Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de bachelor 2010

Filière Informatique de gestion

Gestion de la connaissance et recherche d'informations avec des archives de documents

Etudiant : Roger Schaer

Professeur : Dr. Henning Müller

Préface

Ce travail de bachelor traite de trois domaines principaux : l'extraction de texte et d'images de documents complexes, l'indexation de texte en vue d'une recherche de documents et la gestion des connaissances pour l'identification d'experts.

Ce sujet a été sélectionné par mon professeur responsable, Dr. Henning Müller, pour la raison suivante : les collaborateurs de l'institut de recherche « Informatique de gestion » de la HES-SO Valais disposent tous de plusieurs domaines d'expertises, mais il est actuellement difficile de déterminer de manière simple les connaissances d'une personne. La gestion des connaissances vise à rendre ces informations d'expertises accessibles à tous les membres de l'institut, afin de permettre l'identification d'un expert dans un certain domaine. De cette manière, les collaborateurs peuvent identifier rapidement une personne possédant une expertise donnée.

Une deuxième raison concerne la duplication de données existantes telles que des graphiques. En rendant les documents de l'institut accessibles en recherche, un collaborateur peut potentiellement trouver un document contenant du texte, un schéma, un graphique ou d'autres éléments réutilisables qu'il n'a pas besoin de reproduire.

L'automatisation est également un aspect important de ce travail. Le but est d'obtenir un maximum d'informations avec un minimum d'implication humaine. La gestion des connaissances, par exemple, se fait donc de manière automatique et ne nécessite pas d'annotations manuelles de la part des auteurs de documents.

L'objectif du travail de bachelor étant de répondre à un besoin réel, il serait intéressant de voir ce travail implémenté en pratique dans l'institut de recherche.

Remerciements

Je souhaite remercier mon professeur responsable, Dr. Henning Müller, pour son soutien et tous les conseils qu'il m'a fournis. Un grand merci également à Ivan Eggel, assistant à la HES-SO et auteur du travail de bachelor qui m'a servi de base, pour avoir toujours répondu rapidement à mes questions techniques. Finalement, je remercie ma mère pour ses encouragements et la relecture de ce rapport.

Structure du document

Le rapport se divise en quatre parties principales :

1. Introduction : décrit le contexte et les objectifs du travail (à partir de la page 1)
2. Méthodes : décrit les outils et technologies utilisées (à partir de la page 7)
3. Résultats : décrit le développement réalisé (à partir de la page 14)
4. Conclusions : diverses observations et réflexions sur le travail réalisé (à partir de la page 64)

Résumé

Deux solutions ont été mises en œuvre dans le cadre de ce travail de bachelor : la création d'un **moteur de recheche** pour les documents de l'institut de recherche « Informatique de gestion » de la HES-SO Valais et la mise en oeuvre d'un **système de gestion des connaissances** permettant d'identifier les domaines d'expertises des membres de l'institut.

Tout d'abord, une application Web effectuant les tâches suivantes a été développée :

- Exploration du partage réseau de l'institut afin d'établir une liste des documents.
- Extraction du texte et des images des documents.
- Indexation du texte extrait à l'aide de la librairie gratuite et « open-source » Lucene, un projet de la fondation Apache. Des métadonnées telles que l'auteur, le nom du fichier ou la langue sont également stockées.

La Figure 1 résume de manière visuelle les étapes citées ci-dessus :

Figure 1 - Résumé des étapes pour la mise en oeuvre du moteur de recherches

Une fois les données extraites et indexées, il est possible d'effectuer des recherches dans le contenu des documents. Pour ce faire, une page Web similaire à des moteurs de recherche tels que Google ou Yahoo a été créée. Une galerie d'images accompagne chaque document.

Les expertises des auteurs sont ensuite déterminées sur la base du contenu de leurs documents. Une liste de termes apparaissant dans de nombreux documents de l'auteur est établie à l'aide d'un autre projet de la fondation Apache, nommé Mahout. Les expertises déterminées se sont révélées être très pertinentes à condition de disposer d'un catalogue de documents complet (~100 documents) et varié (divers projets, anciens et nouveaux fichiers).

Finalement, l'application Web a été modifiée afin d'y incorporer des éléments relatifs aux expertises déterminées. Ces éléments comprennent, entre autres :

- Page de profil d'auteur listant ses expertises ainsi que ses liens avec d'autres auteurs dans des « Tag Clouds » (ou nuages de mots-clés).
- Diverses visualisations permettant d'afficher les liens entre auteurs et mots-clés de manière ludique et interactive. Exemples : « Tag Cloud » en trois dimensions, MindMap.
- Page de recherche de mots-clés permettant de trouver les auteurs associés à un terme donné.

Table des matières

1 Introduction

Cette section donne un aperçu global du projet. Tout d'abord, la problématique du projet est décrite, suivie d'une liste des objectifs à remplir. Elle est suivie d'un résumé succinct du travail. Les motivations de quelques choix stratégiques effectués durant le travail font suite au résumé. Une section concernant les étapes déjà implémentées et celles qui sont à réaliser conclut cette introduction.

1.1 Problématique

Toutes les institutions et entreprises produisent des documents structurés en grande quantité. Ces documents peuvent être utilisés pour gérer la connaissance des collaborateurs, mais ceci n'est actuellement que rarement pratiqué. Un système exploitant les connaissances contenues dans cette masse de documents permettra d'identifier des experts d'une entreprise de manière automatique. De plus, il sera possible de rechercher dans les documents produits par le passé afin d'éviter de reproduire des informations existantes telles que des graphiques [1].

1.2 Objectifs

Le but de ce travail est l'analyse des documents des instituts de recherche de la HES-SO Valais à Sierre [1]. Le travail se focalise plus particulièrement sur l'institut « Informatique de gestion », pour lequel des documents sont fournis par ses différents membres.

Le processus d'analyse se décompose en plusieurs étapes :

- Exploration d'un répertoire partagé contenant les documents produits par les différents membres de l'institut.
- Extraction du texte et des images de documents de types exploitables tels que PDF (Portable Document Format), Word, Excel, Powerpoint, HTML (HyperText Markup Language) ou texte brut.
- Indexation du texte extrait dans un moteur de recherche (Apache Lucene)
- Création d'une interface Web 2.0 permettant d'effectuer des recherches dans l'index généré.
- Analyse approfondie des documents et leurs auteurs afin de déterminer automatiquement des domaines d'expertise des professeurs et assistants de l'institut.
- Analyse des similarités et différences entre les différents auteurs.
- Adaptation de l'interface de recherche pour y inclure les connaissances définies pour chaque auteur (« tag cloud » résumant les domaines d'expertises d'un auteur, graphique représentant les liens entre auteurs et leurs domaines d'expertises, recherche d'experts pour un mot-clé donné, ...).
- Système de mise à jour permettant de rafraîchir les données.

1.3 Motivations

Cette section explique certains choix stratégiques faits durant ce travail et les motivations qui ont mené à ces choix.

1.3.1 Extraction automatique de la connaissance

Il existe globalement deux approches pour la gestion de la connaissance dans une archive de documents : l'annotation manuelle par les auteurs des documents ou l'extraction automatique. [2,3,4]

Dans la première variante, chaque collaborateur doit documenter lui-même tous ses fichiers, en y affectant des mots-clés (ou « tags »), par exemple. Cette méthode possède l'avantage d'une fiabilité élevée des informations mais repose sur la bonne volonté des collaborateurs à investir du temps pour créer cette valeur ajoutée. Un potentiel problème de cette approche est donc la pénurie d'informations, c'est-à-dire que pas ou peu de documents sont annotés, rendant difficile la découverte d'expertises des différents collaborateurs.

La seconde approche est celle qui a été choisie pour ce projet. L'extraction automatique ne nécessite aucune intervention des auteurs car elle se base sur le contenu de leurs documents. Une analyse statistique permet de déterminer quels sont les mots ou groupes de mots les plus intéressants. Contrairement à la première méthode, une abondance d'informations est à disposition et il s'agit d'identifier celles qui ont une réelle valeur. La fiabilité des résultats extraits est donc logiquement moindre par rapport à des termes sélectionnés consciemment par un auteur.

1.3.2 Affinement manuel de la connaissance

Après avoir extrait automatiquement des mots-clés pour plusieurs auteurs et perfectionné le filtrage de résultats non voulus, il est devenu clair qu'il est impossible d'obtenir 100% de termes pertinents.

C'est pourquoi un système de filtrage manuel (facultatif) a été mis en place afin que les auteurs puissent supprimer de leur profil les termes qui ne leur correspondent pas. Cette opération, peu coûteuse en temps devrait plus facilement être réalisée par les auteurs que l'annotation de chaque document publié.

De plus, l'historique des termes supprimés est conservé afin qu'ils ne réapparaissent pas dans le profil de l'auteur lors d'une nouvelle extraction / indexation des documents.

1.3.3 Extraction et affichage des images dans le moteur de recherche

Une image vaut mille mots, comme disait Confucius. C'est pourquoi le choix a été fait d'afficher directement à côté de chaque résultat d'une recherche les 3 premières images contenues dans le document. De cette manière, une personne pourra en principe déceler plus rapidement de quel sujet traite le document.

1.4 Étapes

Cette section décrit globalement les étapes nécessaires à la réalisation du système de gestion des connaissances. Notons que certaines étapes ont déjà fait l'objet d'un précédent travail de bachelor réalisé par Ivan Eggel en 2008, s'intitulant « Analyse komplexer Dokumente für die Extraktion und Indizierung von Texten und Bildern zur anschliessenden Informations-Suche ».

1.4.1 Étapes déjà implémentées

Une partie des étapes ayant déjà été réalisées, un court aperçu de ces dernières sera donné, avec en supplément la description des modifications apportées au système existant.

1.4.1.1 Extraction (Texte & Images)

Divers processus permettant l'extraction de texte et d'images ainsi que l'indexation de documents étaient mis en place. Ils permettaient d'agir sur un fichier chargé manuellement par l'utilisateur, un fichier disponible publiquement sur le Web ou un répertoire complet se situant sur le serveur de l'application.

C'est ce dernier processus qui a servi de base à la création du nouveau processus permettant d'agir sur un dossier partagé dans un environnement réseau Microsoft Windows (partage utilisant le protocole SMB[1]/CIFS[2]). Pour plus de détails, voir section « Extraction à partir d'un dossier partagé Windows » en page 14.

De plus, vu la durée relativement longue du processus complet (voir section 4.1.2.1 p. 67), la visualisation de la progression a été modifiée : au lieu de zones de texte affichant un message indiquant quelle étape est « en cours », des barres de progression dynamiques ont été implémentées afin de mieux informer l'utilisateur du progrès (voir Figure 2 p. 3 et Figure 3 p. 4).

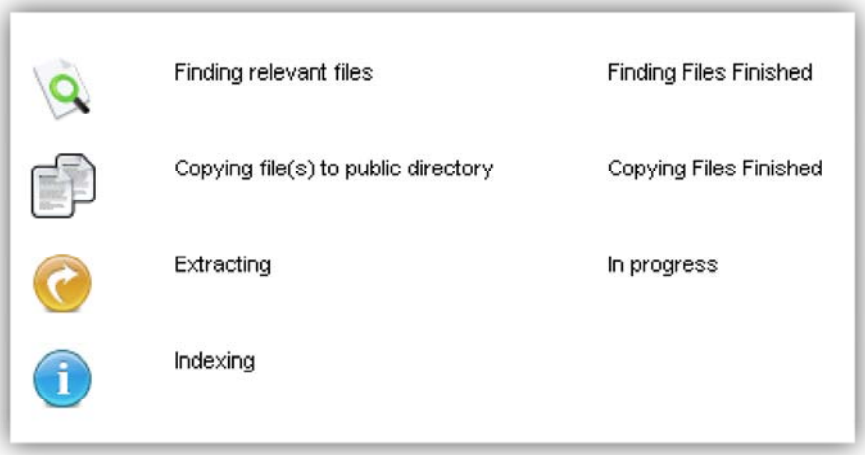


Figure 2 - Indicateur de progrès d'origine

[1] Server Message Block
[2] Common Internet File System

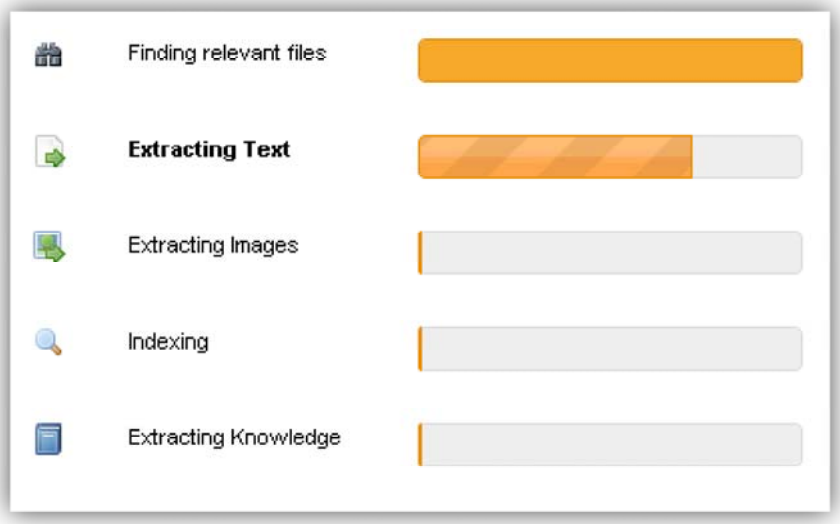


Figure 3 - Nouvelles barres de progression dynamiques

En outre, de nouveaux formats de fichiers sont supportés, tels que les documents Office 2007 (voir section « Différences d'extraction entre Office 2003 et Office 2007 » en page 15), ou les documents HTML.

1.4.1.2 Indexation (Texte & Images)

Le processus existant permettait d'indexer et de stocker le texte d'un document, le nom du fichier ainsi que le nom de l'auteur. De plus, l'adresse complète du fichier et un court extrait du document étaient également stockés.

La modification majeure apportée à cette étape est l'introduction de la détection du langage du document (voir section « Indexation multilingue » en page 17). Chaque document est analysé et indexé de manière différente selon le langage déterminé. De plus, cette nouvelle information est également stockée dans l'index du moteur de recherche.

M. Eggel avait également implémenté un processus d'indexation d'images, qui n'a pas été utilisé dans le cadre de ce travail.

1.4.1.3 Recherche (Texte & Images)

Une page de recherche fonctionnelle existait déjà et a été utilisée comme base pour ce travail. La page existante proposait d'effectuer une recherche soit par rapport au contenu des documents ou par rapport à leur auteur. Un système de pagination des résultats était déjà implémenté.

Principalement, des changements ont été effectués au niveau de la visualisation des images contenues dans les documents. D'une part, les trois premières images d'un document sont affichées à côté du texte (voir Figure 4 en page 5). D'autre part, le visionnement des images d'un document, qui était présenté sur une page distincte dans le travail de M. Eggel, est à présent géré par une galerie dynamique (voir Figure 5 en page 5).

Figure 4 - Images à côté d'un résultat de recherche

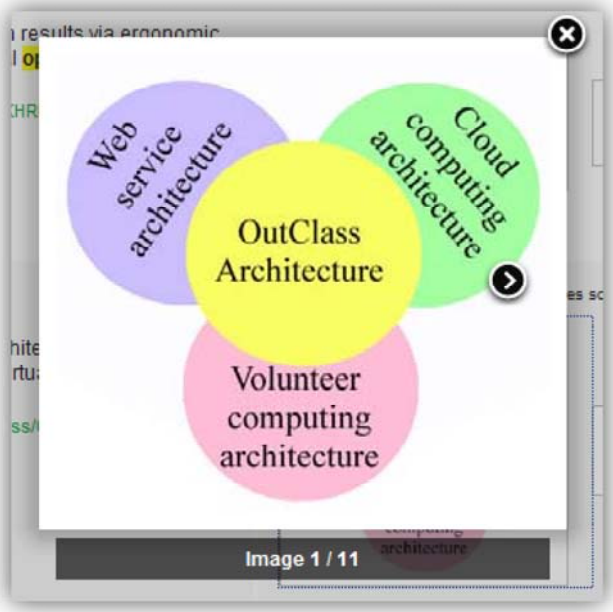


Figure 5 - Galerie dynamique des images d'un document

D'autres modifications ont été apportées au moteur de recherche, telles que l'ajout d'une fonction de remplissage automatique de la barre de recherche (voir section 3.2.5.1 p. 19 pour plus de détails) ou l'implémentation d'un système de mise en évidence du texte recherché dans les résultats (voir section 3.2.5.2 p. 20).

Des fonctionnalités de recherche d'images similaires à une image donnée existaient mais n'ont pas été inclus dans ce travail.

1.4.2 Étapes à réaliser

Sur la base du moteur de recherche implémenté par Ivan Eggel, agrémenté des nouvelles fonctionnalités décrites au point 1.4.1.3 en page 3, il s'agit de construire une application permettant de générer et de visualiser des profils spécifiques aux domaines d'expertises de chaque auteur, et ce de manière automatisée.

1.4.2.1 Création de profils d'auteurs

Afin de déterminer les domaines d'expertises d'un auteur, il faut établir une liste de mots-clés pertinents en se basant sur les documents qu'il a rédigés. De plus, il est intéressant de découvrir si l'auteur a des connaissances similaires à d'autres auteurs. Pour définir cette similarité, nous nous basons sur le nombre de mots-clés communs existant entre les auteurs. Toutes ces informations sont ensuite enregistrées dans un fichier représentant le profil de l'auteur.

Ce profil est composé des éléments suivants :

- **Nom complet de l'auteur**
- **Liste de mots-clés associés à l'auteur**
  Chaque mot-clé contient un identifiant, une pondération de 1 à 10 et le texte du mot-clé.
- **Liste de liens avec d'autres auteurs**
  Chaque lien contient l'identifiant de l'auteur (3 lettres), son nom complet et une pondération du lien entre 1 et 10.

1.4.2.2 Exploitation des profils et des liens

Les profils d'auteurs sont utilisés de la manière suivante : chaque auteur possède sa propre page de profil comportant entre autres un « Tag Cloud » des mots-clés qui lui sont associés (la valeur de la pondération est utilisée pour déterminer la taille du texte affiché) ainsi qu'un deuxième nuage de tags pour les liens avec d'autres auteurs.

Un fichier global comportant tous les liens est également généré durant le processus. Ce dernier peut être utilisé pour présenter de manière visuelle les informations à l'utilisateur sous différentes formes : Mind Map, Tag Cloud interactif en 3 dimensions, etc…

1.4.2.3 Mise à jour des données

Un processus peut être lancé périodiquement afin de mettre à jour les données à tous les niveaux : index du moteur de recherches, documents et leurs images, profil des auteurs, etc… Le tout est automatisé et peut être personnalisé : l'utilisateur a la possibilité de sélectionner les répertoires à inclure dans le processus d'extraction. De plus, les auteurs peuvent créer un fichier nommé « _noindex.txt » dans les répertoires contenant des informations confidentielles afin de les exclure du processus d'extraction et d'indexation.

2 Méthodes

Cette section traite des différentes technologies et outils utilisés durant la réalisation de ce projet. Une courte description de chaque entrée est fournie ainsi qu'une explication des choix faits.

2.1 Technologies utilisées

Les technologies utilisées durant ce projet proviennent essentiellement du monde « open-source » et plus particulièrement de la plateforme logicielle Java.

2.1.1 Plateforme Java

Bien plus qu'un simple langage de programmation, Java correspond à toute une suite de logiciels et de spécifications développées par l'entreprise Sun Microsystems [5], acquise en 2009 par Oracle Corporation [6]. Cette suite technologique permet de déployer des applications très diverses (sites Web, applications d'entreprise, applications mobiles, etc…) en faisant abstraction de la plateforme logicielle utilisée (Windows, UNIX, etc…) [5].

2.1.1.1 Langage de programmation Java

Le langage Java est orienté objet, multi-plateforme, performant et supporte très bien le « multi-threading ». [7] Java a plusieurs autres avantages qui le rendent très attractif pour la réalisation de ce travail:

- **Familiarité** : Java est actuellement le principal langage étudié durant la formation à la HES-SO. Des connaissances avancées et une expérience importante ont donc été acquises au cours des trois dernières années.
- **Popularité** : Java est un langage très populaire, ce qui a pour conséquence qu'il est extrêmement simple de trouver du support et des solutions à des problèmes sur Internet.
- **Librairies** : Java est très orienté sur le développement de librairies réutilisables. Tous les outils utilisés dans le cadre de ce travail sont disponibles en Java (et dans certains cas uniquement dans ce langage). Par expérience, il semble qu'une librairie existe pour quasiment toute fonctionnalité qui n'est pas directement intégrée au langage.
- **Solution existante** : La solution existante développée par M. Eggel utilise ce langage et ce travail se base sur celle-ci. Il est donc plus efficace de conserver la même technologie plutôt que de convertir tout le code existant dans un autre langage.

2.1.1.2 JavaServer Faces

JavaServer Faces (ou « JSF ») est un framework (ou « kit de de composants logiciels structurels »[3]) de développement d'applications Web basé sur le langage Java. [8]

Le framework est orienté MVC[4], ce qui signifie qu'une stricte séparation des couches de l'application est privilégiée. La couche de présentation est écrite en JSP[5] par défaut, mais d'autres langages de présentation comme XUL[6] sont également utilisables. La couche de traitement de données est implémentée sous forme de classes Java spécialisées nommées JavaBeans [8]. Ces JavaBeans ont la particularité d'être gérés par un serveur d'applications en termes de sécurité, de transactions, de persistance des donneés, etc… [9]

Il existe de nombreuses implémentations différentes de ce framework : celle qui a été utilisée dans le cadre de ce projet est nommée « Project Woodstock »[7]. Il s'agit du même framework utilisé dans le cadre du travail de M. Eggel. Il a été conservé pour des raisons de simplicité et de compatibilité du code, malgré qu'il s'agisse d'un projet abandonné en faveur d'un produit nommé ICEfaces[8] à la fin de 2008. [10]

Le problème d'une migration vers une solution plus actuelle est que les noms et le fonctionnement des composants (boutons, tableaux, listes déroulantes, etc…) diffèrent et un temps considérable aurait dû être consacré à la réécriture du code existant et aux tests.

Après une petite analyse des possibilités de Woodstock, il a été déterminé qu'elles seraient suffisantes dans le cadre de ce projet qui repose surtout sur un travail en arrière-plan plutôt que sur une couche de présentation complexe.

2.1.2 Ajax (Asynchronous Javascript And XML)

Ajax n'est pas un langage de programmation, mais bien une combinaison de techniques de développement Web utilisée du côté client (navigateur de l'internaute) visant à créer des applications Web interactives. Grâce à Ajax, les applications Web peuvent obtenir de manière asynchrone des données à partir d'un serveur sans interférer avec l'affichage ou le comportement de la page dans le navigateur du client. La communication se fait à travers des objets de type « XMLHttpRequest ». Notons qu'il n'est pas nécessaire d'utiliser le format XML durant la communication et qu'il est également possible de faire des appels synchrones. [11]

Ajax est utilisé dans le cadre de ce travail dans le but de rendre plus dynamique l'application : suggestions en temps réel dans la barre de recherche, galerie d'images, barres de progression, etc... Nous remarquons qu'il s'agit surtout de fournir une expérience plus visuelle et vivante à l'utilisateur.

---

[3] Source : http://fr.wikipedia.org/wiki/Framework (Accès le 9 août 2010)
[4] Modèle-Vue-Contrôleur
[5] JavaServer Pages
[6] XML User Interface Language
[7] Informations : https://woodstock.dev.java.net/index.html
[8] Informations : http://icefaces.org

2.2 Outils & Principales librairies utilisés

Cette section comprend une liste non-exhaustive des principales librairies et outils utilisés dans le cadre de ce projet. La section se focalise sur les librairies indispensables au fonctionnement de base du projet. Pour les parties moins critiques, les librairies seront indiquées au fur et à mesure du document.

