

SPIM

Thèse de Doctorat



école doctorale sciences pour l'ingénieur et microtechniques
UNIVERSITÉ DE BOURGOGNE

Approche Big Data et Web
Sémantique pour la fouille et la
classification automatique de
données web
Application aux nouvelles économiques

THOMAS HASSAN

SPIM

Thèse de Doctorat



école doctorale sciences pour l'ingénieur et microtechniques
UNIVERSITÉ DE BOURGOGNE

THÈSE présentée par
THOMAS HASSAN
pour obtenir le
Grade de Docteur de
l'Université de Bourgogne Franche-Comté
Spécialité : **Informatique**

Approche Big Data et Web Sémantique pour la fouille et la classification automatique de données web

Application aux nouvelles économies

Unité de Recherche :
Laboratoire Electronique, Informatique et Images (LE2I)

Soutenue publiquement le XX/XX/2017 devant le Jury composé de :

CHRISTOPHE CRUZ	Directeur de thèse	Maître de Conférence à l'Université de Bourgogne
AURÉLIE BERTAUX	Coencadrant de thèse	Maître de Conférence à l'Université de Bourgogne

SOMMAIRE

I Contexte	1
1 Introduction	3
1.1 Contexte	3
1.2 Définitions	6
1.3 Problématique	7
1.3.1 Principales approches en apprentissage automatique	8
1.3.2 Apprentissage automatique et classification	9
1.3.3 Recherche d'information sur le web	10
1.3.4 Approches sémantiques	12
1.3.5 Verrous scientifiques	13
1.4 Contributions	13
1.5 Composition du document	14
II Classification automatique de données dans un contexte Big Data	15
2 Etat de l'art : classification automatique basée sur la sémantique	17
2.1 Classification hiérarchique multi-étiquette	18
2.2 Approches basées sur les ontologies	21
2.2.1 Ontologies et classification	21
2.2.2 Raisonnement sémantique pour la classification	23
2.3 Proposition : processus de "HMC Sémantique"	23
3 Apprentissage de taxonomie non supervisée	25
3.1 Architecture "SHMC"	26
3.2 Indexation	28
3.2.1 Description de l'approche	29
3.2.2 Implémentation	31
3.3 Vectorisation	33
3.3.1 Description de l'approche	33

3.3.2	Implémentation	34
3.4	Hiérarchisation	35
3.4.1	Description de l'approche	35
3.4.2	Implémentation	36
3.5	Évaluation quantitative	37
3.5.1	Jeux de données	38
3.5.2	Environnement de test	38
3.5.3	Paramétrage	38
3.5.4	Résultats	39
3.6	Évaluation qualitative	41
3.6.1	Évaluation lexicale	43
3.6.1.1	Protocole d'évaluation	43
3.6.1.2	Résultats	44
3.6.2	Évaluation hiérarchique	45
3.6.2.1	Protocole d'évaluation	46
3.6.2.2	Résultats	47
3.7	Conclusion	48
4	Classification basée sur un moteur de raisonnement	49
4.1	Résolution	49
4.1.1	Définitions	50
4.1.2	Description de l'approche	50
4.1.3	Implémentation	53
4.2	Réalisation	53
4.2.1	Description de l'approche	54
4.2.2	Implémentation	55
4.3	Évaluation	57
4.3.1	Méthode d'évaluation de la classification	57
4.3.2	Évaluation du passage à l'échelle	58
4.3.2.1	Jeux de données	58
4.3.2.2	Résultats	58
4.3.3	Évaluation qualitative en contexte non-supervisé	60
4.3.3.1	Variation des seuils <i>st1</i> et <i>st2</i>	60
4.3.3.2	Variation des seuils de résolution	61
4.3.3.3	Interprétations	63

