du bloc peut devenir trop importante. Cela peut affecter la durée de mise à jour des instances de *DOC_EXPRESSION* dans le cas où la nouvelle expression est insérée au début du bloc. Pour palier ce problème, l'éclatement d'un grand bloc permet de réduire le nombre de mises à jour à effectuer dans *DOC_EXPRESSION*.

Suppression d'expression :

La suppression d'une expression entraîne une modification des instances du modèle comme l'indique l'algorithme 4.

```
Debut
    Changer l'état du DOCUMENT à l'état  Modifié
    Supprimer l'occurrence dans OCCURRENCE_EXPRESSION
    Décrémenter la propriété fréquence correspondante dans
    DOC_EXPRESSION
    Si la somme des fréquences est égale à zéro alors
        Supprimer l'entrée dans DOC_EXPRESSION
        Décrémenter NbDoc dans EXPRESSION
        Si Nbdoc=0 alors supprimer l'expression dans EXPRESSION
    FinSi
```

```
    Si la Taille du bloc courant < Seuil et le nombre de bloc du
    document >=2 alors /* Bloc devenu trop petit suite à plusieurs
    suppressions*/
        Si le bloc courant correspond au premier délimiteur du document
        alors Prendre le bloc suivant comme bloc courant
        Pour chaque expression du bloc courant Faire
            Affecter l'expression au bloc précédent
            Mettre à jour sa position relative par rapport au bloc
            précédent
        FinPour
        Supprimer le délimiteur du bloc dans DELIMITEUR
    Fin
    Changer l'état du DOCUMENT à l'état  Normal
Fin
```

Algorithme 4 : Modification d'un document – suppression d'une expression

Pour limiter le nombre de blocs dans un document, les blocs de petite taille seront fusionnés avec un autre bloc car un nombre important de blocs entraîne plusieurs mises à jour dans *DELIMITEUR* à chaque ajout d'un nouveau bloc. L'affectation des expressions à la fin du bloc précédent permet d'éviter de modifier toutes les positions relatives des expressions du bloc suivant.

Modification d'une occurrence d'expression :

La modification d'une occurrence d'expression se traduit par la séquence :

- Suppression d'une expression
- Ajout d'une nouvelle expression

Dans tous les cas (Ajout, Suppression ou Mise à jour de documents de la collection), la mise à jour dynamique de l'index n'est réalisée que sur les documents concernés. Cela permet de diminuer le temps d'indexation, et ainsi d'augmenter la disponibilité de la collection à tout moment. De plus, l'indexation dynamique permet de conserver une cohérence entre collection et index, ce qui permet au système de trouver les documents pertinents à tout moment. A l'indexation, aucune méthode de compression n'a été utilisée pour ne pas dégrader le temps de réponse.

5. Conclusions et perspectives

Cet article présente un modèle de données dans le cadre d'une indexation à base d'une ontologie de référence. Cette structure de données permet en outre une mise à jour dynamique et en temps réel des résultats de l'indexation lors de la mise à jour

de la collection de documents. Cette structure assure ainsi la cohérence permanente entre l'index, le corpus et l'ontologie de référence. L'avantage principal du modèle que nous proposons est qu'il n'est plus nécessaire de reconstruire l'index car il est à jour à tout moment. Ainsi, la structure que nous proposons permet de mettre en place une indexation sémantique dynamique.

Plusieurs perspectives sont envisagées pour ce travail. Une ontologie en tant que représentation de connaissances d'un domaine évolue pour représenter au mieux le domaine. L'ontologie qui a servi de base de référence pour le choix des termes d'indexation des documents peut donc évoluer. La cohérence de l'ontologie et de l'indexation des documents n'est alors plus assurée.. Notre prochaine étude consiste à l'étude de la prise en compte de cette évolution et son impact sur l'indexation des documents. Dans cet article, nous avons opté pour le calcul des poids des expressions au moment de l'évaluation d'une requête afin de simplifier le traitement et de gagner de temps à la mise à jour dynamique d'index. Partant de ce fait, nous comptons évaluer le temps nécessaire pour calculer le poids d'une expression, à partir des informations statistiques, et de le comparer avec celui de la lecture d'un poids pré calculé et enregistré dans la base. De la même manière, la complexité du système en temps d'exécution et de taille de données devra être évaluée suivant la taille de la collection, du format des documents et de la taille de l'ontologie. Nous sommes partenaire du projet Dynamo, soutenu par l'ANR. Les documents métiers des autres partenaires comme les fiches de diagnostic de pannes de voitures et les fiches d'incidents logiciels serviront de collection d'expérimentation.