2.2.1 Tika

Licence : Apache Software License 2.0 (http://www.apache.org/licenses/LICENSE-2.0.txt)
Site : http://tika.apache.org[9]
Version utilisée : 0.7

Figure 6 - Logo Tika

Apache Tika, anciennement un sous-projet de Lucene, est une suite d'outils permettant de détecter et d'extraire des métadonnées et du texte structuré de divers types de documents à l'aide de librairies existantes. [12] En effet, Tika utilise une multitude de librairies différentes pour chaque type de document :

- PDFBox : Un projet Apache également, cette librairie s'occupe de fichiers au format PDF. [13]
- POI : Un autre projet Apache permettant d'extraire le contenu de documents Office (97-2007). [14]
- TagSoup : Une librairie créée par John Cowan permettant de "nettoyer" du code HTML potentiellement mal formaté afin d'obtenir un document XHTML/XML valide. [15]. Le texte pur sans balises peut ensuite être extrait de manière fiable.

Alternatives envisagées

Après quelques recherches, il ne semble pas que des alternatives complètes à Tika existent. Il aurait bien évidemment été possible de trouver des alternatives pour la plupart des librairies d'extraction utilisées (PDFBox, POI, etc...), mais la simplicité d'utilisation et l'efficacité de Tika rendent peu avantageuse la programmation manuelle de l'extraction pour chaque type de document.

2.2.2 Lucene

Licence : Apache Software License 2.0 (http://www.apache.org/licenses/LICENSE-2.0.txt)
Site : http://lucene.apache.org[10]
Version utilisée : 3.0.1

Figure 7 - Logo Lucene

Apache Lucene est en soi un projet destiné au développement de logiciels de recherche open-source. Plusieurs implémentations du projet existent, telles que Lucene.net (.NET), PyLucene (Python) ou encore Lucy (C avec des liens vers Perl et Ruby). [16]

[9] Source du logo : http://tika.apache.org/tika.png
[10] Source du logo : http://lucene.apache.org/images/lucene_green_300.gif

Ce qui nous intéresse plus particulièrement est l'implémentation originelle en Java, utilisée dans le cadre de ce travail. Cette dernière propose des fonctionnalités d'indexation, de recherche, de correction orthographique, de mise en évidence des termes recherchés et d'analyse avancée. [16]

Plus globalement, Lucene Java (dorénavant nommée simplement « Lucene ») est une librairie de moteur de recherche très performante avec de nombreuses fonctionnalités écrite entièrement en Java. La technologie Lucene s'adapte très bien à toute application qui requiert des fonctionnalités de recherche "plein texte", tout particulièrement s'il s'agit d'une application multiplateformes. [17]

Alternatives envisagées

Le choix de Lucene était clairement défini dans la donnée du travail fournie par M. Müller [1], les alternatives ne sont donc données qu'à titre indicatif.

- **ht://Dig**[11] : Une librairie open-source sous licence GPL programmée en C++ permettant l'indexation et la recherche sur tout un réseau de serveurs Web (le logiciel émule le fonctionnement d'un navigateur Internet), la recherche dans des documents HTML et texte brut. Des fonctionnalités de recherches approximatives avancées (préfixes, suppression d'accents, synonymes, etc…) sont également fournies. [18] Les quelques points négatifs apparents après une courte exploration du site et d'avis d'internautes sont les suivants : le projet n'a pas été mis à jour depuis 2004 [19], alors que Lucene est encore développé très activement (dernière version apparue le 18 juin 2010) [20]. D'autre part, ht://Dig étant implémenté en C++, il aurait été plus difficile de le faire interagir avec le reste de l'application, développé en Java.
- **Terrier**[12] & **Solr**[13]: Deux plateformes de recherche complètes programmées en Java. La solution Solr se base sur le moteur Lucene et y ajoute d'autres fonctionnalités telles que l'intégration de Tika, des optimisations de performance, une interface d'administration, etc… [21] De son côté, Terrier fournit également une plateforme complète y compris une application de recherche "Desktop", une interface Web de recherche et le support de divers langages. [22] L'inconvénient de ces deux solutions est le fait qu'elles contiennent un bon nombre de fonctionnalités déjà implémentées dans le travail d'Ivan Eggel (interface de recherche) et d'autres qui peuvent être très facilement implémentées à la main avec probablement un contrôle du fonctionnement (ex : extraction avec Tika) probablement plus fin. Globalement, ces solutions paraissaient trop complètes pour les besoins de ce projet.

[11] Téléchargement : http://www.htdig.org/files/
[12] Source : http://terrier.org/
[13] Source : http://lucene.apache.org/solr/

2.2.3 Mahout

Licence : Apache Software License 2.0 (http://www.apache.org/licenses/LICENSE-2.0.txt)
Site : http://mahout.apache.org[14]
Version utilisée : 0.4-SNAPSHOT (version en développement)

Apache Mahout est un sous-projet de Lucene ayant pour but de fournir des librairies d'apprentissage automatique (machine learning) extensibles. [23] Mahout a été conçu pour fonctionner sur la base de Hadoop, un framework Java destiné aux applications distribuées et intensives en termes de traitement de données. [24]

Figure 8 - Logo mahout

Le point particulièrement intéressant de cet outil est la disponibilité d'algorithmes permettant d'extraire des groupes de mots (nommés également « n-grams ») importants d'un texte. Cette extraction peut être affinée avec de nombreux paramètres tels que la longueur maximale du groupe de mots, la fréquence minimale d'apparition par document, le pourcentage maximal de documents dans lequel le terme peut se retrouver, etc…

De plus, il est possible d'indiquer un analyseur Lucene personnalisé permettant de fournir une liste de « stop words » (mots trop communs à éliminer d'office) taillée sur mesure, de conserver ou de transformer les lettres accentuées, etc…

Alternatives envisagées

- **CollocationFinder**[15] : classe Java développée par Mark Harwood. Elle travaille directement sur un index Lucene et permet d'identifier des n-grams importants s'y trouvant. Malheureusement, cette classe n'identifie que les bigrammes (2 mots) et ne propose pas de fonctionnalités de tri selon la fréquence. Globalement, cette classe est trop incomplète pour être utilisée concrètement dans le cadre de ce travail.
- **Kea**[16] : il s'agit d'un algorithme d'extraction de phrases-clés à partir de documents texte. L'extraction peut se faire de manière libre ou à l'aide d'un vocabulaire contrôlé en rapport avec un certain domaine (exemple donné sur le site : l'agriculture).

  Malheureusement, cet outil nécessite une phase de formation, durant laquelle il faut lui fournir de nombreux documents annotés manuellement avec des phrases-clés (ce qui n'est pas dans l'optique de ce projet qui vise à automatiser tout le processus). [25]

[14] Source du logo : http://mahout.apache.org/images/mantle-mahout.png
[15] Source: https://issues.apache.org/jira/browse/LUCENE-474
[16] Source: http://nzdl.org/Kea

- **LingPipe**[17] : suite de librairies Java axées sur l'analyse linguistique de textes. Cette suite possède également une fonctionnalité d'extraction de phrases-clés. Malheureusement, les licences proposées par l'entreprise créatrice (alias-i) sont assez restrictives : la seule licence gratuite inclut une clause requérant la disponibilité publique des données traitées[18], ce qui ne peut être assuré dans le cadre de ce travail (certains documents fournis par l'institut sont confidentiels).

2.2.4 jCIFS

Licence : LGPL (http://www.gnu.org/licenses/lgpl-2.1.txt)
Site : http://jcifs.samba.org

La librairie jCIFS (Java Common Internet File System) permet de parcourir des répertoires partagés dans un environnement réseau Microsoft Windows à partir d'une application Java.

2.2.5 jQuery & plugins

Licence : MIT (http://github.com/jquery/jquery/blob/master/MIT-LICENSE.txt) ou GPLv2 (http://github.com/jquery/jquery/blob/master/GPL-LICENSE.txt)
Site : http://jquery.com[19]

jQuery est une librairie Javascript qui simplifie le développement d'applications Web riches. Elle gère l'exploration de documents HTML, la gestion d'événements, la création d'animations et les interactions Ajax. [26]

Figure 9 - Logo jQuery

Une grande force de jQuery est sa communauté très active qui met à disposition de nombreux plugins étendant encore les possibilités déjà très impressionnantes du cœur de la librairie.

jQuery a été sélectionné pour sa grande popularité, son extensibilité et des expériences personnelles positives vécues durant des projets précédents.

Les divers plugins utilisés sont présentés au fil du document.

2.2.6 NetBeans

Licence : CDDL1.0 et GPLv2 (http://netbeans.org/cddl-gplv2.html)
Site : *http://netbeans.org*

NetBeans est un environnement de développement intégré (IDE[20] en anglais) open source spécialisé dans la création d'applications Java (bien qu'il supporte de nombreux autres langages telles que PHP[21], C++ ou encore Ruby). [27]

[17] Source: http://alias-i.com/lingpipe/
[18] Source : http://alias-i.com/lingpipe/web/download.html
[19] Source du logo : http://static.jquery.com/files/rocker/images/logo_jquery_215x53.gif
[20] Integrated Development Environment
[21] PHP Hypertext Processor

Cet environnement a été sélectionné d'une part parce qu'il simplifie énormément le déploiement d'applications Web sur des serveurs d'applications locaux ou distants et d'autre part parce que ce logiciel était celui utilisé par Ivan Eggel durant la réalisation de son travail. De cette manière, il était possible d'ouvrir directement son projet et travailler dessus.

De plus, de nombreux modules supplémentaires existent pour NetBeans, facilitant encore la vie du développeur. Par exemple, le plugin « Visual JSF » fournit un éditeur graphique de pages Web (accélère la création de la base des pages).

2.3 Données utilisées

Pour des raisons de respect de la confidentialité, la recherche de domaines d'expertises était limitée à certaines personnes. Voici les personnes impliquées :

- Bruno Alves
- Laurent Bagnoud
- Yann Bocchi
- Florian Doche
- Nicole Glassey Ballet
- Jean Hennebert
- Anne Le Calvé
- Henning Müller
- Jean-Pierre Rey
- David Russo
- Anne-Dominique Salamin
- Michael Ignaz Schumacher

Certaines de ces personnes n'ont pas pu fournir de demandes de projets ou autres documents et les informations se basent donc sur les documents trouvés sur le système de publications scientifiques de la HES-SO[22]. Au total, 245 fichiers ont été fournis, pour un total de 288 Mo[23] de données.

[22] Site : http://publications.hevs.ch
[23] Mégaoctet

3 Résultats

Cette section représente la partie principale de ce dossier. Elle contient les explications techniques de toutes les étapes réalisées durant ce travail et donne également un aperçu global de l'architecture de la solution développée.

3.1 Données relatives à l'application

Cette section contient diverses informations de configuration et d'accès à l'application.

3.1.1 Accès à la page Web

Adresse du site: http://medgift.hevs.ch:8080/ExpertSearch

Nom d'utilisateur : expertsearch

Mot de passe : xperts3arch

3.1.2 Administration des tags des auteurs

Nom d'utilisateur : author

Mot de passe : auth0rPa55

3.2 Modifications apportées à la solution existante

Cette section passe en revue les quelques différences significatives apportées à la solution existante, non seulement en termes d'interaction avec l'utilisateur, mais également par rapport à des différences techniques importantes et des nouveautés particulières apparues depuis le travail de Bachelor d'Ivan Eggel en 2008.

3.2.1 Extraction à partir d'un dossier partagé Windows

À l'aide de la librairie jCIFS, il est très simple d'accéder à un partage réseau Windows. Il suffit de créer une nouvelle instance de la classe `SmbFile` et de lui passer en paramètre le chemin réseau auquel elle doit se connecter dans le format suivant :

```
smb://[admin]:[password]@[host]/[share]
```

Les termes entre crochets correspondent aux paramètres suivants :

`[admin]` : nom d'utilisateur pour accéder au partage

`[password]` : mot de passe correspondant à l'utilisateur

`[host]` : nom d'hôte DNS[24] de l'ordinateur cible

`[share]` : nom du partage

[24] Domain Name System

3.2.2 Différences d'extraction entre Office 2003 et Office 2007

Les documents Office 2007 se basent sur le nouveau format OOXML[25] [26] (**O**ffice **O**pen e**X**tensible **M**arkup **L**anguage) créé par Microsoft (communément appelé Open XML également). Il n'est donc plus possible d'utiliser les modules POI (**P**oor **O**bfuscation **I**mplementation) développés pour les versions antérieures de Microsoft Office (HWPF[27] pour Word, HSSF[28] pour Excel et HSLF[29] pour Powerpoint).

3.2.2.1 Texte

Des modules alternatifs compatibles avec le nouveau format Open XML ont été créés : ils se nomment XWPF[30], XSSF[31] et XSLF[32] pour Word 2007, Excel 2007 et PowerPoint 2007, respectivement. Ces derniers héritent tous de la classe `POIOOXMLTextExtractor` contenue dans le paquet `org.apache.poi` qui fournit des fonctionnalités supplémentaires permettant de récupérer les métadonnées Open XML [28].

3.2.2.2 Images

L'extraction d'images est également drastiquement différente dans Office 2007. Le nouveau format Open XML est essentiellement une collection de documents XML et autres médias compressés dans un fichier unique. La librairie OpenXML4J[33] qui fait partie intégrante de POI permet de décompresser les documents Office 2007 et de récupérer très simplement les médias qu'ils contiennent [29].

Des tests ont révélé qu'il est également possible de « dé-zipper » le fichier à l'aide d'un logiciel d'archivage tel que WinZip.

La Figure 10 en page 16 indique la structure des répertoires d'un document Word 2007 décompressé. Cette dernière est composée de divers répertoires et sous-répertoires. Celui qui nous intéresse est le dossier « media » contenu dans le répertoire « word » : c'est à l'intérieur de ce dernier que toutes les images insérées dans le document sont stockées (elles conservent leur format et leur taille d'origine).

---

[25] Standard ECMA (European Computer Manufacturers Association) international : http://www.ecmainternational.org/publications/standards/Ecma-376.htm
[26] Standard ISO (International Organization for Standardisation) : http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=45374
[27] Horrible Word Processor Format
[28] Horrible SpreadSheet Format
[29] Horrible Slide Layout Format
[30] XML Word Processor Format
[31] XML SpreasSheet Format
[32] XML Slide Layout Format
[33] Informations : http://poi.apache.org/oxml4j/index.html

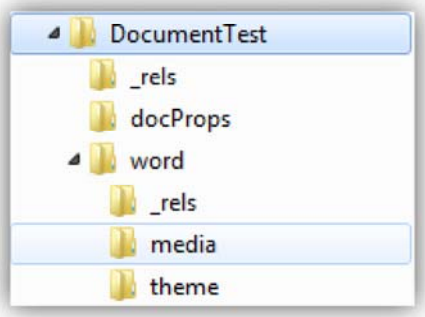


Figure 10 - Structure des répertoires d'un document Word 2007 décompressé

A l'aide de la librairie OpenXML4J, la tâche d'extraction d'images de documents Open XML devient donc pratiquement triviale[34]. Pour nos besoins, il a même été possible d'utiliser une méthode plus simple : à l'aide des classes Java utiles à la gestion de fichiers compressés (telles que `ZipFile` qui représente une archive ou `ZipEntry` pour chaque entrée dans celle-ci), il a été possible d'extraire le contenu d'un fichier OOXML et de récupérer tous les fichiers contenus dans le dossier « media » décrit dans le paragraphe précédent.

3.2.3 Extraction automatisée à l'aide de Tika

Un grand avantage de Tika, la nouvelle librairie utile pour l'extraction de texte de documents complexes, est sa classe « façade » nommée `Tika` qui permet avec une grande simplicité de détecter le type d'un fichier et d'en extraire le contenu. [30]

L'Extrait de Code 1 montre comment extraire le contenu d'un fichier :

```
Tika tika = new Tika();
tika.setMaxStringLength(-1);
InputStream input = new FileInputStream(path);
Metadata metadata = new Metadata();
metadata.set(Metadata.RESOURCE_NAME_KEY, file.getName());
String content = tika.parseToString(input, metadata);
```

Extrait de Code 1 - Extraction de texte avec Tika

[34] Exemple d'extraction : http://marxsoftware.blogspot.com/2008/02/using-openxml4j-to-access-office-open.html

Les étapes sont très simples :

- Créer un objet de type "`Tika`"
- Définir les paramètres de cet objet (p.ex la longueur maximale de texte qui sera extraite : ici elle a été définie à -1 pour extraire la totalité du contenu du document)
- Définir le flux d'entrée qui correspond au fichier à analyser
- Créer un objet de type "`Metadata`" pour stocker les métadonnées du document
- Définir les paramètres de l'objet de métadonnées (p.ex le nom du fichier)
- Récupérer le contenu du document à l'aide de la méthode "`parseToString`" de la façade de Tika

Il n'y a donc aucun besoin de se soucier du type de document fourni à Tika, qui va automatiquement détecter le type de fichier et utiliser la librairie appropriée pour l'extraction.

3.2.4 Indexation multilingue

Les documents de l'institut de recherche sont majoritairement en anglais, français et allemand. Il paraît donc logique de respecter cette diversité plutôt que de partir du principe que tous les documents peuvent simplement être indexés comme un texte anglophone.

Heureusement, Lucene permet de tenir compte des caractéristiques linguistiques des documents, grâce à de nombreux analyseurs adaptés à différents langages. Un analyseur est chargé d'extraire des éléments atomiques nommés « tokens » en anglais (la plupart du temps, un « token » correspond à un mot) qui seront indexés et d'éliminer tout autre contenu. [31]

Un analyseur peut avoir de nombreuses fonctionnalités supplémentaires, telles que l'élimination de « stop words », qui sont des mots très communs rarement porteurs d'information (« de », « la », « est », « un », etc…).

Une autre fonctionnalité est la transformation des tokens en lettres minuscules ou la suppression de signes diacritiques tels que les accents. Très pratique également, le « stemming » tente de raccourcir des mots pour n'en conserver que la racine, en se basant sur des règles grammaticales propres à la langue de l'analyseur. De cette manière, si l'utilisateur recherche le mot « grandes », tous les résultats comportant le terme « grand » et toutes ses variations seront également pris en compte.

Un dernier exemple est la reconnaissance de données qui possèdent un format bien précis (adresse e-mail, numéro de téléphone, date, etc…) afin de pouvoir les conserver dans leur forme d'origine. [31]

Afin de déterminer quel analyseur doit être utilisé pour chaque document à indexer, il faut disposer d'un moyen de reconnaissance de la langue de ce document. Ceci est fait à l'aide de l'API[35] Google Translate (et plus particulièrement une librairie Java simplifiant l'accès à cette API[36]) qui, mis à part la traduction, permet aussi d'obtenir le langage d'une chaîne de caractères transmise au service à l'aide de la méthode `getLanguage`.

L'Extrait de Code 2 en page 18 démontre comment cette fonctionnalité est implémentée dans le processus d'indexation :

```
Detect.setHttpReferrer("http://www.hevs.ch");

String documentFragment = doc.get("content").length() > 800 ?
doc.get("content").substring(0, 799) : doc.get("content");
documentFragment = URLEncoder.encode(documentFragment, "utf-
8");

DetectResult result = Detect.execute(documentFragment);
Language lang = result.getLanguage();

doc.add(new Field("language", lang.toString(), Field.Store.YES,
Field.Index.NO));
```

Extrait de Code 2 - Détection du langage d'un document

Voici les différentes étapes de la détection :

- Définir l'adresse d'origine de la requête (requis par Google pour établir des statistiques d'utilisation de leur service).
- Obtenir un fragment du contenu du document, d'une longueur maximale de 800 caractères (quantité suffisante pour déterminer le langage avec une certitude de 75% selon les tests effectués). Encodage de ce fragment pour qu'il soit valide en tant qu'URL[37].
- Exécuter la requête et récupérer la langue dans une variable `Language` .
- Ajouter l'information de la langue dans le document qui sera indexé dans Lucene.

Il est important de stocker dans l'index de Lucene le langage du document car il aura un impact sur le surlignage des termes cherchés (voir section 3.2.5.2 p. 20).

[35] Application Programming Interface
[36] Téléchargement : http://code.google.com/p/google-api-translate-java / Licence : LGPL
[37] Uniform Resource Locator

3.2.5 Fonctionnalités supplémentaires du moteur de recherche

Quelques nouvelles fonctionnalités ont été intégrées à la page de recherche afin de la rendre plus intéressante pour l'utilisateur et d'explorer les possibilités de Lucene.

3.2.5.1 Auto-complétion du champ de recherche

L'auto-complétion est une fonctionnalité permettant de fournir des suggestions de recherches en temps réel à l'utilisateur. Cette technique permet souvent d'accélérer les interactions homme-machine [32] et elle est utilisée de nos jours par d'importants acteurs du Web tels que Google, Yahoo ou encore Facebook.

Grâce à un plugin de la librairie Javascript « Prototype », la réalisation de l'auto-complétion a été grandement facilitée. La Figure 11 en page 19 montre un aperçu de la fonctionnalité.

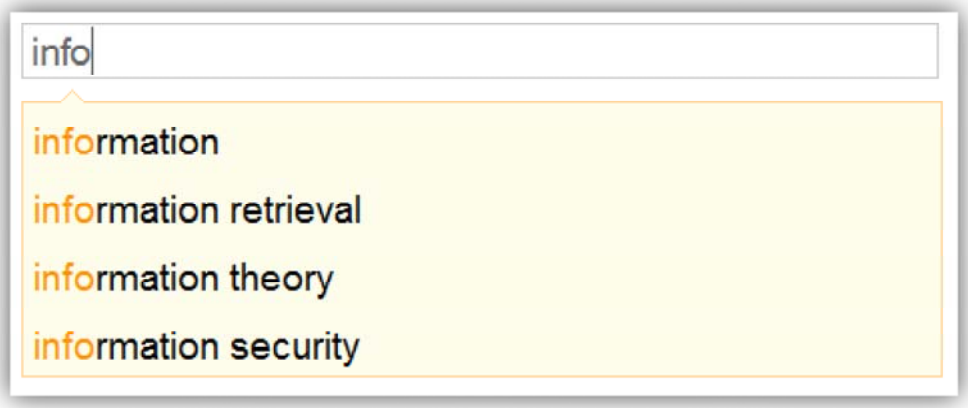


Figure 11 - Auto-complétion de la recherche

Afin de proposer des suggestions de recherche, une sélection de données doit bien sûr exister. Pour ce travail, il a été décidé que la source des suggestions se baserait sur l'historique des requêtes saisies par la collectivité (tous les utilisateurs).

La journalisation de ces requêtes se fait dans une base de données SQLite, un SGBD[38] très simple qui ne nécessite pas d'installation et d'exécution d'un service spécialisé. La base de données n'est constituée que d'une seule table comportant 2 colonnes :

- Une colonne « searchText » de type texte qui correspond à une requête entrée par un utilisateur. Cette colonne est la clé primaire de la table.
- Une colonne « occurrences » de type nombre entier qui indique le nombre de fois que cette requête a été saisie.

En conservant le nombre d'occurrences de chaque requête, il est possible de les trier par ordre de popularité lors de la saisie de l'utilisateur.

La communication entre la base de données et l'application se fait au travers du pilote JDBC[39] pour SQLite[40]. Ce connecteur est utilisé dans une classe `QueryLogger` comprenant deux méthodes :

[38] Système de gestion de bases de données
[39] Java DataBase Connectivity
[40] Téléchargement : http://www.zentus.com/sqlitejdbc / Licence : BSD

- `logSearchQuery(String query)` : Cette méthode prend la requête tapée par l'utilisateur en paramètre et l'insère dans la base de données. Si la requête est déjà présente, le nombre d'occurrences est incrémenté.
- `getSuggestions(String query)` : Le contenu de la zone de recherche est transmis à cette méthode qui retourne un ensemble de résultats (`ResultSet`). Un nombre maximal de 20 requêtes enregistrées commençant par les lettres passées en paramètre sont retournées. Ces dernières sont triées par ordre décroissant de leur nombre d'occurrences.

Cette classe est interrogée par un Servlet Java appelé au moyen d'une requête Ajax à chaque fois qu'un caractère est tapé dans la zone de recherche.

3.2.5.2 Surlignage des termes recherchés

Le surlignage (ou « highlighting ») permet de mettre en évidence chaque occurrence des termes recherchés dans les résultats trouvés. Le surlignage est une bonne pratique, car il permet à l'utilisateur de voir réellement où dans le document se trouvent les termes saisis dans la barre de recherche, augmentant l'utilisabilité d'une application. [33]

Dans le cas d'un moteur de recherche affichant un extrait du contenu au-dessous de chaque résultat, le surlignage n'est pas qu'une question de mise en forme. Il faut parcourir le document pour identifier un fragment contenant le ou les termes cherchés, en tenant compte du fait qu'il existe potentiellement plusieurs fragments possibles dont certains sont plus appropriés que d'autres.