4.3.4	Évaluation qualitative en contexte supervisé	63
4.3.4.1	Description	64
4.3.4.2	Résultats	65
4.4	Conclusion	66
III	Recherche d'information sur le web par classification sémantique	67
5	Etat de l'art : recherche d'information sur le web	69
5.1	Fondements de la recherche d'information	70
5.1.1	Recherche d'information sur le web	71
5.1.2	Caractéristiques d'un crawler web	71
5.2	Crawlers ciblés	75
5.2.1	Stratégies initiales de parcours	77
5.2.2	Crawlers sémantiques	78
5.2.3	Approches basées sur l'apprentissage automatique	79
5.2.4	Approches hybrides	81
5.3	Proposition : processus SEMXDM	82
6	SEMXDM : web crawler sémantique adaptatif	85
6.1	Architecture SEMXDM	86
6.1.1	Module de recommandation	87
6.1.2	Module de recherche	88
6.1.3	Module de classification	89
6.1.4	Module d'évolution	92
6.1.5	Module de priorité	96
6.2	Implémentation	100
6.2.1	Module de recommandation	100
6.2.2	Module de recherche	101
6.2.3	Module de classification	101
6.2.4	Module de maintenance	101
6.2.5	Module de priorité	103
6.3	Évaluation	103
6.3.1	Méthodes d'évaluation des robots d'indexation	103
6.3.2	Évaluation quantitative	104
6.3.2.1	Environnement de test	104

6.3.2.2 Résultats	104
6.3.3 Évaluation qualitative pour le croisement d'information	109
6.3.3.1 Impact de l'évolution du modèle	110
6.4 Conclusion	114
7 Application au domaine de la veille stratégique	117
7.1 Introduction : veille stratégique	117
7.2 Système de recommandation existant	120
7.2.1 Base de connaissances	120
7.2.2 Processus de recommandation	121
7.2.3 Objectifs de l'application	121
7.3 Architecture	124
7.3.1 Apprentissage supervisé d'un modèle de classification de nouvelles économiques	124
7.3.2 Application du processus de recherche pour la veille stratégique .	125
7.3.3 Déploiement	127
7.3.4 Aperçu du prototype	129
IV Discussion et conclusion	133
8 Conclusion	135
8.1 Discussion, contributions	135
8.2 Limitations et travaux futurs	136
Bibliographie	139
A Logiques de description et raisonnement logique	159
A.1 Les logiques de description et le web sémantique	159
A.1.1 Les bases de connaissances	161
A.1.2 Les familles de logiques de description	162
A.1.3 Le langage ontologique OWL	164
A.2 Les raisonnements logiques	167
A.2.1 Les inférences	167
A.2.2 Approches de raisonnement	168
A.2.2.1 Raisonnement basé sur la comparaison structurelle	168
A.2.2.2 Raisonnement basé sur l'algorithme de tableaux	169

A.2.3 Complexité de l'inférence	169
A.2.4 Les raisonneurs pour OWL	170
B Le paradigme MapReduce	173
B.1 MapReduce et architecture Hadoop	173
B.2 Implémentation de tâches MapReduce	174
B.2.1 Word Count	175
B.2.2 Vectorisation (algorithme stripes)	176
B.2.3 Résolution (règles)	177
C Production des documentalistes	179
C.1 Méthode de travail des documentalistes	179
C.1.1 Sources et supports de l'information économique	179
C.1.2 Sélection des informations	180
C.1.3 Extraction des informations	181
C.1.4 Rédaction de la synthèse	182
D Implémentation de l'analyseur syntaxique	183
D.1 Objectifs	183
D.2 Tokenisation	185
D.3 Stemming	185
D.4 Suppression des mots-vides	186
D.4.1 Flexions, synonymes	186
D.4.2 Ressource utilisée	187
D.5 Filtres Regex	187

|

CONTEXTE

INTRODUCTION

1.1/ CONTEXTE

L'hypothèse d'une transition majeure pour l'économie globale actuelle, dans un contexte de mondialisation et de consumérisme de masse dont les limites sont observées aujourd'hui, est de plus en plus plébiscitée dans les milieux scientifiques et économistes.

Cette transition, la "révolution numérique"¹ est intimement liée à l'économie des données et la démocratisation de l'intelligence artificielle, nées de l'explosion exponentielle de la quantité des données numériques générées chaque jour dans le monde. L'explosion quantitative des données numériques issues du web "2.0", ou web social, ont fait naître de nouvelles problématiques, mais aussi de nouvelles opportunités économiques et sociales McAfee et al. (2012), au sein du domaine d'activité aujourd'hui nommé Big Data, ou "mégadonnées". L'intérêt du Big Data est également majeur dans la sphère scientifique, dans des domaines tels que la génomique, la physique des particules ou l'astronomie, où des quantités massives de données sont à disposition. L'exploitation de ces données qui par le passé était difficilement concevable est désormais l'objet de nombreux projets de recherche.