6. Remerciements

Ces travaux s'inscrivent dans le cadre du projet de recherche Dynamo (Dynamic Ontology for information retrieval) soutenu par l'ANR. Cependant, les éléments présentés dans cet article n'engagent que leurs auteurs et n'est pas un résultat commun du projet Dynamo.

7. Bibliographie

Baeza-Yates, R.A., Navarro, G.: « Block addressing indices for approximate text retrieval». *Journal of the American Society on Information Systems* 51(1), p. 69–82 (2000).

Berners-Lee T., Hendler J., Lassila O., « The Semantic Web », *Scientific American*, p. 28–37, (2001).

Büttcher S., Clarke C.: « A Hybrid Approach to Index Maintenance in Dynamic Text Retrieval Systems ». *28th European Conference on IR Research, (ECIR),*, LNCS 3936, Springer Berlin / Heidelberg, p. 229-240 (2006).

Chang B., Ham D.H., Moon D. S., Choi Y. S., Cha J., «Using Ontologies to Search Learning Resources». *Computational Science and Its Applications – ICCSA 2007*, LNCS 4705, Springer Berlin / Heidelberg, p. 1146-1159, (2007).

Cho, J., Garcia M. H.: « The evolution of the web and implications for an incremental crawler ». *26th Intl. Conf. on Very Large Data Bases*, p. 200-209, (2000).

Cutting D. R., Pedersen and J. O.. « Optimization for Dynamic Inverted Index Maintenance ». *Proceedings of the 13th Annual International ACM SIGIR*. New York, USA ACM Press. P. 405–411, (1990).

Galambos L., « Dynamic Inverted Index Maintenance ». Proceedings of *World Academy Of Science, Engineering and Technology, Vol 11*, ISSN 1307-6884. p. 171-176, (2006).

Hernandez, N., Mothe, J., Chrisment, C., Egret, D.. « Modeling context through domain ontologies ». *Journal of Information Retrieval, Springer, Numéro spécial Contextual Information Retrieval Systems, Vol. 10 N. 2*, p. 143-172, (2007).

Hernandez, N., Mothe, J., Ralalason, B., Ramamonjisoa, B. , Stolf, P.: « A Model to Represent the Facets of Learning Objects ».: *Interdisciplinary Journal of E-Learning and Learning Objects, Informing Science Institute, Santa Rosa - USA, Vol. 4*, p. 65-82, (208).

Lim, L., Wang, M., Padmanabhan, S., Vitter, J.S., Agarwal, R.C.: « Efficient update indexes for dymnamically changing web documents ». Published online: 2 March 2007. http://www.cs.duke.edu/~jsv/Papers/LWP05.landmarkdiff.pdf, (2007).

Manber, U., Wu, S.: GLIMPSE: « a tool to search through entire file systems ». *Proceedings of the Winter 1994 USENIX Conf.*, p. 23–32. (1994).

Page, L., Brin, S.: « The anatomy of a large-scale hypertextual web search engine ». *Proceedings of the 7th Intl.WWW Conf.*, p. 107–117 (1998).

Robertson S. E., Sparck Jones K., « Relevance weighting of search terms ». In: *Journal of the American Society for Information Sciences*, 27 (3), p 129-146, (1976).

Salton, G., Allan, J., Buckley, C.: « Approaches to passage retrieval in full text information systems ». In: *Korfhage, R., Rasmussen, E.M., Willett, P. (eds.) Proceedings of the 16th Annual. Intl. ACM-SIGIR Conf*, p. 49–58. (1993).

Song J. F., Ming Z. W., Dong X. W., Hui L. G., Ning X. Z., « Ontology-based Information Retrieval Model for the Semantic Web ». *International Conference on e-Technology, e-Commerce and e-Service, EEE '05*, p 152 - 155, (2005).

Ukkonen, E.: « Algorithms for approximate string matching ». *Inf. Control 64*, p. 100–118 (1985).