C'est là qu'intervient la classe `Highlighter` de Lucene. Cette dernière fait partie du paquet de classes « contrib », signifiant qu'il s'agit d'une extension fournie par la communauté.

L'Extrait de Code 3 - Surlignage des termes d'une recherche en page 20 donne un court aperçu des opérations nécessaires à la mise en place du surglignage.

```
Analyzer analyzerForThisDoc =
AnalyzerDetermination.determineAnalyzerToUse(r.getLanguage());
...

Highlighter h = new Highlighter(
new SimpleHTMLFormatter("<span class='highlight'>", "</span>"),
...
);

h.setMaxDocCharsToAnalyze(200000);
...

String fragment = h.getBestFragment(analyzerForThisDoc,
"content", doc.get("content"));
...
```

Extrait de Code 3 - Surlignage des termes d'une recherche

Globalement, les étapes sont les suivantes :

- Il faut tout d'abord déterminer quel analyseur utiliser. Comme indiqué dans la section 3.2.4 en page 17, un analyseur spécifique est utilisé durant l'indexation. Cet analyseur possédant des propriétés spécifiques à la langue du document, il faut absolument utiliser le même dans la recherche de fragments appropriés. L'analyseur correct est simplement déterminé en lisant la propriété de langage stockée dans l'index Lucene auparavant.
- Le constructeur de l'objet `Highlighter` nécessite plusieurs paramètres dont un objet de formatage de code. Ce dernier détermine de quel contenu chaque instance de terme cherché sera entourée. Dans notre cas, chaque terme trouvé sera placé dans une balise HTML `<span>` avec une classe CSS[41] utilisée par la suite pour la mise en forme.
- Divers paramètres sont ensuite affectés au `Highlighter`, comme par exemple le nombre de caractères maximal à analyser. Cela permet d'éviter des ralentissements dus à des documents de plusieurs centaines ou milliers de pages.
- La méthode `getBestFragment` permet finalement de récupérer le fragment de texte le plus approprié. Ce fragment est attribué au résultat de la recherche et affiché dans l'application Web.

La Figure 12 en page 21 montre le terme « open source » mis en évidence dans un résultat de recherche.

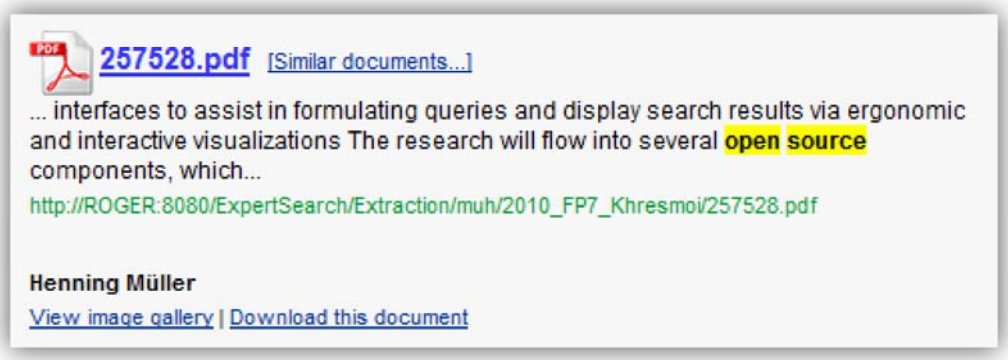


Figure 12 - Exemple de surlignage du terme "open source"

3.2.5.3 Galeries d'images interactives

Afficher les images d'un document sous forme de galerie dynamique possède 2 avantages. Premièrement, l'expérience est assez ludique grâce aux animations et transitions. De plus, les images sont mises en valeur (occupation maximale de l'espace disponible). Deuxièmement, l'utilisateur reste sur la page de recherche et n'a donc pas besoin de faire des aller-retours entre une page de visualisation d'images et la page des résultats de recherche.

Il existe de nombreuses solutions de galeries dynamiques, dont Fancybox[42], une librairie simple d'utilisation, polyvalente et open source bâtie sur jQuery. Elle s'est avérée excellente suite à des expériences personnelles faites par le passé.

[41] Cascading Style Sheets
[42] Téléchargement : http://fancybox.net / Licence : MIT ou GPLv2 (voir jQuery p. 10)

Pour éviter un surnombre de requêtes Ajax pour chaque clic sur une image, un tableau Javascript de toutes les images des documents trouvés est généré du côté serveur durant la construction de la page. C'est une des rares fois où la génération directe de code est utilisée (contrairement à la philosophie JSF de contrôles spécialisés liés à des propriétés et méthodes d'une classe Java). L'Extrait de Code 4 en page 22 détaille la gestion des galeries d'images.

```
<c:forEach
items="${TextSearchController.showListResults}" var="result"
varStatus="i">
picturesArray[${i.index}] = "${result.pictureNamesString}";
</c:forEach>

jQuery(".gallery").click(function(){
    fancy(this, picturesArray);
});


jQuery("a[rel^='imagegroup']").click(function(){
    fancy(this, picturesArray, true)
});
```

Extrait de Code 4 - Gestion des galeries d'images

Les 3 instructions fonctionnent de la manière suivante :

- La boucle « forEach » écrite en JSTL[43] est liée à la liste des résultats de la recherche contenue dans le contrôleur de la recherche. Pour chaque résultat, une entrée est ajoutée dans le tableau JavaScript « picturesArray », contenant une chaîne de caractères avec toutes les images du document.
- La première instruction Javascript gère le clic sur le lien « View Image Gallery » située au bas de chaque résultat. La méthode `fancy` n'est pas un appel direct à la librairie Fancybox, car une petite gestion du lien cliqué est encore nécessaire. C'est pourquoi le lien est passé en paramètre (this) tout comme le tableau des images (picturesArray).
- La seconde instruction Javascript fait le même travail, mais gère le clic sur une image située à côté d'un résultat. Le paramètre supplémentaire indique qu'un indice de départ doit être pris en compte (si le clic est fait sur la deuxième image, la galerie doit démarrer sur celle-ci).

[43] JavaServer Pages Standard Tag Library

3.3 Processus global

Une majeure partie de ce travail a consisté en la réalisation du processus d'extraction, d'indexation et d'analyse du contenu des documents de l'institut de recherche. Cette section détaille chacune de ces étapes. La Figure 13 p. 23 donne un aperçu du déroulement des étapes du processus global.

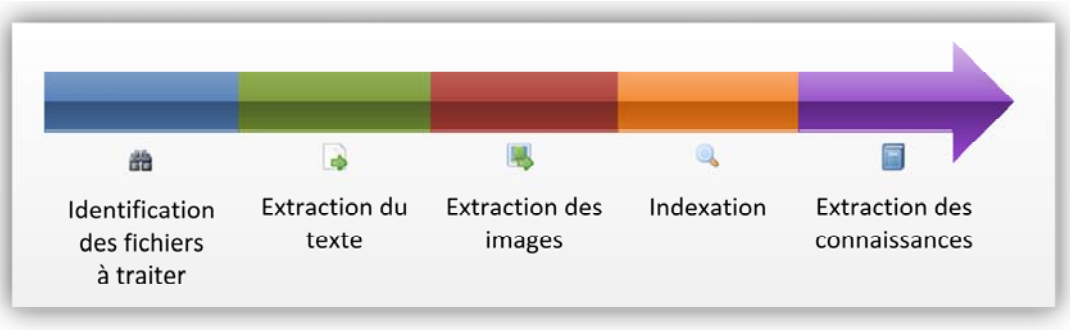


Figure 13 - Aperçu des étapes du processus global

3.3.1 Identification des fichiers à traiter

Avant toute extraction ou indexation, la première étape consiste à explorer le partage réseau Windows indiqué (voir section 3.5.1 p. 44) afin de constituer une liste des fichiers à traiter.

3.3.1.1 Types de documents supportés

Les types de documents suivants peuvent être traités par l'application :

- Documents Microsoft Word (97-2007)
- Feuilles de calcul Microsoft Excel (97-2007)
- Présentations Microsoft PowerPoint (97-207)
- Documents PDF
- Documents HTML
- Documents au format texte brut

Théoriquement, l'extraction devrait également fonctionner avec des documents Office 2010 puisqu'ils utilisent le même format de fichier qu'Office 2007, mais la version finale de ce logiciel n'étant pas encore disponible au début de ce travail, aucune garantie n'est faite compte au bon fonctionnement de l'extraction avec ces documents.

3.3.1.2 Protection de données confidentielles

Les auteurs des documents doivent pouvoir garantir la confidentialité des données qu'ils ne souhaitent pas voir publiées dans le cadre de l'application Web. Pour ce faire, il suffit à l'auteur de placer un fichier nommé « _noindex.txt » dans un répertoire. Ce répertoire et tous ses sous-répertoires seront alors exclus du processus d'exploration.

3.3.1.3 Déroulement de l'identification

Pour constituer la liste des fichiers à traiter, une méthode récursive `getFilesFromDir` va explorer et analyser chaque dossier et sous-dossier du partage.

La méthode prend 3 paramètres : Un objet `SmbFile` pouvant représenter un fichier ou un répertoire, un niveau de profondeur (permet de limiter la profondeur de l'exploration : la valeur configurée est 20) et un tableau des sous-répertoires sélectionnés (permet à l'utilisateur de cocher ou décocher les répertoires à explorer dans l'interface Web).

La méthode se déroule comme suit :

- Clause de finitude de la méthode récursive : si la profondeur atteinte est de 0, la profondeur maximale a été atteinte et la méthode termine son exécution.
- Récupération de la liste des fichiers du répertoire courant à l'aide de la méthode `listFiles`. Un filtre de noms de fichier est fourni à cette méthode afin de limiter les fichiers retournés aux types supportés (ce filtrage se base sur un tableau d'extensions de fichier acceptées).
- Pour chacun des fichiers de la liste, un test est réalisé : si le fichier courant n'est pas un répertoire, il est directement ajouté à la liste finale des fichiers à analyser. S'il s'agit d'un répertoire, un premier contrôle est effectué pour voir s'il contient le fichier « _noindex.txt » l'excluant de l'exploration.

  Si ce n'est pas le cas, un deuxième contrôle est effectué pour déterminer si le répertoire a été sélectionné pour extraction ou non par l'utilisateur (ce contrôle n'est effectué qu'au premier niveau de l'exploration).

  Finalement, la méthode est rappelée pour le sous-répertoire avec une profondeur décrémentée de 1.

Une fois l'exécution de la méthode terminée, une liste complète de tous les fichiers à traiter est retournée.

3.3.2 Extraction du texte

Une fois que la liste des fichiers à traiter est établie, l'extraction du texte peut être lancée.

La complexité de l'extraction du texte d'un fichier dépend du type de fichier : pour un document texte brut, il suffit de lire le fichier et de retourner son contenu. Pour un fichier structuré du type HTML, il est possible de supprimer toutes les balises et de conserver le texte uniquement.

Par contre, les fichiers Office 97-2003 et PDF sont des formats propriétaires complexes qui n'ont pas de structure directement lisible et interprétable.

Grâce à Tika, cette complexité est totalement masquée et l'utilisateur n'a qu'une seule commande à utiliser pour tous les types de fichiers.

Voici le déroulement global de la méthode d'extraction, qui va traiter chaque fichier de la liste fournie :

- Ouverture du fichier.
- Définition d'une métadonnée pour le fichier : nom du fichier.
- Récupération du texte du fichier avec Tika (voir section 3.2.3 p. 16).
- Définition du nom de l'auteur par rapport au nom du répertoire du fichier (chaque auteur possède son répertoire personnel sur le partage réseau). La section 3.7 p. 63 fournit une explication de la raison pour laquelle le nom d'auteur n'est pas récupéré dans les métadonnées du fichier.
- Copie du texte du fichier vers un répertoire du serveur (ces fichiers texte seront utilisés durant l'extraction des connaissances).
- Création d'un objet `InformationToIndex` contenant toutes les informations sur le fichier courant (chemin du fichier, contenu, auteur) et ajout de cet objet à une liste d'informations à indexer.

3.3.3 Extraction des images

Tika ne permet actuellement pas d'extraire de manière automatisée les images de tout type de documents comme il est possible de le faire avec le texte. C'est pourquoi une structure plus complexe a été nécessaire pour cette étape. La Figure 14 en page 25 représente la hiérarchie des classes créées pour l'extraction d'images.

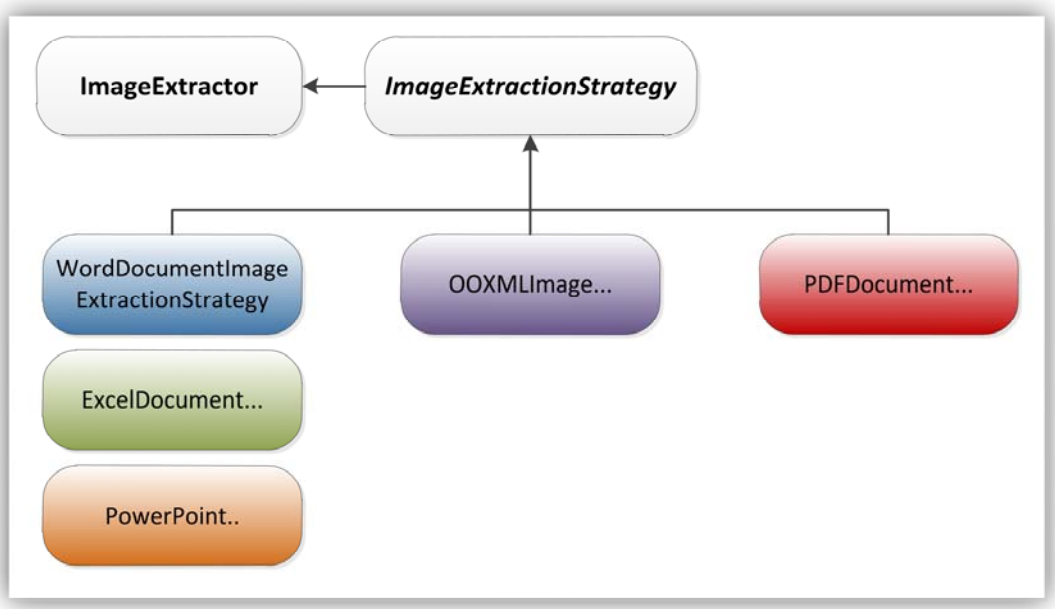


Figure 14 - Hiérarchie des classes destinées à l'extraction d'images

Le « design pattern[44] » nommé « Stratégie » a été utilisé dans la conception du modèle de ces classes. De cette manière, l'ajout de nouvelles stratégies d'extraction pour d'autres types de fichiers est facilité. En effet, le but de ce pattern est de définir une famille d'algorithmes, les encapsuler dans des objets et les rendre interchangeables, même durant l'exécution d'un programme. [34]

[44] Informations : http://en.wikipedia.org/wiki/Design_pattern

La classe `ImageExtractor` contient les méthodes principales d'extraction : copie des images sur le serveur, suppression d'images non désirées et création de miniatures. De plus, elle contient une référence vers un objet de type `ImageExtractionStrategy`. Cette classe abstraite contient les éléments communs à toutes les stratégies d'extraction, c'est-à-dire une liste d'images extraites et une méthode pour créer le répertoire de destination des images extraites. De plus, elle contient la méthode abstraite `extractImagesToDestination` qui doit être implémentée par toutes les stratégies d'extraction concrètes.

Chacune des stratégies concrètes (Word, Excel, PDF, etc…) va donc pouvoir déterminer une façon spécialisée d'obtenir les images du document en cours.

3.3.3.1 Suppression des images non désirées

Certaines règles métier ont été définies pour déterminer quelles images doivent être conservées. Afin d'éviter d'implémenter ces règles dans chaque stratégie d'extraction, un filtrage est effectué au niveau de l'objet `ImageExtractor` après la copie des images pour supprimer celles qui ne correspondent pas à ces règles.

Les règles définies dans l'application sont :

- **Dimension minimale** : l'image n'est pas prise en compte si la largeur ou la longueur de l'image est plus petite qu'une valeur définie (48 pixels dans notre cas).
- **Rapport longueur / largeur maximal** : l'image n'est pas prise en compte si le rapport entre la longueur et la largeur dépasse une certaine valeur (4 dans notre cas). Ceci évite de conserver des images telles que des en-têtes de documents.
- **Type d'image affichable dans un navigateur** : l'image n'est pas conservée si elle n'est pas d'un type qu'un navigateur Web peut afficher (JPEG, GIF, PNG ou BMP dans notre cas).

3.3.3.2 Création de miniatures

Certaines images contenues dans les divers documents de l'institut ont une taille et un poids relativement élevés (plus de 3 mégapixels et plus d'un Mo). Il serait donc peu judicieux d'afficher les versions originales de ces images à côté des résultats de recherche, et ce pour 2 raisons. Premièrement, une image de grande taille réduite par une règle CSS ou directement dans la balise HTML sera déformée et ne sera pas nette. La Figure 15 en page 27 illustre ce problème.

Figure 15 - Image non redimensionnée (gauche), redimensionnée (droite)[45]

La deuxième raison concerne le volume de données transféré : même réduites, le poids de l'image reste inchangé, ce qui représente une grande quantité de données en considérant qu'une page affiche jusqu'à 15 images simultanément (5 documents x 3 images).

La création de miniatures (ou « thumbnails ») est donc inévitable. Il existe plusieurs méthodes pour redimensionner des images en Java, mais celle qui fournit les meilleurs résultats est la librairie JAI[46]. [35,36] Grâce à des paramètres avancés d'interpolation et des opérations de redimensionnement spécialisées, le résultat est très satisfaisant : l'image est très lisse et si l'image contient du texte, ce dernier reste tout à fait lisible.

3.3.3.3 Stockage des images

Les images tout comme les miniatures sont stockées dans des répertoires spécifiques sur le serveur. Afin de simplifier l'enregistrement et le nommage des fichiers d'images, ces derniers conservent la même structure de répertoires que le partage réseau contenant les documents. La Figure 16 p. 28 illustre ce principe.

[45] Source de l'image : Document de l'institut de recherche
[46] Java Advanced Imaging / Téléchargement : https://jai.dev.java.net/binary-builds.html

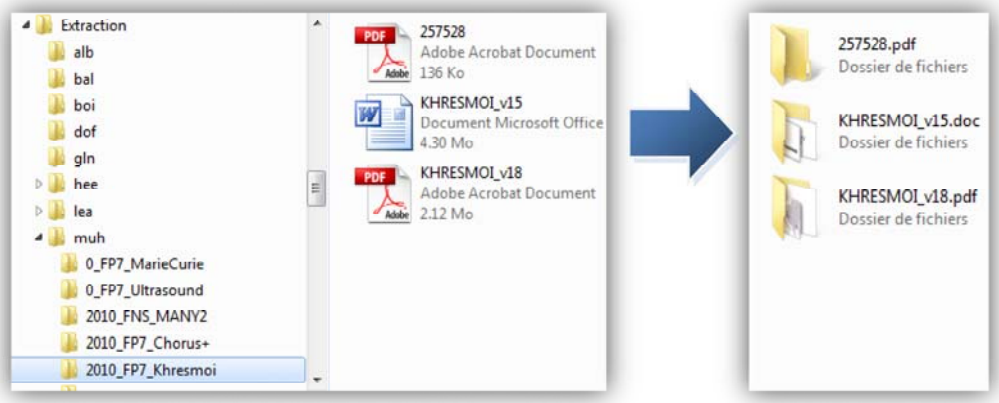


Figure 16 - Structure du partage réseau (gauche) & répertoire d'images (droite)

Chaque document contenu dans le répertoire d'origine se transforme donc en répertoire homonyme dans les emplacements d'images et de miniatures. Ces répertoires contiennent à leur tour les images du document concerné (ou leur version miniaturisée).

Cette méthode a l'avantage d'être facile à implémenter. Il est inutile de gérer les doublons des noms de document (2 documents de même nom ne pourront jamais être situés dans le même répertoire) et il n'est pas nécessaire de sauvegarder dans un fichier ou une base de données la liste des images correspondant à chaque document.

Le seul point qui mérite une attention particulière est la présence de caractères spéciaux dans les noms des fichiers ou des répertoires (lettres accentuées, espaces, ponctuation, etc…).

En partant du principe que les noms des fichiers et répertoires du partage de l'institut ne sont pas censés être modifiés, le plus simple est de conserver également les caractères spéciaux dans les noms des répertoires d'images et de miniatures. Il faut alors simplement veiller à utiliser de préférence l'encodage UTF-8[47] pour toutes les communications dans l'application Web et de convertir de manière correcte ces caractères en entités valides dans une URL (exemples : un espace peut être représenté par la chaîne « %20 », la lettre « é » par « %E9 »). [37]

3.3.4 Indexation

À la fin du processus d'extraction, nous possédons une liste d'informations prêtes à être indexées. Avant d'entrer dans les détails techniques de l'indexation, ce chapitre explique dans les grandes lignes ce qu'est l'indexation et comment fonctionne Lucene. La Figure 17 p. 29 décrit les fonctionnalités de Lucene et leur étendue. Elle est inspirée d'une figure disponible dans le livre « Lucene in Action » [31].

L'aspect le plus important de cette figure est la séparation des responsabilités : Lucene n'est pas un moteur de recherche complet et ne gère pas la totalité des tâches impliquées dans le processus d'une recherche.

[47] 8-bit Unicode Transformation Format

Il est donc du ressort de l'application de récolter les données à partir de diverses sources (bases de données, fichiers, pages web, etc…) ainsi que de gérer l'interaction avec l'utilisateur (récupération des requêtes de recherche et présentation des résultats).

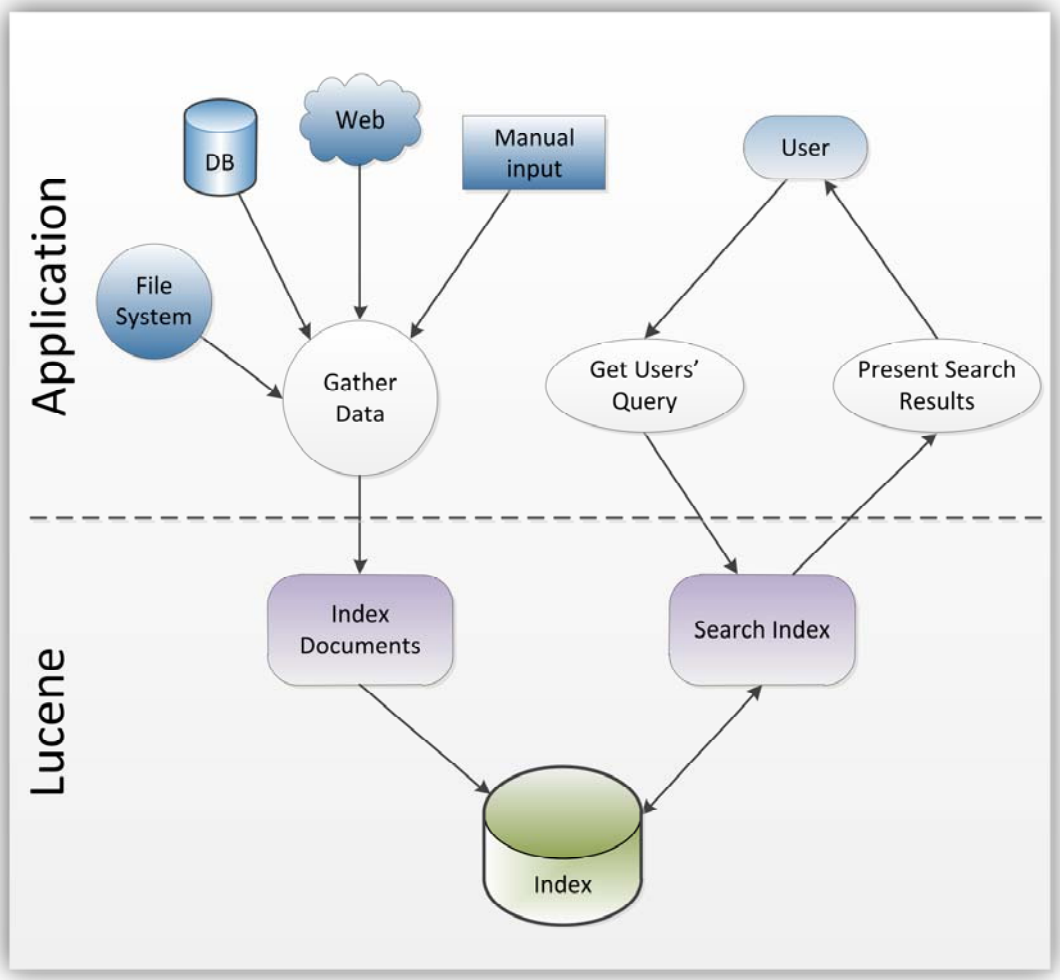


Figure 17 - Fonctionnement de Lucene

3.3.4.1 Qu'est-ce que l'indexation ?

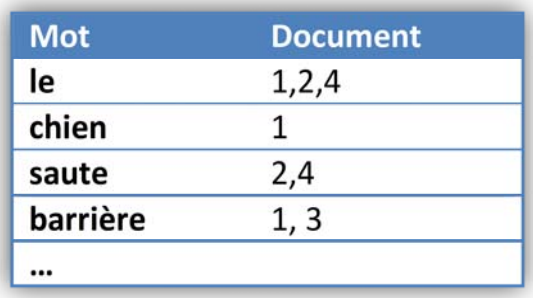
La recherche de documents contenant un ou plusieurs termes donnés est un concept pointu. Une première approche naïve serait de parcourir de manière séquentielle chaque document à la recherche du ou des termes. Les deux problèmes majeurs de cette solution sont la performance (l'analyse complète de chaque document prend un temps considérable) et l'évolutivité (plus il y a de documents, plus le processus sera long et la durée évoluera de manière linéaire).