En 2001, un rapport de recherche du groupe Gartner² décrivait pour la première fois la notion de "Big Data" et les caractéristiques qui définissent ce domaine. Dans ce rapport, la problématique de gestion de ce nouveau type de données est déjà définie autour de trois dimensions : elle correspond à l'union des problématiques liées au Volume, à la Vélocité et la Variété de ce type de données. Le Volume définit la quantité croissante de données, générée et stockée au fil du temps par les réseaux sociaux, les données de capteurs, d'objets connectés, etc Chen et al. (2014). La Vélocité concerne la vitesse importante de production des données, et par conséquent le besoin de traitement rapide des données. La Variété représente la grande hétérogénéité des formats du Big Data. En particulier, les données non-structurées et semi-structurées nécessitent des traitements importants, et représentent 90% du contenu associé généralement au Big Data (pages web, documents en langage naturel, audio, etc.) Syed et al. (2013). Ces trois dimensions sont encore utilisées aujourd'hui par la grande majorité des acteurs industriels et scientifiques.

1. http://www.lemonde.fr/economie/article/2017/01/17/la-revolution-numerique-est-une-opportunite_5064208_3234.html http://www.lesechos.fr/12/08/2015/LesEchos/21999-029-ECH_les-quatre-piliers-de-la-revolution-numerique.htm

2. Entreprise des technologies de l'information et de la communication spécialisée dans le conseil pour l'analyse de données, à l'époque nommée META-GROUP

Le domaine étant relativement jeune, les perspectives du traitement des Big Data sont extrêmement variées et en partie encore insoupçonnées. Cependant, les infrastructures et les services liés au Big Data se sont développés rapidement à travers d'offres de cloud computing (IaaS, PaaS et SaaS³), afin de répondre aux problématiques techniques inhérentes à ce type de données. De fait, il existe une corrélation entre des innovations technologiques majeures, et les contraintes de gestion de données inhérentes aux dimensions du Big Data, i.e. Volume, Variété et Vélocité Chen et al. (2014) Hitzler et al. (2013).

D'autres dimensions nommées par des "V" supplémentaires ont émergé au cours du temps, comme la Véracité, la Visualisation ou la Valeur. Contrairement aux 3V du Big Data, il ne s'agit plus d'indicateurs quantitatifs, liés à la robustesse face aux données, et ne sont donc pas considérées comme des dimensions au même titre que Valeur, Vélocité et Variété. En revanche, la Valeur définit la pertinence d'une information pour l'utilisateur final⁴, c'est donc une caractéristique primordiale.

Dans le même temps, les méthodes et paradigmes d'exploitation des données ont progressé. L'accès à ces technologies étant de plus en plus facilité, de nombreuses entreprises dans des domaines variés considèrent la mise en place de processus Big Data pour valoriser leurs données, grâce à un coût de mise en place et de maintenance réduit. Là où l'optimisation des processus de production tend à atteindre ses limites, la maîtrise des Big Data offre de nouvelles perspectives pour accroître son avantage compétitif.

La définition du Big Data a par conséquent évolué, et la définition actuelle communément utilisée évoque également cette finalité, i.e. l'acquisition de données/d'informations de Valeur à travers l'exploitation du Big Data : *"Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value"*⁵.

Gary King, professeur et directeur de l'institut des sciences sociales quantitatives à l'université d'Harvard dit en 2016, *"le sujet premier du Big Data n'est en fait pas les données elles-mêmes [...] La révolution, c'est que nous savons désormais quoi faire avec"*⁶. En effet, c'est dans l'exploitation et les méthodes d'analyse des données que réside le réel intérêt du Big Data.

Les besoins exprimés de façon croissante par les entreprises et les chercheurs pour ces méthodes d'analyse et les métiers associés ("data scientists" et "data analysts") sont reflétés à travers les statistiques de recherche des utilisateurs du moteur Google (google trends). L'impact du Big Data et des méthodes d'exploitation sont désormais mieux comprises dans la sphère scientifique et économique.

Les travaux de thèse ont été réalisés dans le cadre du programme Jeunes Chercheurs Entrepreneurs, et financée par le Conseil Régional de Bourgogne Franche-Comté et le Fond Européen de Développement Régional. Les travaux sont également motivés et financés dans le cadre d'une collaboration entre l'entreprise Actualis SARL, et le laboratoire Electronique, Informatique et Image (LE2I) localisé à l'Université de Bourgogne.

Une des thématiques de recherche du laboratoire est le développement de systèmes de gestion de connaissances basés sur des ontologies, permettant de répondre à des

3. <http://www.culture-informatique.net/cest-quoi-le-cloud-2/>

4. Dans un cadre industriel, il peut s'agir de l'entreprise elle-même ou directement de ses clients

5. <http://www.gartner.com/technology/home.jsp>

6. "Big data is not actually about the data. The revolution is not that there's more data available. The revolution is that we know what to do with it now. That's really the amazing thing."

problématiques métier spécifiques. Les travaux de recherche de l'équipe proposent des solutions à des verrous liés au volume et à l'hétérogénéité des données à traiter, à la contextualisation des informations, au profilage des utilisateurs et à l'analyse de données complexes.

L'entreprise Actualis SARL est spécialisée dans la production et la distribution de revues de presse. Le web est par définition une masse de données immense, et une source d'information clé dans le domaine de la veille économique. Les entreprises utilisent de façon récurrente des taxonomies de domaine, afin de représenter leur connaissance métier. Ils formalisent ainsi un modèle associant une valeur aux données (Lambe, 2014). Dans le cadre de la veille économique, ce type de taxonomie peut permettre de qualifier un information de façon précise. Cette qualification, qui se base sur les concepts clés du domaine de l'entreprise est un vecteur de valeur pour un client. En outre, c'est une étape indispensable dans un objectif de recommandation de l'information. Les travaux de Werner (2015) qui précèdent les travaux décrits dans ce manuscrit se sont intéressés à la mise en place d'un tel système de recommandation, où une approche ontologique permet d'une part une qualification précise de l'information, et d'autre part une description précise du profil utilisateur, en accord avec la description sémantique du domaine définie par une ontologie. Un vecteur de concepts de la taxonomie définit alors les profils et les informations, permettant la recommandation des informations les plus pertinentes pour chaque profil. La figure 1.1 décrit ce système de recommandation.

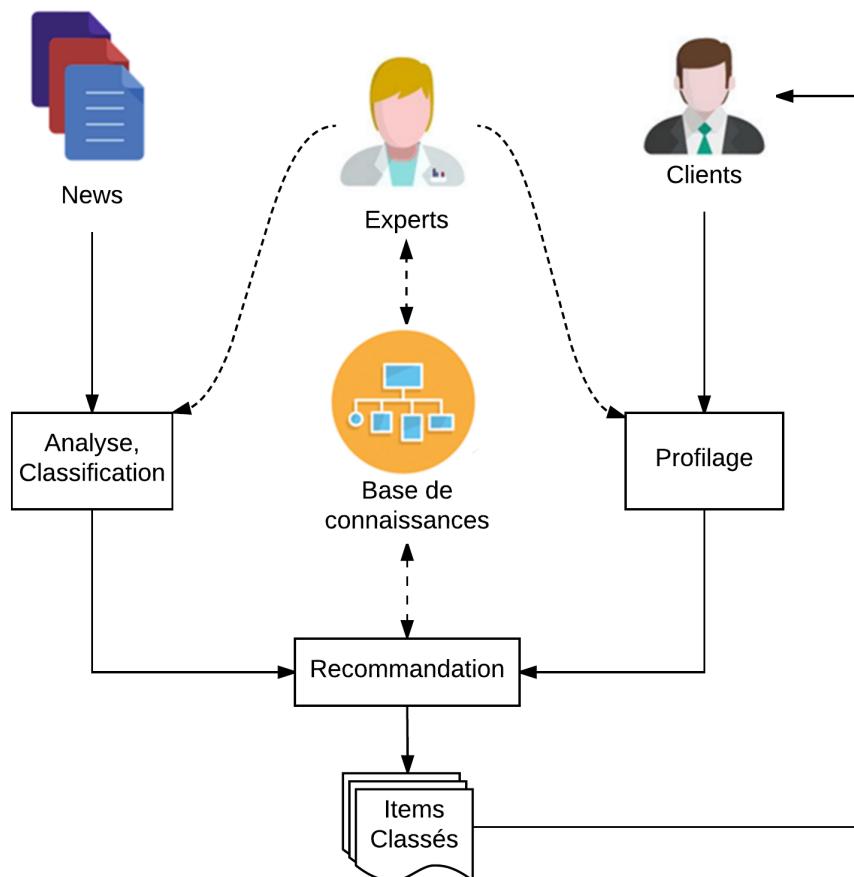


FIGURE 1.1 – Système de recommandation mis en place dans les travaux de Werner (2015)