L'indexation existe afin de régler ces deux problèmes. Son but est de transformer les documents en un format permettant une recherche rapide, ce qui élimine le problème de la lecture séquentielle. Les documents transformés sont alors stockés dans une structure de données que l'on nomme « index ».

Cet index fournit un accès aléatoire rapide, c'est-à-dire que l'ordre des documents n'a pas d'impact et que la quantité de documents a un impact moins prononcé que dans la première approche.

Un index de recherche fonctionne environ de la même manière qu'un index à la fin d'un livre, qui permet de localiser rapidement toutes les pages contenant un certain terme. [31]

Lucene utilise un index inversé, c'est-à-dire qu'au lieu de stocker pour chaque document la liste des mots qu'il contient, il stocke une liste des mots avec pour chaque mot la liste des documents le contenant. [38] La Figure 18 p. 30 est un exemple basique du fonctionnement d'un index inversé.



Mot	Document
le	1,2,4
chien	1
saute	2,4
barrière	1, 3
...	

Figure 18 - Exemple rudimentaire d'index inversé

Lucene stocke normalement cet index dans une série de fichiers sur le disque dur du système.

3.3.4.2 Comment se passe l'indexation avec Lucene ?

Lucene est composé de nombreux paquets et classes, mais seul un petit nombre est requis pour effectuer une indexation. [31]

Les classes requises sont les suivantes :

- IndexWriter
- Directory
- Analyzer
- Document
- Field

IndexWriter

La classe `IndexWriter` est la pièce centrale de l'indexation Lucene. Elle permet de créer un nouvel index ou d'ajouter des documents à un index existant. L'écriture étant la seule opération possible avec cette classe, il n'est donc pas possible d'effectuer de lecture ou de recherche à l'aide de l'`IndexWriter`. [31]

Directory

Une instance de la classe abstraite `Directory` est fournie au constructeur de l'`IndexWriter` afin de lui indiquer où stocker l'index. Comme il a été précisé dans la section 3.3.4.1 p. 29, la plupart des applications définissent un répertoire sur un disque dur pour le stockage.

La sous-classe de `Directory` nommée `FSDirectory` est prévue à cet effet. Il existe également une deuxième sous-classe de `Directory` permettant de stocker l'index

dans la mémoire vive (RAM[48]) de l'ordinateur (`RAMDirectory`). Cette alternative fournit des avantages en termes de performances (vitesse d'indexation et de recherche), mais n'est pas prévue au stockage permanent des données. Si l'ordinateur s'éteint, l'index est tout simplement perdu. [31]

La performance n'étant pas l'aspect le plus important dans ce travail, la première solution (la plus utilisée) a été choisie.

Analyzer

Le concept de base de l'analyseur a été expliqué dans la section 3.2.4 en page 17, mais quelques spécificités supplémentaires méritent d'être détaillées. Plusieurs implémentations concrètes de la classe abstraite `Analyzer` sont fournies avec Lucene. Voici celles qui sont utilisées dans le cadre de ce travail :

- **`StandardAnalyzer`** - L'analyseur le plus complexe faisant partie du cœur de Lucene. Il transforme les lettres en minuscules et élimine un bon nombre de « stop words » anglais. De plus, il est capable de reconnaître et préserver une multitude de données telles que : abréviations, noms d'entreprises, adresses e-mail, noms d'hôtes, nombres, mots avec apostrophe, numéros de série, adresses IP, etc…
- **`FrenchAnalyzer`** - Implémentation fournie par la communauté adaptée à la langue française (« stop words » français, règles de stemming particulières, etc…)
- **`GermanAnalyzer`** - Mêmes caractéristiques que le `FrenchAnalyzer`, mais adapté à l'allemand.
- **`IIGAnalyzer`** - Analyseur personnalisé utilisé pour l'extraction de connaissances. Il est basé sur le StandardAnalyzer mais possède une liste de « stop words » multilingue (anglais, français, allemand) beaucoup plus fournie (927 mots).[49]

Document

Un objet `Document` est une collection d'objets de type `Field`. Un `Document` peut représenter n'importe quelle source de données : e-mail, page web, enregistrement d'une base de données, document Word, chapitre d'un livre, etc… La source importe peu, car Lucene gère uniquement du texte (objets Java de type `String`). C'est le travail du développeur de transformer sa source de données en texte exploitable par Lucene.

Les champs (`Field`) du document représentent le contenu du document ou des métadonnées associées à ce contenu (nom de l'auteur, date de modification, sujet, etc…). [31]

Field

Comme indiqué dans la section précédente, un `Field` (ou champ) est le composant de base qui constitue un `Document`. Chaque champ est nommé et contient une donnée qui peut être retrouvée lors d'une recherche ou exploitée par accès direct.

[48] Random-Access Memory
[49] Sources des « stop words » : http://www.ranks.nl/resources/stopwords.html

Il existe trois paramètres permettant de configurer la manière dont est traité un champ : `Field.Index`, `Field.Store` et `Field.TermVector`.

Field.Index

Spécifie si un champ est indexé ou non, et si oui de quelle manière. Un champ indexé peut être retrouvé à l'aide d'une recherche. Voici les différentes valeurs les plus courantes de ce paramètre :

- **NO** - Le champ n'est pas indexé
- **NOT_ANALYZED** - La valeur du champ est stockée dans l'index telle quelle
- **ANALYZED** - La valeur du champ est traitée par un analyseur et le résultat est stocké dans l'index

Field.Store

Spécifie si le champ est stocké dans l'index. Le fait de stocker un champ signifie simplement qu'il est possible d'y accéder, mais pas à travers une recherche. C'est l'équivalent d'une colonne dans une base de données relationnelle. Les deux valeurs possibles sont :

- **NO** - La valeur du champ n'est pas stockée
- **YES** - La valeur du champ est stockée (sans aucun traitement)

Field.TermVector

Les « term vectors », ou vecteurs de termes, sont des informations concernant la fréquence d'apparition de chaque terme dans le contenu d'un champ. Ils peuvent être stockés si besoin.

Ces informations peuvent être très pratiques dans le domaine de la gestion des connaissances.

Les valeurs possibles sont :

- **NO** - Ne pas stocker les vecteurs de termes
- **YES** - Stocker les vecteurs de termes
- **WITH_POSITIONS** - Stocker les vecteurs + les informations sur la position de chaque instance du terme dans le texte
- **WITH_OFFSETS** - Stocker les vecteurs + les informations sur la distance entre chaque instance du terme
- **WITH_POSITIONS_OFFSETS** - Stocker les vecteurs ainsi que les informations sur la position et la distance

Configuration d'un champ

Le Tableau 1 en page 33 contient quelques exemples pratiques de configuration d'un champ pour divers types de données (les vecteurs de termes ne sont pas inclus dans ce tableau car ils sont surtout utiles pour le contenu d'un document) :

Analysé	Indexé	Stocké	Exemples d'utilisation
	✓	✓	Numéros de téléphone, numéro AVS[50], URL
		✓	Type d'un document (PDF, HTML, etc…) s'il n'est pas utilisé comme critère de recherche
✓	✓	(✓)[51]	Titre et contenu d'un document

Tableau 1 - Exemples de configurations d'un champ dans Lucene

3.3.4.3 Implémentation de l'indexation

Cette section explique la manière dont les différentes classes Lucene décrites précédemment sont utilisées dans le cadre de cette application.

Champs utilisés

Le Tableau 2 p. 33 est un récapitulatif des champs composant les documents de l'index :

Indexé	Analysé	Stocké	Vecteurs de termes	Nom du champ	Description
		✓		fileaddress	Chemin complet du fichier
		✓		abstracttext	Extrait du document
		✓		language	Langage du document
✓		✓		filename	Nom du fichier
✓	✓	✓		author	Auteur du document
✓	✓	✓	✓	content	Contenu du document

Tableau 2 - Champs des documents indexés

Les champs peuvent être globalement répartis dans 4 catégories :

- Les champs utilisés dans le contexte de l'application sans que l'utilisateur ne s'en rende forcément compte (chemin du fichier, extrait du document et langage). Il n'est pas nécessaire de les indexer, car ils ne feront jamais l'objet d'une recherche.
- Les champs qui peuvent potentiellement faire partie d'une recherche, mais qui doivent être conservés dans leur forme originale (nom du fichier). Afin de les conserver tels quels, ils ne sont pas soumis à l'analyse.

[50] Assurance-vieillesse et survivants
[51] Selon l'application, il n'est pas forcément nécessaire de stocker le contenu original du document

- Les champs qui peuvent faire l'objet d'une recherche partielle (auteur du document). L'utilisateur peut saisir uniquement le prénom ou le nom de famille de l'auteur : le champ doit donc être analysé.
- Le champ principal (contenu) qui fera l'objet de la majorité des recherches. Le contenu est stocké en plus d'être indexé afin de permettre le surlignage des termes recherchés (voir section 3.2.5.2 p. 20). De plus, les vecteurs de termes sont conservés pour la fonction de recherche de documents similaires (voir section 3.5.2.4 p. 49).

Stockage de l'index

L'index est stocké dans un répertoire du serveur à l'aide de la classe `FSDirectory`.

Écriture de l'index

Il est important de porter une attention particulière à la gestion de l'`IndexWriter`, car une seule opération modifiant l'index est autorisée à la fois. [31]

L'instance de l'IndexWriter est donc créée dès le démarrage de l'application Web et elle est disponible au niveau du contexte de l'application. Cela signifie que cette instance unique est accessible par toutes les autres classes de l'application. Qui plus est, la classe est « thread-safe[52] » et supporte donc sans problèmes des accès concurrents (la Figure 19 p. 34 illustre cela. Elle est inspirée d'une figure disponible dans le livre « Lucene in Action »). [31]

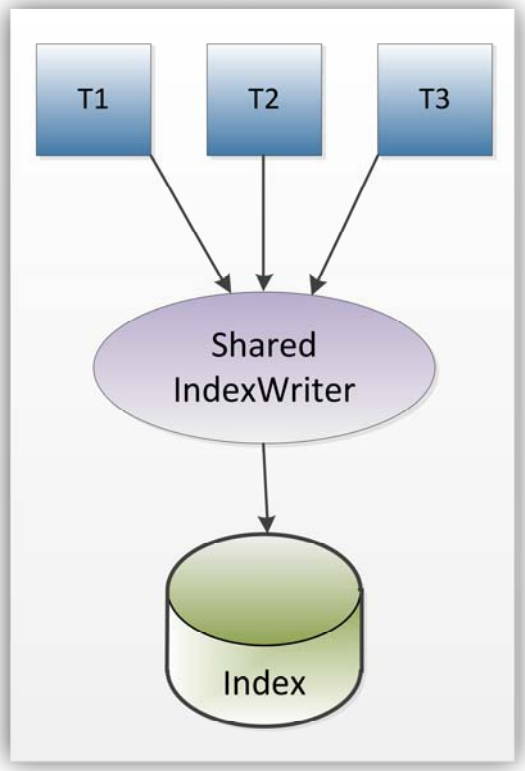


Figure 19 - IndexWriter partagé par plusieurs Threads

[52] Explication : http://en.wikipedia.org/wiki/Thread_safety

La gestion de l'`IndexWriter` au fil de l'application consiste en quelques méthodes clés détaillées dans l'Extrait de Code 5 en page 35. Cet extrait est issu de la classe `StandardFileIndexer` et plus particulièrement des méthodes suivantes :

- `indexInformationList`, qui gère l'indexation dans sa globalité
- `indexInformation`, qui représente le détail d'indexation d'un document.

Pour information, l'instance de l'`IndexWriter` est passée au constructeur de la classe `StandardFileIndexer`.

```
//indexInformationList
indexWriter.setMaxFieldLength(Integer.MAX_VALUE);
while(i.hasNext())
{
    this.iti = i.next();
    indexInformation();
    ...
}
indexWriter.commit();
indexWriter.optimize();
...
```

Extrait de Code 5 - Gestion globale de l'indexation

Le processus global se déroule de cette façon :

- La longueur maximale d'un champ est définie (cette valeur est de 10'000 par défaut, ce qui peut être trop peu pour certains longs documents). La valeur a donc été fixée à un nombre beaucoup plus élevé afin d'assurer l'indexation complète des documents.
- La liste des informations à indexer est parcourue et chaque information est ajoutée à l'index (voir Extrait de Code 6 p. 35).
- La méthode `commit` permet de valider tous les changements effectués dans l'index (ajout / suppression de documents, optimisations, etc...) afin de les rendre visibles aux lecteurs d'index `IndexReader` et de permettre la « survie » de l'index dans un état consistent en cas de panne de courant par exemple. [39]

```
//indexInformation
Document doc = this.getDocument();
Analyzer analyzer =
AnalyzerDetermination.determineAnalyzerToUse(doc);

if(analyzer != null){
    indexWriter.addDocument(doc, analyzer);
}else{
    ...
}
```

Extrait de Code 6 - Indexation d'un document individuel

Les points intéressants de cet extrait sont :

- Un objet `Document` Lucene est créé (tous les champs tels que le contenu ou l'auteur sont récupérés et affectés).
- L'analyseur du document est déterminé à l'aide du champ du langage contenu dans le `Document`.
- À l'aide de la méthode `addDocument`, le nouveau document est ajouté à l'index. Ce nouveau document n'est pas encore visible pour un objet de lecture `IndexReader` ou un objet de recherche `IndexSearcher` et ne le sera qu'une fois l'instruction `commit` réalisée sur l'`IndexWriter`. [39]

3.3.5 Extraction de la connaissance

Cette étape n'est en réalité pas directement liée à la fin de l'indexation des documents comme pourrait le faire croire le déroulement du processus global de l'application Web, car l'extraction va se baser sur le texte contenu dans les documents extraits, qui a été sauvegardé dans un répertoire du serveur durant l'étape d'extraction (voir section 3.3.2 en page 24).

Tout comme pour l'indexation, une petite partie théorique présentant les concepts et les outils utilisés dans l'application précède l'explication technique de l'implémentation.

3.3.5.1 Collocations & N-grammes

Principalement deux concepts de la linguistique et du traitement automatique des langues[53] sont utilisés dans ce travail : les collocations et les n-grammes.

Collocation

Dans le domaine de la linguistique de corpus[54], qui est l'étude du langage dans des ensembles de documents (données concrètes provenant du « monde réel »), une collocation est une séquence de mots ou de termes qui apparaissent côte à côte plus souvent que ce que permettent le hasard ou la chance.

Un exemple est donné par Michael Halliday[55], un linguiste britannique : « thé fort ». Alors qu'en théorie, le terme « thé puissant » a pratiquement le même sens, le fait est qu'un thé est généralement plutôt décrit comme étant « fort » plutôt que « puissant ». Le contre-exemple serait les ordinateurs, où le mot « puissant » est préférable à « fort ». [40]

Les collocations peuvent également posséder une valeur sémantique plus élevée que des mots individuels. Les termes « information » et « recherche » sont en soi quelque peu banals et peuvent apparaître dans un grand nombre de phrases ou expressions différentes. Par contre, la combinaison des deux termes, « recherche d'informations », a un sens plus précis et n'est probablement utilisée que dans un contexte bien défini.

[53] Informations : http://fr.wikipedia.org/wiki/Traitement_automatique_des_langues
[54] Informations : http://fr.wikipedia.org/wiki/Corpus
[55] Informations : http://en.wikipedia.org/wiki/Michael_Halliday

N-grammes

De manière générale, un n-gramme est un sous-ensemble de « n » objets d'un ensemble donné. Il peut s'agir de lettres, de mots, de phonèmes[56], etc…

Quelques termes spécifiques existent pour des n-grammes d'une taille donnée :

- 1 : unigramme (chien, chat, …)
- 2 : bigramme (un chien, ces chats, …)
- 3 : trigramme (un chien brun, les chats miaulent, …)

Au-delà de trois mots, le « n » est simplement remplacé par le nombre de termes (exemple : 4-gramme). [41]

Une collocation est donc un n-gramme, mais un n-gramme n'est pas forcément une collocation.

Dans le cadre de ce travail, les unigrammes ne sont pas conservés, car trop peu de termes intéressants arrivaient en tête des résultats extraits.

3.3.5.2 Algorithme LLR

L'algorithme LLR[57] permet d'identifier des collocations dans un ensemble de n-grammes. Il s'agit de l'algorithme implémenté par la librairie Mahout. Un score de log-likelihood (ou « vraisemblance logarithmique ») est calculé pour chaque collocation, indiquant son utilité relativement à d'autres combinaisons trouvées dans le texte. Les collocations obtenant les scores les plus élevés seront donc normalement les plus intéressantes. [42]

Le calcul du score se base sur deux événements, A et B, correspondant aux deux termes d'un bigramme. Un compte est ensuite fait du nombre d'occurrences où les deux mots apparaissent côte à côte, du nombre d'occurrences où chaque terme apparaît sans l'autre et du nombre d'occurrences où aucun des deux termes n'apparaît [42]. Le Tableau 3 p. 37 résume ce principe :

	Événement A	Tout sauf A
Événement B	A et B ensemble (nommé k11)	B sans A (k12)
Tout sauf B	A sans B (k21)	Ni A ni B (k22)

Tableau 3 - Comptes d'occurrences dans l'algorithme LLR

Une fois que ces occurrences ont été comptées, il suffit d'appliquer la formule suivante afin d'obtenir le score LLR de la collocation : [43]

$$LLR = 2 \times sum(k) \times (H(k) - H(rowSums(k)) - H(colSums(k)))$$

Le détail de la formule peut être consulté à l'adresse suivante:

http://tdunning.blogspot.com/2008/03/surprise-and-coincidence.html

[56] Informations : http://fr.wikipedia.org/wiki/Phon%C3%A8me
[57] Log-Likelihood Ratio

Nous constatons que cet algorithme n'est adapté qu'à des bigrammes. Actuellement, le calcul du LLR pour des n-grammes de taille 3 et plus fonctionne de la manière suivante : le premier mot est considéré comme l'en-tête du bigramme et le reste du n-gramme constitue le deuxième terme. Exemple : le terme « comme chien et chat » sera séparé en « comme » et « chien et chat ». [42]

3.3.5.3 Extraction de collocations avec Mahout

La version de Mahout utilisée dans le cadre de ce travail (0.4 SNAPSHOT) est axée sur une utilisation en ligne de commandes, mais il est également possible d'appeler ces méthodes dans un projet Java.

L'extraction se déroule en trois phases :

- Transformation des fichiers textes en fichiers de séquence utilisables par Mahout
- Extraction des collocations
- Transformation des fichiers de collocations en un format lisible par un programme Java

L'extraction des collocations se fait de manière individuelle, c'est-à-dire que le processus d'extraction sera répété pour chaque auteur.

Transformation des fichiers texte en fichiers de séquence

Mahout ne peut pas travailler sur des fichiers texte directement. Il faut d'abord les transformer dans un format spécial compréhensible par Mahout : les fichiers de séquence.

La classe `SequenceFilesFromDirectory` fournit la possibilité de transformer les fichiers textes contenus dans un répertoire en fichiers de séquence. [44] L'Extrait de Code 7 p. 38 montre l'instruction permettant de le faire.

```
//Transform text files into sequence files
SequenceFilesFromDirectory.main(new String[]{
    "--input", "...",
    "--output", "...",
    "--charset", "UTF-8"
});
```

Extrait de Code 7 - Transformation de fichiers texte en fichiers de séquence

Les paramètres à fournir sont les suivants :

- **input** - répertoire contenant les fichiers texte
- **output** - répertoire de sortie des fichiers de séquence
- **charset** - encodage des caractères des documents d'entrée (UTF-8 dans notre cas)

Extraction des collocations

Sur la base des fichiers de séquence, Mahout génère une multitude de fichiers contenant des statistiques sur leur contenu. Une collection de n-grammes constitue le fichier le plus utile pour nous. Il s'agit à présent d'extraire les collocations à partir de cette liste de n-grammes.

C'est la classe `SparseVectorsFromSequenceFiles` qui fournit cette fonctionnalité. [42] L'Extrait de Code 8 en p. 39 explique son fonctionnement.

```
//Create collocation files
SparseVectorsFromSequenceFiles.main(new String[]{
   "-i", "...",
   "-o", "...",
   "--minDF", 2,
   "--maxDFPercent", 60,
   "--maxNGramSize", 3,
   "--analyzerName",
ch.hesso.tb.rogerschaer.indexing.IIGAnalyzer,
   "--overwrite"
});
```

Extrait de Code 8 - Extraction de collocations à partir de fichiers de séquence

Les paramètres à fournir sont :

- **i** - répertoire d'entrée contenant les fichiers de séquence
- **o** - répertoire de sortie contenant les fichiers de collocations
- **minDF** - fréquence d'apparition minimale d'un n-gramme (par document) : permet d'éliminer des termes apparaissant un nombre insuffisant de fois dans les documents
- **maxDFPercent** - fréquence d'apparition maximale d'un n-gramme en pourcentage (sur la totalité des documents) : permet d'éliminer des termes trop communs apparaissant dans la majorité des documents (nom de l'auteur, en-tête ou pied de page, etc...)
- **maxNGramSize** - taille maximale d'un n-gramme
- **analyzerName** - nom de l'analyseur Lucene à utiliser (permet de fournir une liste de « stop words » spécialisée, par exemple)
- **overwrite** - permet d'écraser les fichiers existants

Notons l'utilisation de l'analyseur personnalisé `IIGAnalyzer` décrit dans la section 3.3.4.2, partie Analyzer, en page 31.

Transformation des données de collocation en format texte

Afin de pouvoir utiliser les fichiers de collocations extraits par Mahout, il faut les convertir en un format texte lisible par un programme Java standard.

C'est la classe `SequenceFileDumper` qui effectue cette tâche. [45] L'Extrait de Code 9 p. 40 détaille son utilisation :

```
//Transform sequence file to text file
SequenceFileDumper.main(new String[]{
    "--seqFile", "...",
    "--output",  "..."
});
```

Extrait de Code 9 - Transformation d'un fichier de collocations en fichier texte

Les paramètres à fournir sont :

- **seqFile** - chemin du fichier de collocations à transformer
- **output -** nom du fichier texte de sortie

La Figure 20 p. 40 illustre la structure du fichier de sortie :

```
Input Path: /home/roger/Dropbox/TravailBachelor/_Developpement/Mahout/Collocations/sad/wordcount/
Key class: class org.apache.hadoop.io.Text Value Class: class org.apache.hadoop.io.DoubleWritable
...
Key: cadre: Value: 2.0
Key: cament: Value: 2.0
Key: caméra: Value: 4.0
Key: cas: Value: 2.0
```

Figure 20 - Fichier de collocations au format texte

3.3.5.4 Transformation des données de Mahout au format JSON[58]

Le fichier texte récupéré à partir de l'étape précédente ne peut être utilisé tel quel. Un processus de lecture, de filtrage, de formatage puis d'écriture est requis afin d'obtenir des données exploitables par l'application Web.

Le fichier texte généré par Mahout est trié par ordre alphabétique des termes. Nous nous intéressons plus aux scores obtenus par chaque terme. Il s'agit donc de lire le fichier est de le trier par ordre décroissant du score.

Lecture & Filtrage des n-grammes

Le fichier mahout est lu ligne par ligne, puis la ligne est séparée en ces différentes composantes (n-gramme et score). Le n-gramme est ensuite soumis à plusieurs tests de filtrage secondaires permettant de réduire le nombre de résultats non pertinents :

- Le n-gramme n'est pas un unigramme (contient un espace au minimum)
- Le n-gramme ne doit contenir que des lettres et des espaces : les valeurs numériques sont éliminées.
- Le n-gramme ne doit pas contenir de composants d'une seule lettre (ex : « a dog » ne sera pas retenu).

[58] Javascript Object Notation

- Le n-gramme ne contient pas de termes contenus dans une liste de « stop n-grammes » établie spécifiquement pour les documents de l'institut : cette liste contient des termes qui n'ont pas de valeur dans le contexte de l'institut de la HES-SO, tels que « haute école» ou « work package ».
- Le n-gramme ne débute ou ne se termine pas par un mot très court (2 caractères ou moins). Ceci permet de filtrer certains n-grammes débutant par un mot qui est potentiellement un « stop word », mais n'a pas été inclus dans la liste des « stop words » de l'analyseur. Exemple : le n-gramme « de recherche ».
- Le n-gramme ne fait pas partie de la liste de termes supprimés par l'auteur (voir section 3.5.5 p. 53).

Si le terme remplit toutes les conditions, il est ajouté à la liste des termes conservés. Une fois cette liste complète, elle est triée à l'aide d'un objet `Comparator` Java personnalisé nommé `CollocationScoreComparator`. Cette classe contient une méthode `compare` permettant de définir un tri personnalisé (dans notre cas, le tri se rapporte au score du n-gramme et doit être décroissant).

Génération des fichiers JSON

JSON est un format de données standardisé très compact qui permet facilement de stocker des informations. La plupart des visualisations de données utilisées dans le cadre de ce travail nécessitent une source de données au format JSON. C'est pourquoi il a été sélectionné comme format pour toutes les données générées.

Le processus de conversion est plutôt long, mais les étapes principales sont les suivantes :

- Pour chaque auteur, un objet JSON est créé. Le nom de l'auteur y est ajouté ainsi qu'un tableau de tous les n-grammes qui lui sont associés (y compris une valeur normalisée du score comprise entre 1 et 10). Simultanément, une liste des professeurs associés à chaque n-gramme est maintenue.
- Une fois toutes les données des auteurs disponibles, la liste des professeurs par n-gramme est examinée afin d'établir les connexions entre les auteurs possédant des n-grammes communs. Ces connexions sont ensuite insérées dans l'objet JSON de chaque auteur.
- Vient alors l'écriture des fichiers. La liste des auteurs par n-gramme est sauvegardée, car elle est utile dans l'application Web (voir section 3.5.6 p. 53). Ensuite, chaque profil d'auteur est écrit.

Le Tableau 4 p. 42 résume les informations contenues dans le profil d'un auteur.

Champ JSON	Valeur	Exemple
name	Nom de l'auteur	« Henning Müller »
tags	N-grammes associés à l'auteur (avec pondération)	{**tags** : [{"**id**":"largescale","**weight**":5,"**tag**":"large scale"}, ...]}
links	Liens avec d'autres auteurs (avec pondération)	{**links** : [{"**id**":"lea","**weight**":1,"**name**":"Anne Le Calvé"}, ...]}

Tableau 4 - Contenu d'un profil d'auteur

Génération du fichier pour la visualisation MooWheel

La visualisation MooWheel (décrite dans la section 3.6.5 en page 61) nécessite un format de données particulier. C'est pourquoi toutes les informations récoltées durant l'étape précédente (la liste d'auteurs par mots-clés, la liste des n-grammes par auteur avec leur pondération ainsi que la liste des objets JSON des auteurs) sont réutilisées et reformatées afin de correspondre à la structure requise.

Mis à part les noms des champs JSON qui sont différents, les liens doivent être définis de manière bidirectionnelle, c'est-à-dire que les liens des auteurs vers les n-grammes doivent également être établis des n-grammes vers les auteurs. L'Extrait de Code 10 p. 42 est un extrait du fichier de données MooWheel.

```
{
"id":"bal",
"text":"Laurent Bagnoud",
"connections":[
                    ["hee",1],["rua",1],["muh",1],["boi",5],
                    ["opensource",4]
                ],
"type":"professor"
}
```

Extrait de Code 10 - Extrait du fichier de données de la visualisation MooWheel

Notons qu'une nouvelle information est incluse dans ce fichier. Le champ « type » permet de différencier les éléments de type « professor » (correspond à l'auteur) des éléments de type « keyword » (n-grammes). Cette distinction est faite pour permettre une distinction visuelle et comportementale de ces deux types d'éléments dans la visualisation.

3.4 Architecture de l'application Web

L'application est exécutée sur un serveur d'applications Glassfish et utilise de nombreuses technologies et concepts différents. La Figure 21 p. 43 dresse un panorama des divers éléments auxquels accède l'application Web.

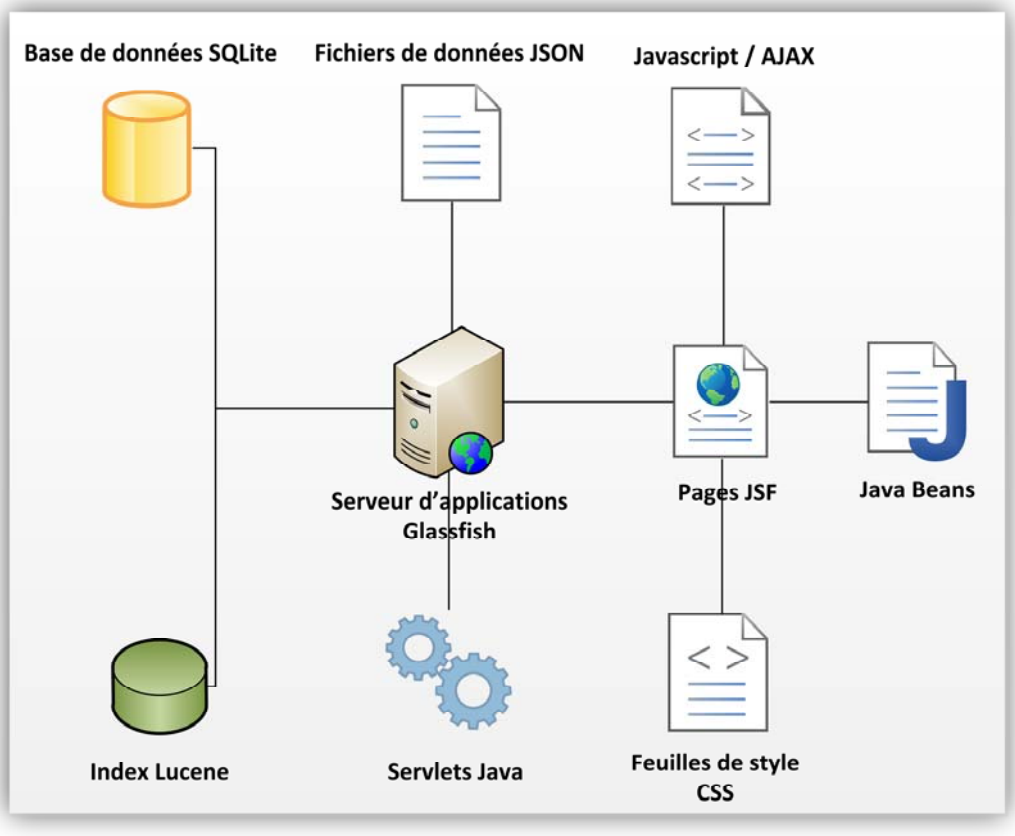


Figure 21 - Interactions du serveur avec les différentes technologies utilisées

L'application accède à trois sources de données principales :

- La base de données SQLite pour l'auto-complétion des requêtes de recherche.
- L'index Lucene pour l'indexation et les recherches.
- Divers fichiers JSON pour les visualisations et les pages de présentation de données (profils d'auteurs, par exemple).

Les pages Web au format JSF sont structurées de la manière suivante :

- Page JSF qui représente la structure du document HTML qui est affiché à l'utilisateur.
- Feuilles de style CSS liées à ces pages JSF pour la mise en forme des éléments (disposition du contenu, couleurs, taille du texte, etc…).
- Classes Java au format EJB contenant l'intelligence de l'application. Ces classes sont une source de données pour certains éléments visuels des pages JSF (tableaux, listes déroulantes, etc…). Elles gèrent également les actions effectuées par l'utilisateur sur les pages (clics sur des boutons ou des liens tout particulièrement).

- Fonctions JavaScript et appels Ajax pour l'interactivité côté client. Certains scripts ajoutent des effets spéciaux visuels à la page (coins arrondis, animations, etc..), d'autres envoient des requêtes Ajax pour obtenir des données de la part du serveur.

Les Servlets sont également une partie importante de l'application. Ces classes Java sont spécialement conçues pour traiter et répondre à des requêtes HTTP. Elles sont donc utilisées en premier lieu en conjonction avec des requêtes Ajax, qui ne sont rien d'autre que des requêtes HTTP envoyées dynamiquement à partir du client.

3.5 Pages Web de l'application

Cette section passe en revue les différentes pages de l'application et présente leur fonctionnement en termes de « front-end » (interaction avec l'utilisateur, couche présentation) et de « back-end » (intelligence, traitement, couche métier).

Les pages sont réparties en trois catégories : indexation, recherche et visualisation. Les pages de la dernière catégorie font l'objet de leur propre section (p. 54).

3.5.1 Indexation du partage réseau

Cette page est le portail donnant accès au processus global décrit dans la section 3.3 p. 23. La Figure 22 p. 44 est une capture d'écran de la page.

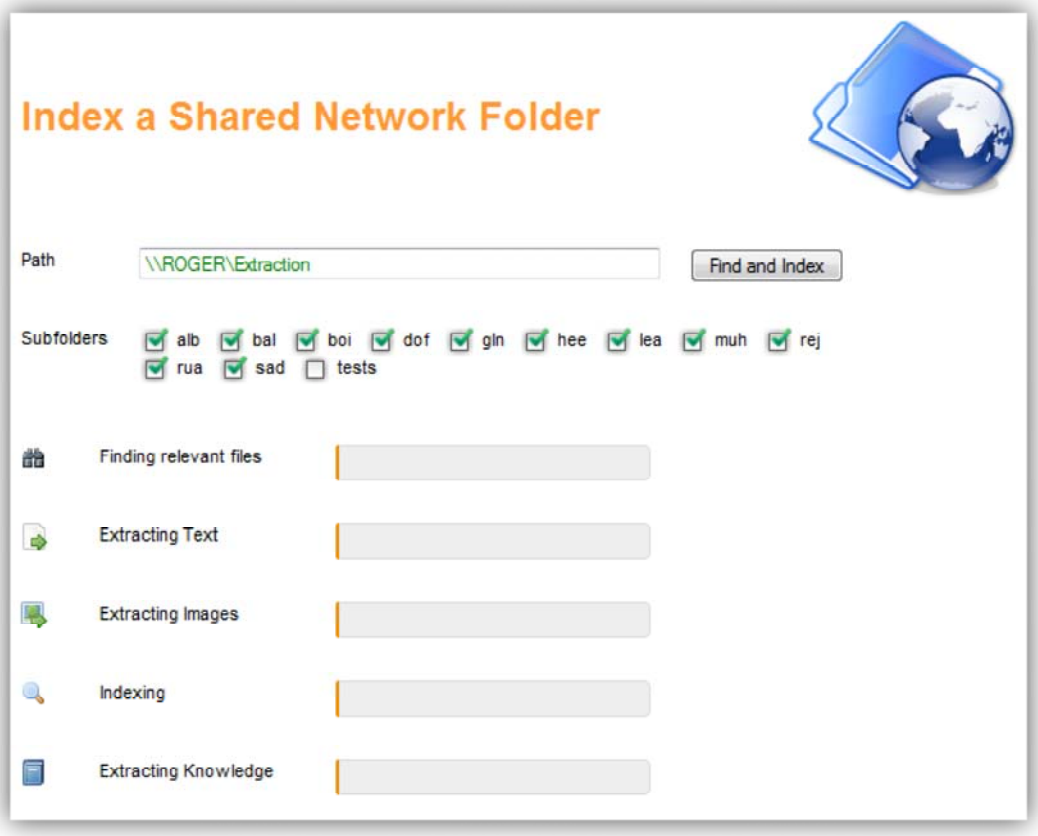


Figure 22 - Page d'indexation du partage réseau

La page est composée des éléments suivants :

- Zone d'entrée du chemin du partage réseau
- Bouton exécutant le processus global
- Zone de sélection des sous-répertoires
- Barres de progression pour les différentes étapes du processus

3.5.1.1 Zone d'entrée du chemin du partage réseau

Cette zone a un aspect dynamique utile : durant la saisie du chemin, un Servlet va être interrogé par le biais d'une requête Ajax afin de déterminer si le chemin entré existe réellement ou non. Si le chemin n'existe pas, le texte de la zone sera coloré en rouge, la zone des sous-répertoires sera vidée et le bouton d'exécution du processus sera désactivé (voir Figure 23 p. 45).

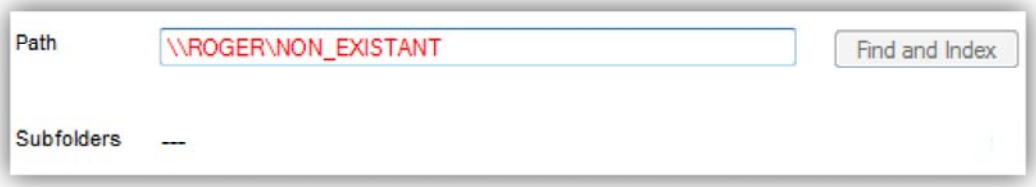


Figure 23 - Entrée d'un chemin réseau non existant dans la page d'indexation

3.5.1.2 Zone de sélection des sous-répertoires

Dans le cas où le chemin indiqué dans la zone de texte est valide, le Servlet de contrôle renvoie une liste des sous-répertoires contenus dans le partage. L'utilisateur a ensuite le choix de définir quels répertoires inclure dans le processus global.

3.5.1.3 Barres de progression dynamiques

Une fois que l'utilisateur a démarré le processus en cliquant sur le bouton correspondant, des requêtes Ajax sont envoyées à intervalles réguliers afin de s'informer sur le progrès de l'étape en cours. C'est encore une fois à l'aide d'un Servlet que cette information est récupérée. Le Servlet reçoit les requêtes et interroge la classe Java utilisée dans l'étape courante.

En effet, toutes les classes faisant partie du processus global contiennent une variable « progress » mise à jour durant l'exécution du processus. Le degré de détail du progrès dépend de l'étape. Voici un résumé des informations disponibles pour chaque étape :

- **Identification des fichiers à traiter** : La méthode d'exploration étant récursive, le nombre total de répertoires à explorer n'est pas connu d'avance. Le progrès est donc simplement mesuré par rapport aux répertoires de premier niveau qui ont été explorés.
- **Extraction du texte et des images** : Ces deux étapes basent la mesure de leur progrès sur la liste des fichiers renvoyée par la première étape.
- **Indexation** : Le processus d'indexation dispose de la liste d'informations à indexer renvoyée par l'étape d'extraction du texte. A l'aide de cette liste, il est possible de mesurer de manière précise le progrès de cette étape.

- **Extraction de la connaissance** : C'est l'étape la plus difficile à évaluer. Le processus d'extraction en lui-même est effectué par une classe externe et il n'a donc pas été possible d'y intégrer un concept de mesure du progrès. Le progrès est donc simplement mesuré par rapport au nombre de répertoires à traiter (l'extraction se fait de manière séparée pour chaque auteur).

Du côté interface utilisateur, le composant « ProgressBar » de la bibliothèque jQuery UI[59] a été utilisé. Cette bibliothèque, qui s'intègre parfaitement à jQuery, fournit une multitude de composants visuels riches et simples à utiliser (onglets, calendrier, fenêtres de dialogue, etc…), ainsi que des effets visuels.

3.5.2 Recherche de documents

La recherche de documents est une partie importante de l'application. Cette page interagit avec de nombreux éléments dont l'index Lucene et la base de données SQLite. La Figure 24 p. 46 est une capture d'écran de la page de recherche avant l'affichage des résultats.

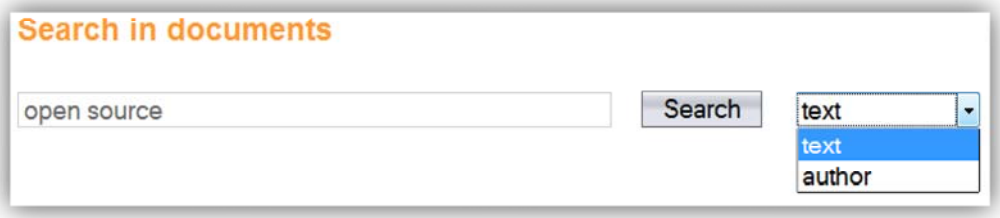


Figure 24 - Page de recherche (avant résultats)

3.5.2.1 Structure de la page (recherche)

La page est composée des éléments suivants :

- Zone de recherche avec auto-complétion
- Bouton pour exécuter la recherche
- Liste pour sélectionner le mode de recherche

Avant d'entrer dans le détail de chaque élément, il est important de noter que cette page est liée à une classe nommée `TextSearchController` qui gère tous ses éléments. Ceci est un peu différent du reste de l'application, où une classe Java spécifique est créée pour chaque page. Dans ce cas, la classe `TextSearchController` est utilisée par plusieurs pages différentes qui nécessitent certaines des mêmes fonctionnalités.

La zone de recherche a déjà été détaillée dans la section 3.2.5.1 en page 19. Le bouton de recherche est lié à une action définie dans le contrôleur `TextSearchController`. La liste déroulante permet de choisir entre une recherche du contenu des documents ou une recherche par auteur.

[59] Téléchargement : http://jqueryui.com / Licence : voir jQuery (p. 10)

3.5.2.2 Fonctionnement de la recherche

Une fois une requête entrée et le bouton de recherche cliqué, le traitement est délégué au contrôleur `TextSearchController`. Ce dernier utilise un objet `IndexSearcher` afin de récupérer les résultats de la recherche dans l'index Lucene. L'Extrait de Code 11 p. 47 montre les deux instructions de base de la recherche (pour les documents en anglais).

```
//ENGLISH DOCUMENTS
documents = performSearchWithParser(
    new QueryParser(Version.LUCENE_30,
    field,
    new StandardAnalyzer(Version.LUCENE_30)
    ),
    isearcher);
insertDocumentsIntoGlobalCollection(
    documents,
    mergedDocuments,
    mergedDocumentAddresses
);
```

Extrait de Code 11 - Recherche de documents (en anglais)

La méthode `performSearchWithParser` contient le code détaillé de la recherche. Elle est détaillée ci-après (p. 48). Elle prend en paramètre un objet `QueryParser` et l'objet `IndexSearcher` qui accède à l'index Lucene.

Le `QueryParser` interprète la chaîne de caractères saisie par l'utilisateur pour la transformer en requête compréhensible par Lucene. Ce `QueryParser` prend deux paramètres : le champ du document à analyser (contenu ou auteur selon le choix fait par l'utilisateur) et un analyseur, capital pour la recherche dans un index contenant des documents dans différentes langues.

Recherche de documents dans plusieurs langues

Un document ne peut dans certains cas être retrouvé qu'à l'aide de l'analyseur utilisé durant son indexation. Ceci est dû en majeure partie au « stemming », qui raccourcit les mots pour en conserver uniquement la racine.

C'est pour cette raison que la recherche est effectuée à trois reprises pour obtenir les documents dans toutes les langues (anglais, français et allemand). La méthode `insertDocumentsIntoGlobalCollection` permet de gérer les doublons dans le cas ou le même document serait trouvé plusieurs fois avec les différents analyseurs.

Méthode performSearchWithParser

Cette méthode effectue la recherche dans l'index et récupère les résultats (voir Extrait de Code 12 p. 48).

```
searchQuery = parser.parse(getSearchString());

TopScoreDocCollector collector =
TopScoreDocCollector.create(Config.MAX_HITS, false);

isearcher.search(searchQuery, collector);
ScoreDoc[] hits = collector.topDocs().scoreDocs;

for (int i = 0; i < hits.length; i++) {
    int docId = hits[i].doc;
    documents.add(isearcher.doc(docId));
}

return documents;
```

Extrait de Code 12 - Recherche dans l'index de Lucene

Voici le détail de cette méthode :

- À l'aide de l'objet QueryParser passé en paramètre, la chaîne de recherche (récupérée par `getSearchString()`) est transformée en objet `Query`.
- Un collecteur de résultats `TopScoreDocCollector` est créé. Celui-ci permet de récupérer les meilleurs résultats de la recherche (par tri décroissant du score attribué par l'`IndexSearcher`). Il est possible de lui indiquer un nombre maximal de résultats à récupérer. [46]
- La méthode `search` effectue la recherche à proprement dire (dans l'index Lucene).
- Un tableau des meilleurs résultats est récupéré à partir du collecteur de résultats.
- Chaque entrée de ce tableau est ajoutée à la liste des documents qui seront affichés dans la page Web.

3.5.2.3 Structure de la page (résultats)

Une fois la liste des résultats à afficher établie, il est temps de les afficher dans la page Web. Le composant JSF `<t:dataTable>` est parfait dans ce genre de situations, puisqu'il permet de lier une collection de données dans le code Java à un tableau HTML dans la page Web. Il suffit ensuite de décrire la structure d'un seul résultat (nombre de colonnes, contenu, style, etc...) et toutes les lignes du tableau seront automatiquement générées. La Figure 25 p. 49 donne un aperçu des résultats d'une recherche.

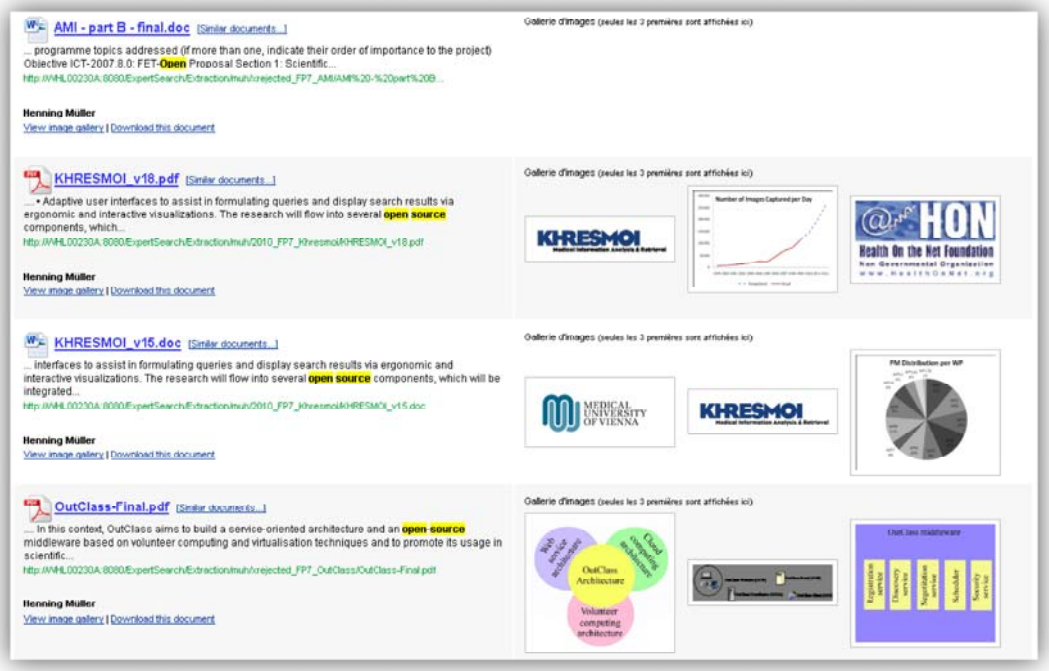


Figure 25 - Résultats de recherche

Notons que la différenciation de couleurs entre les lignes paires et impaires est également une fonctionnalité du composant `<t:dataTable>`.

Les composants d'un résultat de recherche sont les suivants :

- Icône correspondant au type de document
- Nom du fichier du document avec un lien vers celui-ci
- Lien de recherche de documents similaires
- Extrait du document avec surlignage des termes cherchés
- Adresse du fichier
- Auteur du document
- Lien vers la galerie d'images
- Lien de téléchargement du document
- Extrait de la galerie des images (trois images au maximum)

Le seul composant complexe qui n'a pas encore été abordé est le lien de recherche de documents similaires, qui est détaillé dans la section suivante.

3.5.2.4 Recherche de documents similaires

Une fonctionnalité de recherche de documents similaires avait déjà été implémentée dans le travail de M. Eggel. Cette dernière effectuait simplement une nouvelle recherche sur la base de l'extrait du document.

Depuis, une classe avec des fonctionnalités avancées spécialement conçue pour la recherche de documents similaires a été développée par la communauté de Lucene. [47,48] Cette classe se nomme `MoreLikeThis` (en français : « plus de choses comme celle-ci ») et l'Extrait de Code 13 p. 50 démontre son utilisation.

```
URL similarURL = new URL(similarTextFilename);

MoreLikeThis mlt = new MoreLikeThis(indexReader);
mlt.setFieldNames(new String[] {"content"});
mlt.setMinWordLen(4);
mlt.setMinDocFreq(2);
mlt.setMaxDocFreq(70);

Query query = mlt.like(similarURL);
```

Extrait de Code 13 - Recherche de documents similaires

Voici une description des étapes :

- Un objet `URL` est créé sur la base du chemin du fichier pour lequel des documents similaires doivent être trouvés.
- Un objet `MoreLikeThis` est créé (l'`IndexReader` lui est passé en paramètre car il nécessite un accès en lecture sur l'index Lucene)
- Les propriétés de l'objet sont définies. La première opération `setFieldNames` définit dans quels champs du document la comparaison doit être faite (dans notre cas, la similarité est comparée au niveau du contenu du document).

  La deuxième opération définit la longueur minimale d'un mot pour qu'il se qualifie pour la comparaison (quatre caractères dans notre cas, ce qui élimine un bon nombre de « stop words » et d'autres mots sans valeur informative).

  Les deux dernières opérations sont semblables à des paramètres définis dans Mahout (voir section 3.3.5.3, partie «Extraction des collocations» en page 39).
- Un objet `Query` (requête) est créé grâce à la méthode `like` de l'objet `MoreLikeThis` (l'URL du fichier de base est passée en paramètre).

Le reste de la recherche se déroule comme la méthode `performSearchWithParser` expliquée en page 48.

3.5.3 Recherche de mots-clés

Le projet étant axé sur la recherche d'experts, cette page a été créée afin de faciliter la recherche de personnes possédant des connaissances par rapport à un terme donné. L'interface de recherche est très simpliste comme le montre la Figure 26 p. 50.

Figure 26 - Recherche de mots-clés

Des requêtes Ajax qui vont interroger un Servlet sont envoyées à chaque frappe de touche de l'utilisateur (à partir de 2 caractères pour des raisons de performance).

Le Servlet va lire le fichier contenant la liste des professeurs pour chaque mot-clé, enregistré durant la phase d'extraction de connaissances, et renvoyer tous les mots-clés commençant par le texte entré par l'utilisateur avec les professeurs associés (y compris la pondération du lien).

En temps réel, les résultats sont alors affichés en-dessous de la zone de recherche. La Figure 27 p. 51 est une capture d'écran d'une recherche du terme « information ».

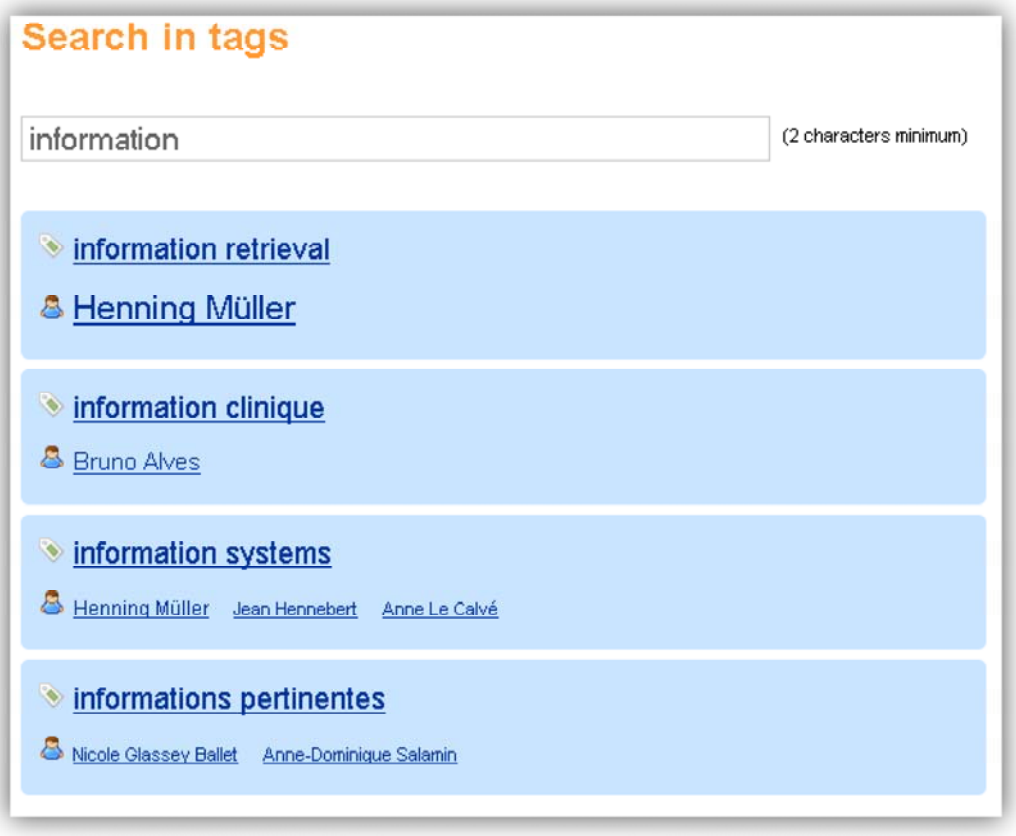


Figure 27 - Résultats d'une recherche de mots-clés

3.5.4 Profil d'auteur

Trois moyens permettent d'accéder à la page personnelle d'un auteur. Le premier est de cliquer sur le nom de l'auteur dans le résultat de recherche. Le deuxième est la page d'auteurs contenant simplement une liste des auteurs disponibles (la Figure 28 p. 52 est une capture d'écran de cette page) et le troisième est un clic sur le nom de l'auteur dans la visualisation MooWheel (voir section 3.6.5 p. 61).

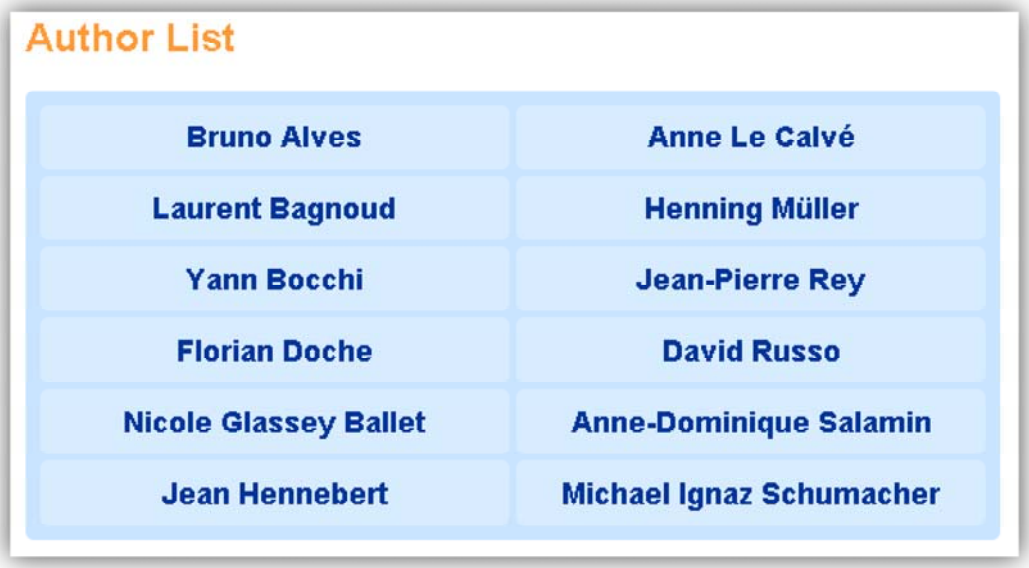


Figure 28 - Page de liste d'auteurs

Après avoir cliqué sur le nom d'un auteur, une page de profil contenant les informations suivantes va s'afficher :

- Tag Cloud des termes associés à l'auteur
- Tag Cloud des liens avec d'autres auteurs
- Liste des documents rédigés par l'auteur

La page du profil individuel est illustrée par la Figure 29 p. 52.

Figure 29 - Page de profil d'auteur

La page utilise le même contrôleur `TextSearchController` que la recherche « classique » pour récupérer la liste des documents de l'auteur. Seule la requête transmise est différente : le nom de l'auteur entouré de guillemets (") est envoyé et ce dernier est transformé en `PhraseQuery` par le `QueryParser` (ce type de requête permet de chercher une expression exacte).

3.5.5 Profil d'auteur - mode administrateur

Le profil d'auteur a également un mode administrateur permettant la suppression de mots-clés non désirés. L'auteur doit tout d'abord se connecter avec un nom d'utilisateur et un mot de passe (partagé par tous les auteurs pour simplifier la tâche) sur la page illustrée par la Figure 30 p. 53.

Figure 30 - Page de login d'auteur

Une fois connecté, le profil d'auteur est légèrement différent : des liens de suppression s'affichent à côté de chaque terme (voir Figure 31 p. 53). Une petite réserve de termes est prévue, c'est-à-dire que 50 termes sont conservés mais uniquement 30 sont affichés dans le Tag Cloud. Les mots-clés supprimés par l'auteur seront donc remplacés par ceux contenus dans la réserve si elle n'a pas été épuisée.

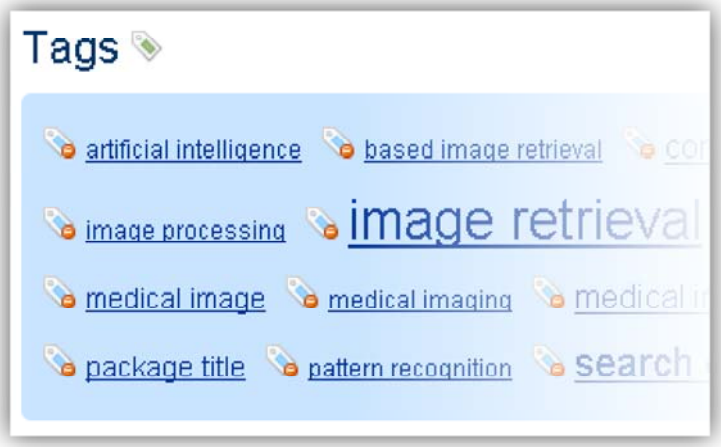


Figure 31 - Liens de suppression de mots-clés en mode administrateur

Les mots-clés supprimés sont conservés dans une liste individuelle pour chaque auteur et ne seront plus pris en compte lors d'indexations suivantes. Cette fonctionnalité est assez importante compte tenu du fait que les mots-clés extraits de manière automatisée ne seront probablement jamais pertinents à 100%.

3.5.6 Page de mot-clé

Tout comme pour les auteurs, il peut être intéressant de découvrir des détails sur un mot-clé donné. C'est pourquoi deux moyens sont fournis pour accéder à une page de

détail d'un mot-clé : un clic sur le mot-clé dans le Tag Cloud du profil d'un auteur ou un clic sur le mot-clé dans la visualisation MooWheel (voir section 3.6.5 p. 61).

La Figure 32 p. 54 est une capture d'écran de la page du mot-clé « open source ».

Figure 32 - Page de détail d'un mot-clé

La page contient les éléments suivants :

- Tag Cloud des auteurs liés à ce mot-clé
- Liste des documents contenant ce mot-clé

Les auteurs pour un mot-clé donné peuvent être récupérés très facilement grâce au fichier d'auteurs par mot-clé enregistré durant la phase d'extraction de connaissances.

3.6 Visualisation des données

La visualisation des données est un aspect important de la gestion des connaissances mais elle est difficile à mettre en place. La visualisation nécessite un mélange du domaine créatif (côté visuel) et du domaine scientifique (côté données), deux mondes qui ne sont pas toujours faciles à réconcilier. [49] Il s'agit de consolider la masse d'informations disponible dans un composant graphique clair, compréhensible, utilisable et esthétiquement agréable.

Il existe plusieurs moyens établis pour visualiser des grandes quantités de données ou des données liées entre elles, comme par exemple les Tag Clouds (ou « nuage de mots-clés »), qui sont très populaires depuis l'apparition du Web 2.0 et des sites collaboratifs. [50]

Plusieurs librairies ont été implémentées durant ce travail pour trouver des moyens de visualisation intéressants. Les composants testés incluent :

- Tag Clouds animés en trois dimensions (une solution Javascript et une en Adobe Flash)
- MindMap (ou carte heuristique)
- Graphique en cercles concentriques (Radius Graph)
- Graphique expérimental en forme de roue nommé « MooWheel »

Notons au passage que toutes les visualisations ont été testées uniquement avec le navigateur Firefox 3.5. Internet Explorer et des versions plus anciennes de Firefox n'étaient pas capables d'afficher la majorité des visualisations.

3.6.1 Tag Cloud animé (Javascript)

Ce Tag Cloud[60] (le premier essai de création d'une visualisation) utilise Javascript et l'élément HTML5 <canvas>, qui permet de dessiner des formes ou du texte dans une zone et d'effectuer des transformations graphiques telles que des translations ou des rotations.

Au niveau des données, ce premier exemple est simplement une collection de mots-clés fournis par les différents membres de l'institut de recherche. Cette expérience avait pour but de découvrir s'il était possible de créer un Tag Cloud avec un grand nombre de termes qui reste utilisable. La Figure 33 p. 55 est une capture d'écran de ce Tag Cloud.

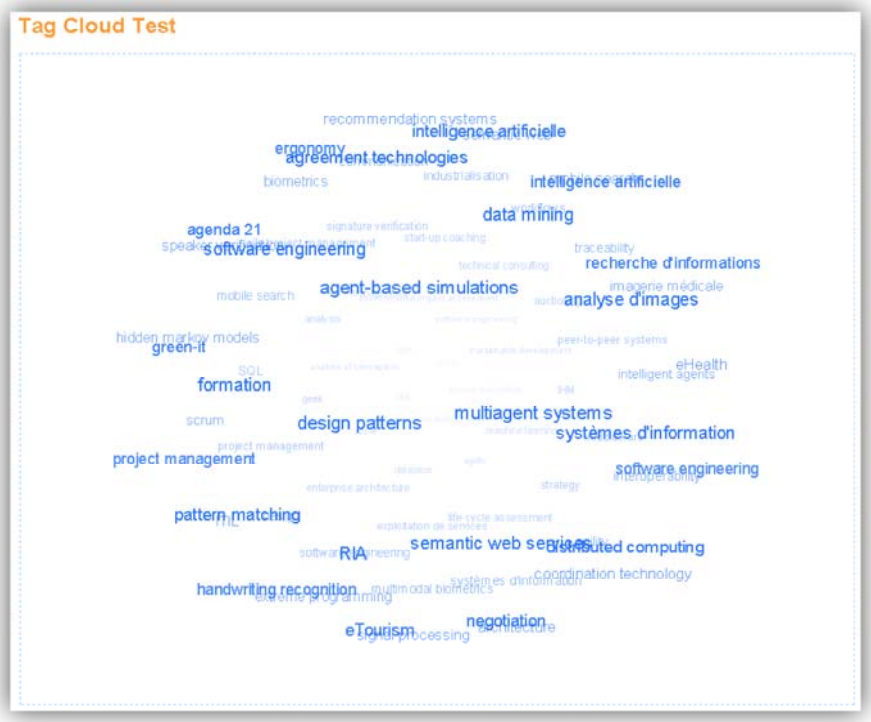


Figure 33 - Tag Cloud animé (Javascript)

[60] Téléchargement : http://www.goat1000.com/tagcanvas.php / License : LGPLv3

Le Tag Cloud est organisé dans une sphère rotative donnant l'illusion d'être en trois dimensions. L'utilisateur peut la faire tourner à l'aide de la souris. Un clic sur un terme redirige l'utilisateur vers la page de mot-clé en fournissant le texte du mot-clé comme paramètre.

Le résultat est le suivant : le nombre de termes et leur uniformité graphique résulte en une certaine surcharge visuelle et rend difficile la recherche d'un terme précis. Par contre, il peut être intéressant de simplement observer la sphère qui tourne pour tenter de repérer un terme intéressant.

3.6.2 Tag Cloud animé (Flash)

Après la réalisation du premier Tag Cloud, mon professeur responsable, M. Müller, m'a indiqué une deuxième solution semblable, mais possédant des fonctionnalités plus avancées. Il s'agit de WP-Cumulus[61] qui était initialement un plugin pour Wordpress, un système de blogs très populaire. Le créateur du plugin, Roy Tanck, a publié par la suite des instructions expliquant comment utiliser WP-Cumulus en dehors de Wordpress.[62]

La Figure 34 p. 56 est une capture d'écran de ce Tag Cloud.

Figure 34 - Tag Cloud animé (Flash)

[61] Téléchargement : http://wordpress.org/extend/plugins/wp-cumulus/download / Licence : GPL
[62] Instructions : http://www.roytanck.com/2008/05/19/how-to-repurpose-my-tag-cloud-flash-movie/

Il devient rapidement apparent que cette solution est supérieure à la première. Chaque mot-clé peut être personnalisé en termes de taille du texte et de couleur. De plus, une différenciation automatique de couleur est faite entre les termes de grande taille (grande importance) et les termes d'une plus petite taille (importance moindre). L'animation est également beaucoup plus fluide, ce qui provient du fait que Flash est conçu spécialement pour le développement d'animations complexes, alors que Javascript et HTML ont d'autres objectifs.

Au niveau des données, ce Tag Cloud recréé la structure du site de l'institut[63] dans le but éventuel d'être intégré à la page d'accueil de ce site afin de la rendre plus ludique. Un clic sur un des mots-clés redirige alors simplement vers la page correspondante du site de l'institut.

3.6.3 MindMap

Les deux Tag Clouds décrits précédemment permettent de visualiser un seul niveau de données. Il est donc impossible d'afficher les liens des auteurs avec les mots-clés qui leurs sont associés.

C'est pourquoi d'autres visualisations ont été recherchées, permettant d'avoir une vue d'ensemble de tout l'institut (auteurs et connaissances / mot-clés).

Un MindMap semble être idéal pour l'affichage de liens entres des concepts. Un composant nommé simplement « js-mindmap » a rapidement été trouvé. Cette solution[64], initialement écrite pour une autre librairie Javascript (MooTools) a par la suite été convertie en plugin jQuery[65].

La Figure 35 p. 57 est une capture d'écran du MindMap.

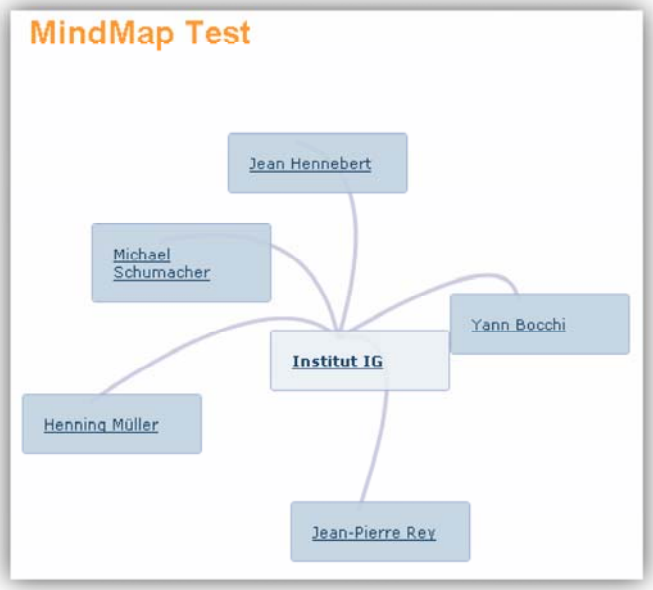


Figure 35 - MindMap (premier niveau)

[63] Site : http://iig.hevs.ch/
[64] Démo et téléchargement : http://kenneth.kufluk.com/google/js-mindmap/
[65] Téléchargement : http://code.google.com/p/js-mindmap / Licence : MIT

Comme il est indiqué dans la légende de la Figure 35, le MindMap possède plusieurs niveaux. Pour accéder à un autre niveau, il suffit de cliquer sur une des cases du MindMap (dans notre cas, un des membres de l'institut de recherche). Les mots-clés associés à cette personne sont alors affichés. De plus, un lien d'une couleur spéciale permet de retourner au niveau supérieur. La Figure 36 p. 58 illustre cela.

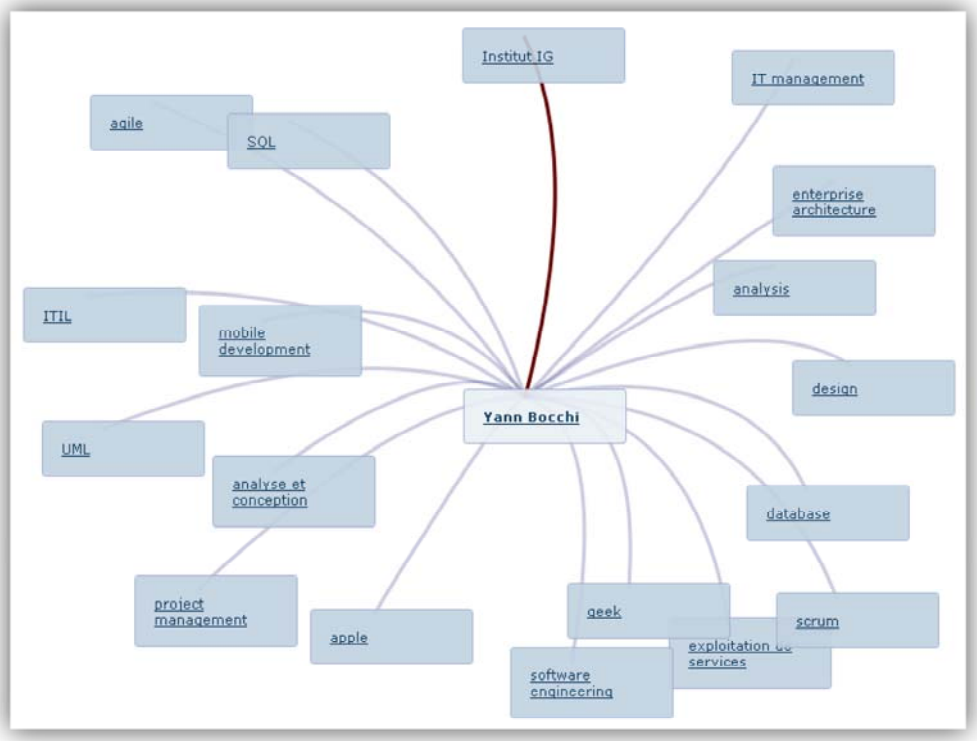


Figure 36 - MindMap (deuxième niveau)

Au niveau des données, cette visualisation utilise la même collection de mots-clés que le Tag Cloud Javascript. L'utilisateur peut double-cliquer sur une case pour accéder soit au profil de l'auteur, à la page du mot-clé ou au site de l'institut de recherche (selon la case cliquée).

Encore une fois, un nombre trop important de mots-clés peut nuire à l'ergonomie de la visualisation. Un filtre par importance du mot-clé pourrait être mis en oeuvre afin d'obtenir un affichage optimal.

3.6.4 Radius Graph

Le MindMap a été sélectionné pour ses fonctionnalités dynamiques telles que le drag&drop (ou cliquer-déplacer) des éléments ou les réactions d'attraction et de répulsion entre les cases.

Une visualisation plus sobre mais plus claire a été développée à l'aide de la librairie de visualisation JIT[66] [67], qui fournit une multitude de visualisations innovatrices adaptées à de nombreux types de données.

La visualisation JIT sélectionnée se nomme « RGraph »[68] (ou « Radius Graph »).Elle répartit des informations de plusieurs niveaux sur des cercles concentriques (voir la Figure 37 p. 59, qui est une capture d'écran de l'implémentation de RGraph).

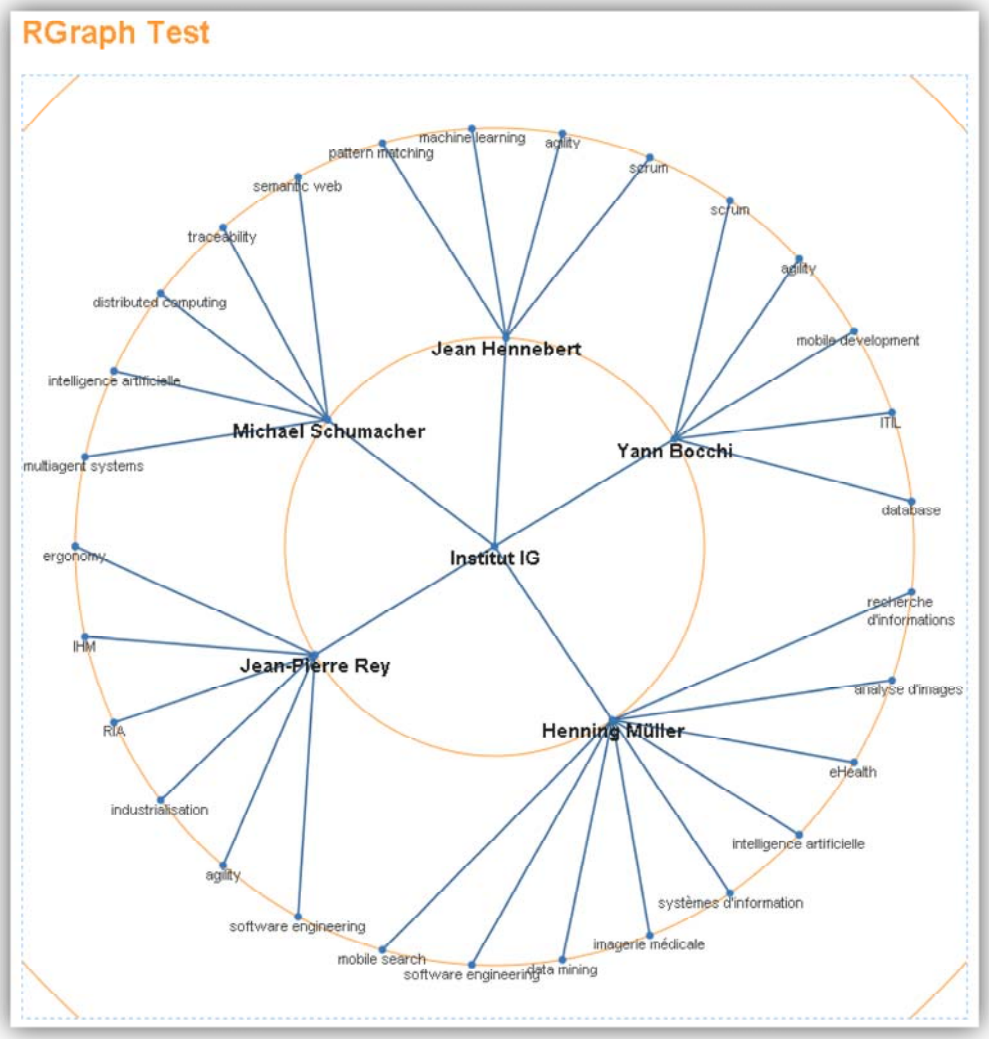


Figure 37 - RGraph (premier niveau)

[66] JavaScript InfoVis Toolkit
[67] Téléchargement : http://thejit.org / Licence : BSD
[68] Démonstrations : http://thejit.org/demos/

L'utilisateur a la possibilité de cliquer sur les différents nœuds du graphique afin de recentrer la visualisation dynamiquement sur l'élément sélectionné (comme démontré par la Figure 38 p. 60). Il a également la possibilité d'effectuer des zooms avec la roulette de la souris ou de se déplacer dans la visualisation en faisant un cliquer-glisser.

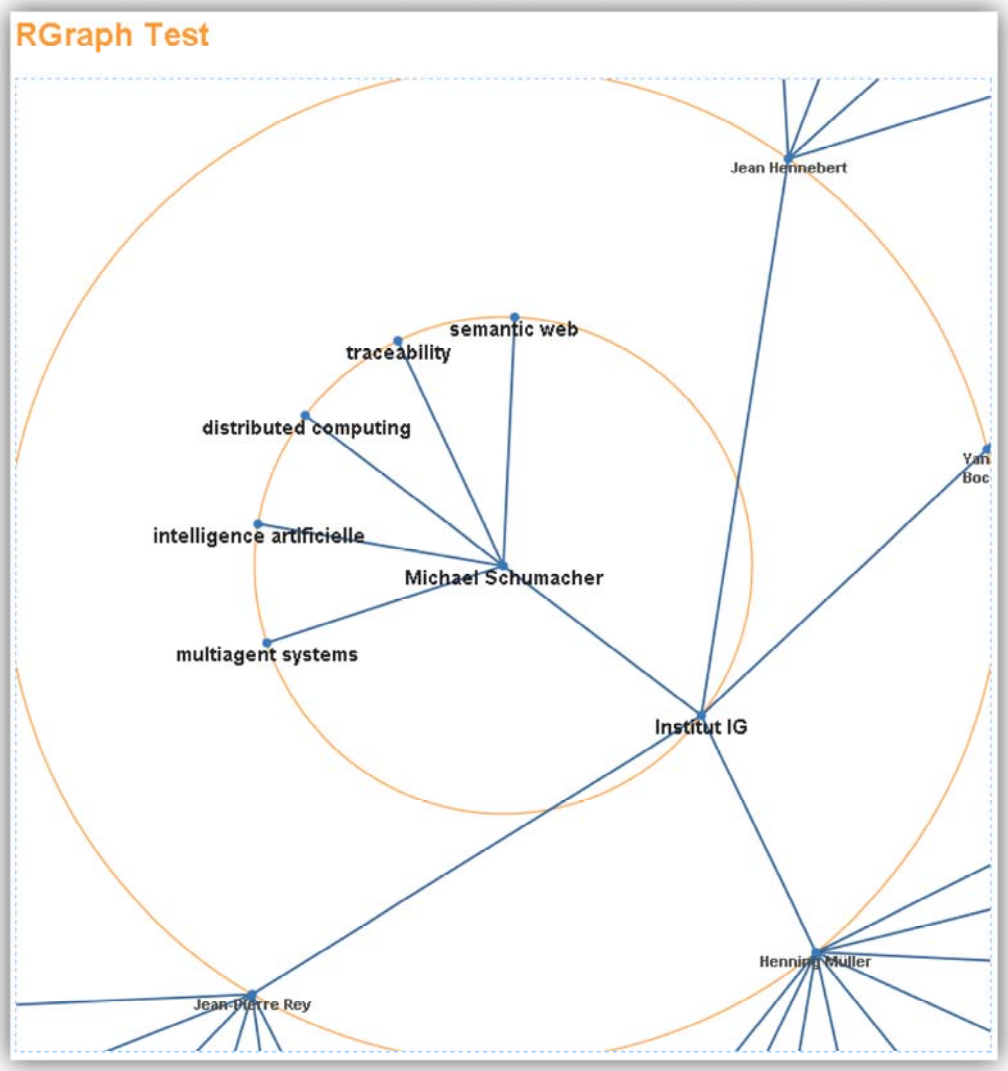


Figure 38 - RGraphe (deuxième niveau)

Les données représentées sont les mêmes que pour le MindMap, uniquement la disposition des éléments est différente. Cette visualisation ne fournit pas de fonctionnalité de liens hypertextes : il n'est donc pas possible de rediriger l'utilisateur vers une autre page. Le RGraph fournit simplement une perspective intéressante sur une collection de données.

3.6.5 MooWheel

Cette visualisation, découverte en dernier lieu, est la plus complète. Elle permet d'afficher des éléments et de les interconnecter à l'aide de liens pondérés. La librairie utilisée se nomme « MooWheel[69] » car elle utilise la librairie Javascript MooTools et la visualisation peut rappeler la forme d'une roue.

La Figure 39 p. 61 est une capture d'écran de la visualisation.

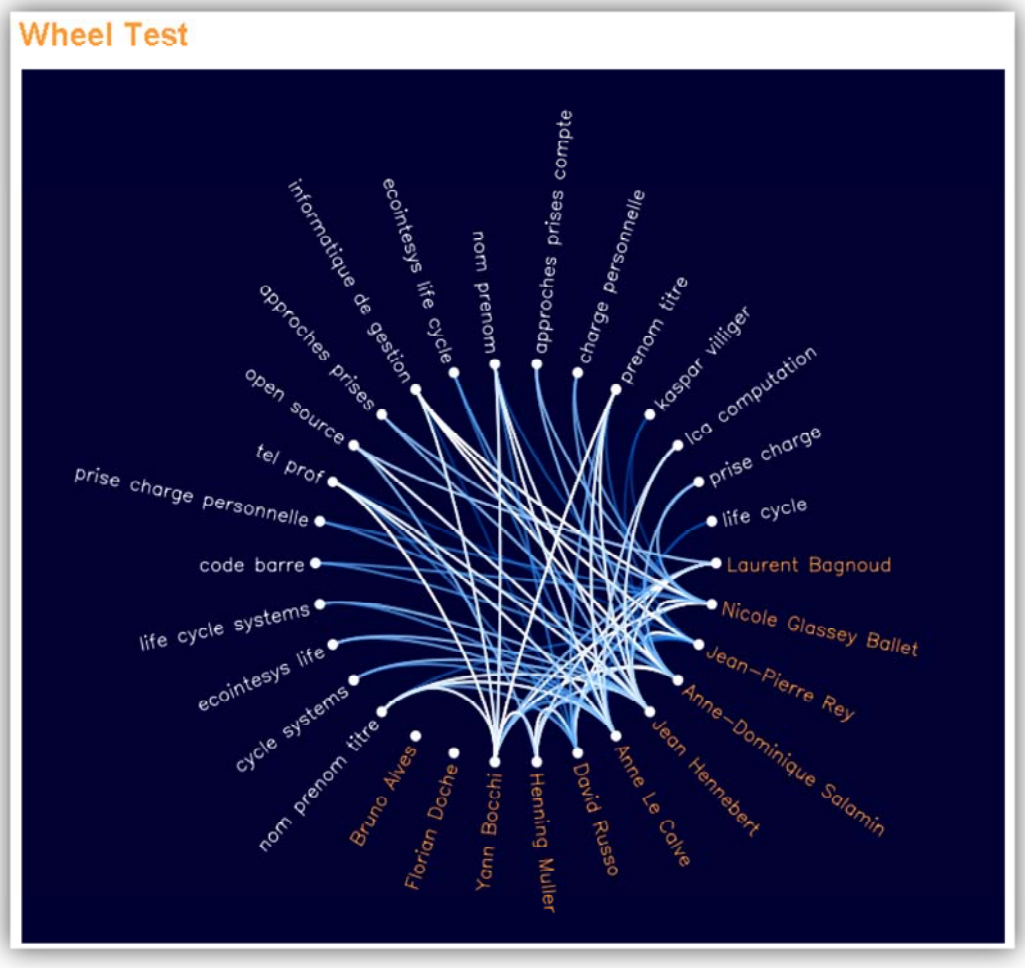


Figure 39 - MooWheel (affichage de base)

La pondération affecte la couleur du lien (plus le lien est fort, plus la couleur devient bleue et sombre). De plus, une modification a été apportée au code source de la librairie afin de distinguer les auteurs de documents (en orange) des mots-clés (en blanc).

Les liens sont évidemment trop nombreux pour distinguer directement toutes les connexions entrantes ou sortantes d'un nœud. C'est pourquoi l'utilisateur peut simplement pointer le curseur sur l'élément qui l'intéresse pour mettre en évidence uniquement les liens de ce dernier (voir l'exemple de la Figure 40 p. 62).

[69] Téléchargement & Informations : http://labs.unwieldy.net/moowheel/

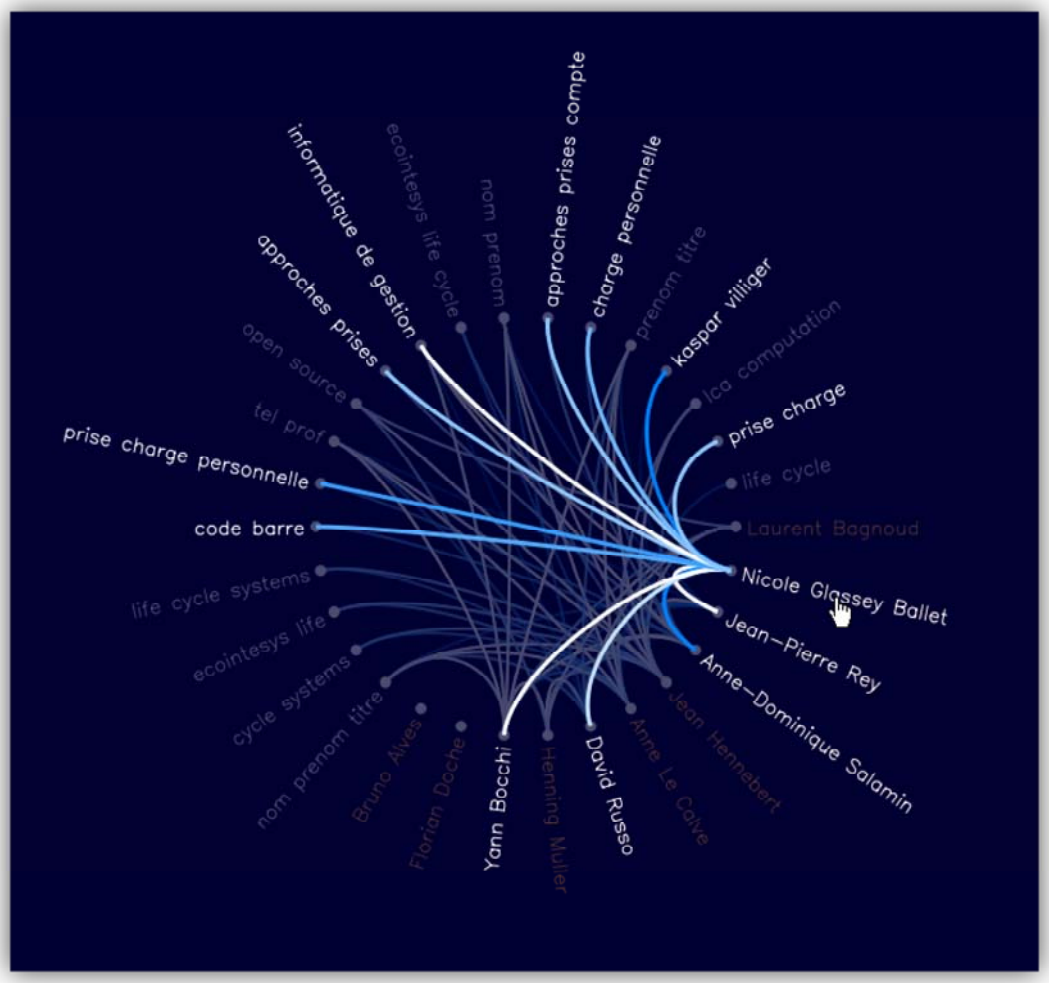


Figure 40 - MooWheel (élément sélectionné)

Les liens deviennent alors bien plus flagrants. Ceci est dû en partie à une autre modification apportée à la source de la librairie. D'origine, seul le texte de l'élément sur lequel se trouve le curseur était illuminé, ainsi que les liens. Malheureusement le texte des éléments liés n'était pas mis en évidence, rendant difficile la lecture de ces derniers.

Notons que pour diminuer le nombre d'éléments dans le graphique, seuls les mots-clés partagés par plusieurs personnes sont affichés, avec un maximum de sept mots-clés par personne.

Cette visualisation étant la plus aboutie, c'est également la seule qui est créée à partir de données générées dynamiquement durant le processus d'extraction de connaissances. Les données sont stockées au format JSON et contiennent tous les éléments nécessaires à la génération des nœuds et des liens, y compris le type de nœud pour la différenciation de couleur.

3.7 Statistiques de la métadonnée « auteur »

Tous les fichiers stockés sur un ordinateur possèdent en principe un attribut « auteur » permettant de définir le créateur d'un document. Cette métadonnée est parfois complétée par le logiciel utilisé lors de la création du document et se base la plupart du temps sur le nom d'utilisateur connecté sur l'ordinateur.

Le professeur responsable, M. Müller, se doutait dès le départ du travail que cette métadonnée ne pourra certainement pas être stockée dans l'index Lucene à cause de son taux de fiabilité présumé être très bas.

M. Müller souhaitait donc obtenir des statistiques plus précises par rapport à la fiabilité de l'attribut, en se basant sur ses documents disponibles sur le partage réseau de l'institut de recherche. La Figure 41 p. 63 est un graphique résumant les différentes catégories de données trouvées dans l'attribut « auteur ».

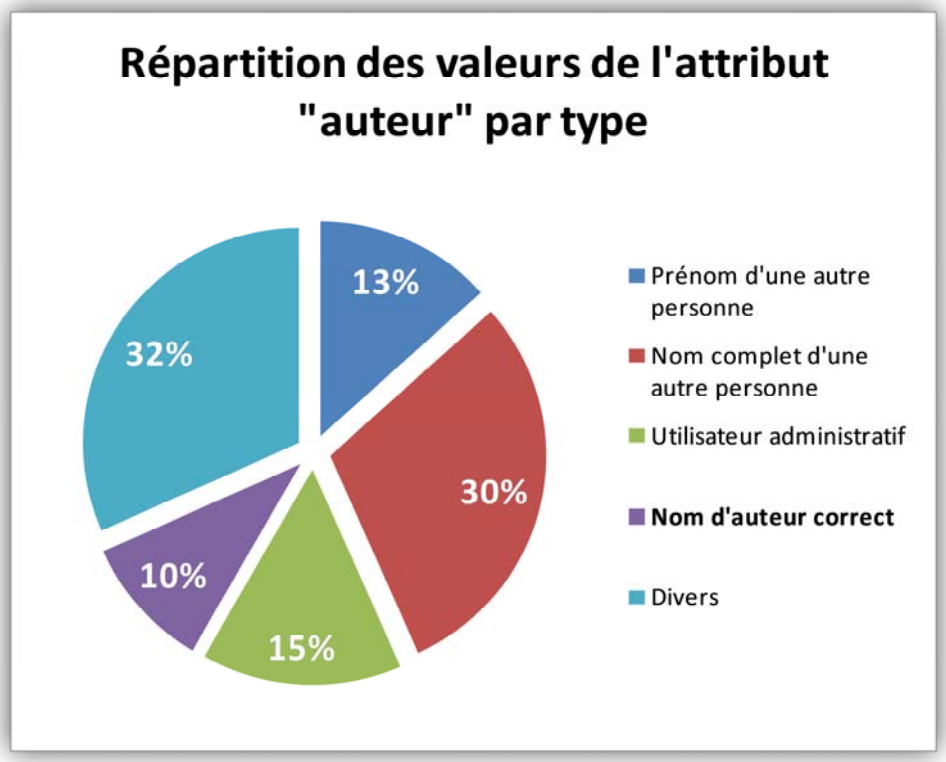


Figure 41 - Répartition des valeurs de la métadonnée "auteur" par type

Le résultat est clair : seuls 10% des documents ont un attribut « auteur » en rapport avec M. Müller (nom complet ou nom d'utilisateur). Les autres valeurs sont composées de chaînes vides, de valeurs inintelligibles telles que « u6774 » ou « xxx », de noms de personnes autres que M. Müller ainsi que de noms d'utilisateurs administratifs tels que « Service Informatique » ou « Administrateur ».

Cette statistique très révélatrice explique l'écrasement de la métadonnée par la valeur déterminée à l'aide du répertoire dans lequel se trouve le document (exemple : le répertoire « muh » correspond à M. Henning Müller, tous les documents s'y trouvant lui seront donc attribués).

4 Conclusions

Cette section présente les réflexions et observations faites sur le travail réalisé, d'un point de vue technique ainsi que personnel. Le déroulement du projet est également abordé ici.

4.1 Bilan technique du projet

Plusieurs critères du projet peuvent être analysés pour déterminer son utilité et sa fiabilité :

- Qualité des mots-clés extraits
- Performances
- Ergonomie de l'application Web

Cette section se concentre sur les deux points les plus simples à mesurer : la qualité et la performance, avec un accent plus particulier sur la qualité.

4.1.1 Qualité des mots-clés extraits

Deux critères sont définis pour évaluer la qualité des mots-clés extraits :

- La correspondance entre des mots-clés fournis par les divers auteurs et la liste des termes extraits de manière automatique.
- Le pourcentage de termes « utilisables » parmi les 50 mots-clés extraits (utilisable signifiant qu'il s'agit d'un terme intelligible, contrairement aux termes qui peuvent être considérés comme étant du « bruit »).

4.1.1.1 Correspondance mots-clés fournis / mots-clés extraits

Il est intéressant de comparer la liste de mots-clés fournis par M. Müller à celle qui est générée par le processus d'extraction des connaissances. Le Tableau 5 p. 65 présente les mots-clés fournis par M. Müller à gauche et les correspondances établies dans les 50 termes extraits automatiquement. Les correspondances sont codifiées à l'aide de couleurs de la manière suivante :

- Vert = Excellente correspondance
- Jaune = Correspondance indirecte (même thème, même domaine)
- Rouge = Pas de correspondance

Mots-clés fournis	Correspondance dans les termes extraits
recherche d'informations	information retrieval
analyse d'images	image retrieval
eHealth	medical informatics
intelligence artificielle	artificial intelligence
systèmes d'information	information systems
imagerie médicale	medical imaging / medical image
data mining	data sets
software engineering	
mobile search	

Tableau 5 - Correspondance des mots-clés fournis par M. Müller

Les résultats sont très satisfaisants en considérant que parmi sept mots-clés sélectionnés initialement par M. Müller (les termes « software engineering » et « mobile search » ont été ajoutés par après pour compléter la liste), tous se retrouvent de près ou de loin dans les termes extraits de manière automatique. De plus, quatre de ces sept mots ont une correspondance exacte, où uniquement la langue diffère (ceci est dû au fait que la majorité des documents de M. Müller sont en anglais).

Notons que les résultats ne sont de loin pas aussi positifs pour les autres auteurs, qui n'ont pas pu fournir un catalogue de documents aussi complet et diversifié que M. Müller (98 documents).

Prenons par exemple M. Yann Bocchi (20 documents). Remarquons que les unigrammes tels que des abréviations ou des noms d'entreprises n'ont pas été pris en compte. Le Tableau 6 p. 65 présente les résultats obtenus.

Mots-clés fournis	Correspondance dans les termes extraits
exploitation de services	service oriented architecture
IT Management	
mobile development	
software engineering	
enterprise architecture	
database	bases de données
analyse et conception	
project management	project management

Tableau 6 - Correspondance des mots-clés fournis par M. Bocchi

Le résultat est moins convaincant : seuls trois des huit termes fournis par M. Bocchi ont une correspondance dans la liste des termes extraits.

Il est donc clair que pour obtenir des termes pertinents, il faut disposer de nombreux documents, qui représentent bien la diversité des expertises de l'auteur. Évidemment, l'extraction automatique est incapable de déduire des expertises qui ne sont pas mentionnées dans les documents fournis. C'est pourquoi des termes comme « Analyse et conception » risquent d'être difficiles à extraire, car il s'agit d'un terme général qui ne sera probablement pas mentionné littéralement dans les documents.

4.1.1.2 Pourcentage de termes utilisables

Parmi les termes extraits, certains n'ont tout simplement pas de réelle signification ou ne sont pas pertinents, mais peuvent difficilement être filtrés de manière automatique. Voici quelques exemples :

- **Nom d'une personne** : mis à part le nom de l'auteur, il est possible que les noms d'autres personnes apparaissent souvent dans les documents d'un auteur. Il est rarement intéressant de conserver ces noms.
- **Nom d'un projet** : Les projets auxquels participent les auteurs peuvent apparaître dans la liste des termes extraits. La valeur de cette information est plus subjective, car selon l'auteur ou le projet, il peut être intéressant de la conserver. Généralement, ces noms ne sont par contre que rarement des indicateurs d'expertises d'un auteur.
- **Termes généraux** : Certains termes extraits sont très généraux et ne fournissent pas de valeur ajoutée. Des termes tels que « informatique de gestion » ne sont pas des indicateurs d'expertises pour un institut essentiellement axé sur l'informatique de gestion.
- **Termes « hors sujet »** : Certains termes apparaissent dans de nombreux documents mais n'indiquent pas une expertise. Le terme « advisory board » (ou conseil d'administration) a par exemple été extrait pour M. Müller.
- **N-grammes problématiques** : L'inconvénient d'extraire des n-grammes d'une longueur supérieure à deux mots est l'apparition de redondances. Si le terme « natural language processing » est extrait, les bigrammes « natural language » et « language processing » le seront également. Ce genre de redondance fait partie des termes indésirables.

  Les n-grammes erronés tirés du milieu d'une phrase sont un autre problème que présentent les longs n-grammes, comme par exemple « profiles individuals undertaking ». Sorti du contexte de la phrase, ce terme n'a aucune signification.

Pour les deux mêmes auteurs cités dans la section précédente, voici la statistique des mots jugés indésirables parmi la liste des 50 premiers termes extraits :

- **Henning Müller** : 14 mots sur 50 (**28%**) peuvent être jugés indésirables. Cela signifie tout de même que 36 termes décrivent de manière adéquate une expertise de M. Müller.
- **Yann Bocchi** : 30 mots sur 50 (**60%**) peuvent être considérés sans doute comme étant indésirables. Les 20 termes restants sont plus intéressants, mais seulement 6 ou 7 d'entre eux sont réellement indicateurs d'une expertise.

La quantité de documents a donc également un impact important sur le nombre de termes indésirables : avec peu de documents, les chances sont meilleures qu'un terme non-pertinent obtienne un score élevé et fasse partie de la liste des 50 résultats conservés.

4.1.2 Performances

La performance n'était pas l'objectif principal de ce travail, mais il est tout de même intéressant de relever quelques chiffres donnant une idée de la vitesse d'exécution de certaines fonctionnalités de l'application.

Les mesures ont été effectuées sur le serveur de production de la HES-SO, qui est une machine virtuelle[70]. Les performances de cet ordinateur ne sont donc pas optimales.

4.1.2.1 Exécution du processus global

La durée totale du processus est de 1 heures et 34 minutes. La Figure 42 p. 67 montre le détail de la durée de chaque étape. Pour information, le même processus lancé dans une machine virtuelle sur mon ordinateur privé ne dure que 37 minutes, ce qui confirme les performances limitées du serveur de production.

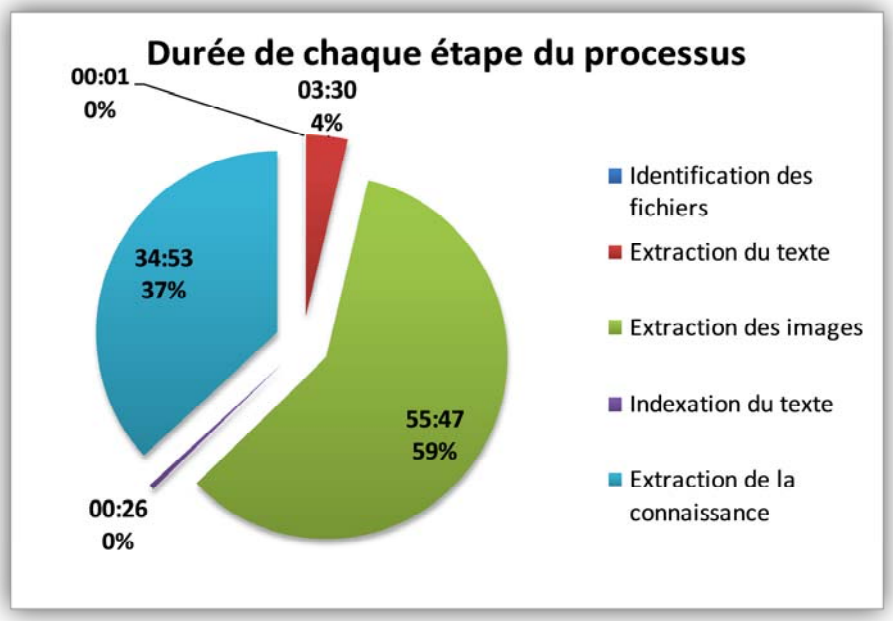


Figure 42 - Durée des étapes du processus global

Les étapes d'extraction des images et de la connaissance éclipsent totalement le reste du processus. L'extraction des images est longue pour la raison suivante : il faut considérer que pour chaque document traité, il existe une multitude d'images qui doivent être extraites, placées dans un répertoire, redimensionnées et copiées dans leur forme miniaturisée dans un autre répertoire du serveur. L'extraction de connaissances est longue car de multiples itérations sont faites sur les documents afin de trouver des collocations et de filtrer les résultats obtenus.

4.1.2.2 Recherche

Une série de dix requêtes de recherche ont été effectuées pour évaluer la durée moyenne de la recherche dans l'index. La mesure se base uniquement sur les trois

[70] Informations : http://en.wikipedia.org/wiki/Virtual_machine

accès réalisés sur l'index (anglais, français, allemand) et ne prend pas en compte la durée nécessaire pour construire et afficher la page Web.

Des requêtes de divers types ont été effectuées :

- Terme unique (ex : `google`)
- Multi-termes en anglais (ex : information `retrieval`)
- Multi-terme avec correspondance exacte (ex : `"open source"`)
- Terme unique avec indication de l'auteur (ex : `systems +author:"Jean Hennebert"`)
- Multi-termes français & allemand (ex : « projet informatique de gestion »)

La durée moyenne d'une requête est de 0.25 secondes, ce qui est comparable à une recherche sur Google. Évidemment, cela ne signifie pas que l'application est aussi performante que Google, qui gère des milliards de données, mais cela permet de confirmer que la durée d'une recherche se situe dans le même ordre de grandeur et n'est pas exponentiellement plus longue.

4.2 Principaux obstacles rencontrés

Il est inévitable de se heurter à des défis durant un travail de bachelor. La recherche de solutions a été une partie importante de ce travail, qui traitait de concepts, d'outils et de technologies qui m'étaient inconnus. Cette section présente donc certains des problèmes majeurs ou récurrents rencontrés durant ce travail. Des problèmes mineurs sont mentionnés dans les rapports hebdomadaires qui se trouvent sur le CD joint à ce rapport.

4.2.1 Documentation de librairies incomplète

Utiliser une librairie inconnue n'est jamais une tâche aisée. Il est donc capital qu'une documentation exhaustive soit fournie ou qu'un forum actif de la communauté soit disponible. Malheureusement, la nature des librairies utilisées a posé quelques problèmes par rapport à la documentation.

D'un côté, la librairie Mahout est un projet encore très jeune, ce qui signifie que la documentation technique (Javadoc) est très lacunaire. De plus, très peu d'informations par rapport à son fonctionnement sont disponibles sur le site officiel du projet et des recherches sur Google ne rapportent quasiment pas de résultats. L'implémentation a donc été faite de manière empirique et de nombreuses itérations ont été nécessaires avant d'arriver à une solution fonctionnelle.

De l'autre côté, l'utilisation du projet « Woodstock » pour la création des pages Web a également été difficile par moments. Le projet ayant été abandonné, très peu d'informations étaient disponibles et la plupart des résultats de recherche étaient des instructions de migration vers la nouvelle solution « ICEfaces ».

Beaucoup de persévérance et de nombreuses recherches Google ont finalement mené aux informations recherchées dans la plupart des cas, mais ces difficultés m'ont fait réaliser l'importance d'utiliser des librairies maintenues à jour et bien documentées, lorsqu'il est possible de le faire.

4.2.2 Encodages des caractères

Les différents encodages de caractères ont souvent été la source de problèmes par le passé et ce travail ne fait pas exception. Voyant que le projet allait traiter des fichiers pouvant contenir des caractères spéciaux, il a été décidé tôt dans le projet d'utiliser l'encodage UTF-8 pour toutes les données.

La difficulté provenait du fait qu'il fallait veiller à ce que toutes les interactions avec les données soient faites avec ce même encodage. De la lecture des fichiers avec Java au chargement des données par Ajax en passant par l'affichage du texte avec des composants JSF et l'extraction des connaissances avec Mahout, des paramétrages ont dû être effectués à de nombreux niveaux de l'application pour assurer la conservation de l'encodage de bout en bout.

4.2.3 Prise en main de JSF

La reprise du projet de M. Eggel avait des avantages certains mais également quelques désavantages. Les technologies Web Java (JSP, JSF, Servlets, …) m'étaient totalement inconnues avant le début de ce travail.

De nombreuses heures ont donc été dédiées à la compréhension de la philosophie de JSF ainsi qu'à l'étude du fonctionnement des différentes librairies de balises utilisées (Woodstock, Tomahawk, JSTL, JSP, JSF Core, JSF HTML, etc…).

Il est clair que la technologie JSF peut être extrêmement puissante dans certains cas : l'affichage des résultats dans la page de recherche ainsi que la pagination sont gérés de manière très élégante, par exemple. Par contre, il était parfois compliqué de réaliser des éléments visuels très simples, ce qui est dû en partie au manque de flexibilité de certains composants graphiques Woodstock.

4.3 Respect du cahier des charges

Toutes fonctionnalités obligatoires ont été réalisées. Certaines fonctionnalités optionnelles n'ont pas été implémentées :

- Extraction du texte de fichiers LaTeX[71] : M. Müller m'a indiqué que les fichiers exportés à partir d'un fichier LaTeX (documents PDF la plupart du temps) sont généralement stockés dans le même répertoire que les fichiers « .tex » source. Il est donc inutile d'en extraire le texte.
- Indexation des images avec GIFT[72] : Cette fonctionnalité a été abandonnée au profit de fonctionnalités plus critiques pour deux raisons.

  Premièrement, le temps à disposition n'aurait pas été suffisant pour intégrer l'aspect d'analyse d'images dans l'application.

  Deuxièmement, l'analyse du texte permet certainement d'extraire des connaissances de manière plus fiable. En effet, de nombreuses images contenues dans les documents sont des graphiques dont il aurait été difficile d'extraire la signification.
- Analyse des similarités entre auteurs au niveau des images : Cette fonctionnalité dépendant de l'indexation des images, elle a également été abandonnée.

De plus, des fonctionnalités initialement non prévues ont été ajoutées par la suite. Par exemple, la page de recherche de mots-clés a été implémentée très tard dans le développement car elle a été identifiée comme un très bon moyen de trouver des personnes possédant une expertise donnée, ce qui était un des objectifs principaux du travail.

4.4 Respect du planning

Un planning basique séparant le travail en cinq phases a été défini durant les premières semaines du travail (voir Annexes). Les cinq phases sont les suivantes :

- Mise en place / Découverte (40h)
- Phase n°1 du développement : moteur de recherches (60-80h)
- Phase n°2 du développement : gestion des connaissances (120-140h)
- Améliorations / finitions (80h)
- Documentation (40h)

Les phases ont été respectées d'assez près, mais le déroulement était un peu différent pour la phase d'améliorations et de finitions. En effet, les corrections de bugs, effets Ajax et autres finitions visuelles ont été réalisées directement durant les phases de développement.

Ces deux semaines ont donc été partagées entre la finition de l'application Web et la documentation, qui a pris plus de temps que prévu. Ceci est dû en majeure partie au fait qu'il était initialement prévu de mettre à jour constamment la documentation par rapport au progrès réalisé. Malheureusement, une fois que la phase n°2 du développement a débuté, la documentation n'a plus été tenue à jour régulièrement. Plus de temps a donc été nécessaire en fin de projet pour rédiger le rapport.

[71] Lamport TeX
[72] GNU Image Finding Tool

4.5 Améliorations futures possibles

Ce travail ne fait certainement qu'effleurer les possibilités des domaines de la gestion des connaissances et de l'identification d'experts. Un certain nombre d'améliorations envisageables sont donc décrites dans cette section.

4.5.1 Visualisations

Une petite sélection de visualisations a été implémentée dans le cadre de ce travail. Il existe certainement une pléthore d'autres librairies fournissant des visualisations uniques, créatives et innovatrices. Disposer d'une large panoplie de visualisations pourrait certainement être bénéfique pour les utilisateurs, qui pourraient découvrir des connaissances ou des liens dont ils ne soupçonnaient pas même l'existence.

4.5.2 Performances

Le processus d'extraction d'images ainsi que l'extraction de connaissances sont relativement longs et pourraient certainement être améliorés. Une limite d'images récupérées par document ou la détection de dimensions d'images non désirables (p.ex : dimensions approximatives d'une feuille A4 pour éliminer les lettres, pages de magazines, etc…) seraient des moyens d'accélérer l'extraction des images.

L'utilisation d'un serveur Hadoop, prévue normalement par la librairie Mahout, pourrait également améliorer la vitesse d'extraction des connaissances.

4.5.3 Gestion des connaissances

La librairie Mahout dispose de nombreux algorithmes pour filtrer une collection de données et les fonctions utilisées dans le cadre de ce projet récupèrent bien plus que la liste des n-grammes. Malheureusement, il n'a pas été possible dans le temps à disposition d'analyser la structure et l'utilité de tous les fichiers générés.

Une amélioration de la pertinence des résultats extraits serait donc certainement atteignable.

4.6 Conclusion personnelle

Ce travail a été une expérience particulièrement enrichissante. Découvrir les concepts du domaine de la gestion des connaissances tout en travaillant avec une technologie inconnue était un défi motivant à relever.

J'ai apprécié le fait que l'organisation administrative du travail (gestion de projet, séances, etc…) pouvait être structurée assez librement avec le professeur responsable. Dans notre cas, le développement était la priorité, ce qui a permis de limiter la masse de tâches administratives à un niveau raisonnable qui m'a totalement convenu.

Certaines étapes du projet ont nécessité beaucoup de patience et d'acharnement. La recherche d'un outil d'extraction de collocations a par exemple été longue et exigeante. En effet, le nombre d'outils est assez restreint et il n'a pas été simple de les identifier. De plus, une certaine frustration pouvait être ressentie après chaque test d'implémentation infructueux des outils non retenus. Au final, ces quelques difficultés m'ont permis d'acquérir plus d'expérience et d'avoir une vue plus globale des solutions disponibles.

Je pense que l'application développée est très satisfaisante. La qualité des mots-clés extraits pour M. Müller m'a surpris : je ne m'attendais pas à des résultats aussi pertinents. Les diverses visualisations ont également été intéressantes à découvrir, implémenter et modifier pour les adapter à nos besoins. L'inclusion d'éléments dynamiques Ajax m'a permis d'explorer un de mes intérêts personnels : le design.

Finalement, ce projet a sans aucun doute été le plus ambitieux et le plus stimulant de ma formation à la HES-SO. Le meilleur aspect de ce travail était le fait d'avoir pu livrer une application fonctionnelle ayant une utilité concrète, qui intègre tout de même des concepts plutôt complexes tels que l'extraction de contenu de documents aux formats propriétaires, la création d'un index de moteur de recherche ou encore l'extraction de mots-clés pertinents d'une masse d'informations considérable.

4.7 Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

5 Références

[1] Henning Müller, Donnée du travail de bachelor - Gestion de la connaissance et recherche d'informations avec des archives de documents, 2010.

[2] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke, "Formal Models for Expert Finding in Enterprise Corpora," in *SIGIR'06*, Seattle, Washington, Etats-Unis, 2006.

[3] Krisztian Balog, "People Search in the Enterprise," in *SIGIR'07*, Amsterdam, Pays-Bas, 2007.

[4] KEA - Keyphrase Extraction Algorithm. [En ligne] - Accès le 2 août 2010. http://www.nzdl.org/Kea/index.html

[5] Wikipedia - Plateforme Java. [En ligne] - Accès le 19 Juin 2010. http://en.wikipedia.org/wiki/Java_%28software_platform%29

[6] Communiqué de presse d'Oracle. [En ligne] - Accès le 19 Juin 2010. http://www.oracle.com/us/corporate/press/018363

[7] Design Goals of the Java Programming Language. [En ligne] - Accès le 9 août 2010. http://java.sun.com/docs/white/langenv/Intro.doc2.html

[8] Wikipedia - JavaServer Faces. [En ligne] - Accès le 9 août 2010. http://en.wikipedia.org/wiki/JavaServer_Faces

[9] Wikipedia - Enterprise JavaBean. [En ligne] - Accès le 9 août 2010. http://en.wikipedia.org/wiki/Enterprise_JavaBean

[10] ICEfaces replaces Project Woodstock. [En ligne] - Accès le 9 août 2010. http://www.prweb.com/releases/2008/12/prweb1765694.htm

[11] Wikipedia - Ajax. [En ligne] - Accès le 2 août 2010. http://en.wikipedia.org/wiki/Ajax_%28programming%29

[12] Apache Tika. [En ligne] - Accès le 6 juillet 2010. http://tika.apache.org/

[13] Apache PDFBox. [En ligne] - Accès le 6 juillet 2010. http://pdfbox.apache.org/

[14] Apache POI. [En ligne] - Accès le 6 juillet 2010. http://poi.apache.org/

[15] John Cowan. TagSoup. [En ligne] - Accès le 6 juillet 2010. http://home.ccil.org/~cowan/XML/tagsoup/

[16] What is Lucene. [En ligne] - Accès le 10 juillet 2010. http://lucene.apache.org/#What+Is+Lucene%3F

[17] Lucene Java. [En ligne] - Accès le 6 juillet 2010. http://lucene.apache.org/java/docs/index.html

[18] ht://Dig Features and Requirements. [En ligne] - Accès le 6 juillet 2010. http://www.htdig.org/require.html

[19] ht://Dig. [En ligne] - Accès le 6 juillet 2010. http://www.htdig.org/

[20] Lucene Java 3.0.2 and 2.9.3 available. [En ligne] - Accès le 6 juillet 2010. http://lucene.apache.org/java/docs/index.html#18+June+2010+-+Lucene+Java+3.0.2+and+2.9.3+available

[21] Solr Features. [En ligne] - Accès le 6 juillet 2010.
http://lucene.apache.org/solr/features.html

[22] Terrier. [En ligne] - Accès le 6 juillet 2010.
http://terrier.org/

[23] Apache Mahout. [En ligne] - Accès le 19 Juin 2010.
http://mahout.apache.org/

[24] Wikipedia - Hadoop. [En ligne] - Accès le 19 Juin 2010.
http://en.wikipedia.org/wiki/Hadoop

[25] Frank Eibe and Medelyan Olena. KEA - Lisez-moi. [En ligne].
http://www.nzdl.org/Kea/Download/Kea-5.0-Readme.txt

[26] jQuery - Site officiel. [En ligne] - Accès le 3 août 2010.
http://jquery.com

[27] Diverses versions de NetBeans. [En ligne] - Accès le 2 août 2010.
http://netbeans.org/downloads/index.html

[28] Nick Burch. Apache POI Project. [En ligne] - Accès le 19 mai 2010.
http://poi.apache.org/text-extraction.html

[29] Dustin Marx. Dustin's Software Development Cogitations and Speculations. [En ligne] - Accès le 19 mai 2010.
http://marxsoftware.blogspot.com/2008/02/using-openxml4j-to-access-office-open.html

[30] Tika Facade. [En ligne] - Accès le 6 juillet 2010.
http://tika.apache.org/0.7/api/org/apache/tika/Tika.html

[31] Otis Gospodnetić and Erik Hatcher, *Lucene in Action*, Tiffany Taylor, Ed. Greenwich, Connecticut, États-Unis: Manning Publications Co., 2005.

[32] Wikipedia - Auto-complétion. [En ligne] - Accès le 3 août 2010.
http://en.wikipedia.org/wiki/Autocomplete

[33] Brian Suda and Matt Riggott. (2004, août) A List Apart - Enhance Usability by Highlighting Search Terms. [En ligne] - Accès le 3 août 2010.
http://www.alistapart.com/articles/searchhighlight/

[34] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts, États-Unis: Addison-Wesley Professional, 1994.

[35] BC Holmes. Java Advanced Imaging. [En ligne] - Accès le 4 août 2010.
http://i-proving.ca/space/Technologies/Java+Advanced+Imaging

[36] Mathew Mannion. Using subsample averaging instead of scaling in JAI to get better results. [En ligne] - Accès le 4 août 2010.
http://blogs.warwick.ac.uk/mmannion/entry/using_subsample_averaging

[37] HTML URL Encoding Reference. [En ligne] - Accès le 4 août 2010.
http://www.w3schools.com/TAGS/ref_urlencode.asp

[38] Wikipedia - Index inversé. [En ligne] - Accès le 4 août 2010.
http://en.wikipedia.org/wiki/Inverted_index

[39] Javadoc Lucene - IndexWriter. [En ligne] - Accès le 5 août 2010. http://lucene.apache.org/java/3_0_1/api/core/org/apache/lucene/index/IndexWriter.html

[40] Wikipedia - Collocation. [En ligne] - Accès le 6 août 2010. http://en.wikipedia.org/wiki/Collocation

[41] Wikipedia - n-gramme. [En ligne] - Accès le 6 août 2010. http://en.wikipedia.org/wiki/N-gram

[42] Collocations in Mahout. [En ligne] - Accès le 6 août 2010. https://cwiki.apache.org/confluence/display/MAHOUT/Collocations

[43] Ted Dunning. Surprise and Coincidence. [En ligne] - Accès le 6 août 2010. http://tdunning.blogspot.com/2008/03/surprise-and-coincidence.html

[44] Creating Vectors from Text. [En ligne] - Accès le 6 août 2010. https://cwiki.apache.org/confluence/display/MAHOUT/Creating+Vectors+from+Text

[45] Intro to Apache Mahout. [En ligne] - Accès le 6 août 2010. http://www.slideshare.net/gsingers/intro-to-apache-mahout

[46] Javadoc - TopScoreDocCollector. [En ligne] - Accès le 9 août 2010. http://lucene.apache.org/java/3_0_1/api/core/org/apache/lucene/search/TopScoreDocCollector.html

[47] Javadoc - MoreLikeThis. [En ligne] - Accès le 9 août 2010. http://lucene.apache.org/java/3_0_1/api/contrib-queries/org/apache/lucene/search/similar/MoreLikeThis.html

[48] Aaron Johnson. How MoreLikeThis works in Lucene. [En ligne] - Accès le 9 août 2010. http://cephas.net/blog/2008/03/30/how-morelikethis-works-in-lucene/

[49] Visualizing Trends in Knowledge Management. [En ligne] - Accès le 10 août 2010. http://www.springerlink.com/content/n303130n61376048/

[50] Wikipedia - Tag Clouds. [En ligne] - Accès le 10 août 2010. http://en.wikipedia.org/wiki/Tag_cloud

[51] Ivan Eggel, "Analyse komplexer Dokumente für die Extraktion und Indizierung von Texten und Bildern zur anschliessenden Informations-Suche," HES-SO Valais / Wallis, Sierre, Rapport de Travail de bachelor 2008.

6 Tables

6.1 Table des illustrations

6.2 Table des extraits de code

6.3 Table des tableaux

6.4 Glossaire

Terme	Signification
Ajax	Technologie permettant de dynamiser des applications Web
Collocation	Groupe de mots apparaissent souvent dans un document
HTTP	Hypertext transfer protocol : Protocole de communication Web
Indexation	Structuration de données en vue d'un accès rapide en recherche
Java	Plateforme logicielle et langage de programmation
JSF	JavaServer Faces : technologie Web Java
Lucene	Librairie pour le développement de moteurs de recherche
Mahout	Libaririe permettant l'extraction de collocation
MindMap	Carte heuristique montrant des liens entre des concepts
N-gramme	Groupe de « n » éléments (mots, lettres, etc…)
Servlet	Classe Java spécialisée traitant des requêtes HTTP
Tag Cloud	Nuage de mots-clés pondérés
Tika	Librairie d'extraction de texte de documents complexes

7 Annexes

Les annexes sont répertoriées séparément (voir page suivante).