1.2/ DÉFINITIONS

Cette section recense diverses définitions de termes et de concepts qui seront réutilisés régulièrement dans ce manuscrit. Les définitions spécifiques à une section seront introduites à l'endroit approprié, ou sous la forme d'une note de pied de page.

Définition 1 : Fouille de données

La fouille de données (data mining) concerne l'ensemble des méthodes automatiques et des algorithmes permettant la découverte d'informations valorisables au sein de données numériques. Les données recherchées peuvent être des motifs récurrents au sein des données, des éléments pertinents pour l'utilisateur, etc. Dans un contexte industriel, ces données peuvent ensuite être exploitées pour aider à la prise de décision, ou gagner en information sur un domaine d'activité. Un grand nombre de méthodes ont été développées, notamment basées sur des modèles statistiques. Ces méthodes peuvent être regroupées en types d'approche comme la classification, l'association ou le clustering.

Définition 2 : Logique de description

Les Logiques de description (\mathcal{LD}) ou logiques descriptives sont une famille de langages basée sur le langage \mathcal{AL} (Attributive Language) introduits par Schmidt-Schaub et al. (1991). Elles définissent un formalisme pour représenter la connaissance terminologique d'un domaine d'application de manière structurée. L'annexe A décrit de façon exhaustive les notions liées aux différentes logiques de description, ainsi que leur syntaxe. Selon la complexité du formalisme, l'expressivité de la base de connaissances est plus ou moins grande. Les différentes couches des technologies du web sémantique sont basées sur les logiques de description (voir annexe A).

Définition 3 : Ontologie

Dans le contexte informatique, il existe plusieurs définitions du terme ontologie. La plus communément utilisée est la définition de Studer et al. (1998a), qui définit l'ontologie comme "la spécification explicite et formelle d'une conceptualisation partagée". L'ontologie est une base de connaissances, qui dans son utilisation permet de représenter des connaissances dans un système informatique, à la fois de façon cohérente pour l'homme d'un point de vue sémantique, et de façon interprétable par la machine. En d'autres termes, l'ontologie est un moyen de rapprocher des connaissances issues du monde réel, compréhensibles par l'homme, et une représentation purement numérique des données. Les ontologies sont la composante principale du domaine du web sémantique, et sont des éléments essentiels des systèmes informatiques qui échangent de façon constante des connaissances. La communication de tels systèmes informatiques nécessite une interprétation commune des données échangées, qui peut être produite par l'intermédiaire d'une ontologie. Aussi, les technologies du web sémantique sont fondées sur l'hypothèse du monde ouvert et tirent partie du mécanisme d'inférence. La combinaison de ces deux notions permet la déduction de connaissances nouvelles à partir de connaissances existantes.

Définition 4 : Moteur d'inférence

Un moteur d'inférence est un logiciel qui implémente des algorithmes de raisonnement déductif, selon une logique de description donnée. Ils implémentent nécessairement un mécanisme d'inférence permettant la déduction des conséquences logiques à partir des axiomes de la base de connaissances. Le mécanisme d'inférence varie suivant les moteurs. Les plus communs sont le chaînage avant (forward-chaining) et le chaînage arrière (backward-chaining, voir 4.2.1).

Définition 5 : Traitement automatique du langage

Le domaine du traitement automatique du langage, ou "natural language processing" (NLP), regroupe l'ensemble des méthodes et techniques liées à l'analyse de texte en langage naturel. De fait, ces méthodes sont transverses et s'appliquent à tout domaine où les données concernées sont du texte en langage naturel. A titre d'exemple, nous pouvons citer la recherche d'information ou la fouille de données. Les méthodes peuvent être purement statistiques, symboliques si des ressources terminologiques/sémantiques externes sont utilisées, ou hybrides.

Définition 6 : Web crawler

Un web crawler, ou robot d'indexation, est un agent autonome qui parcourt le web ou une sous-partie du web via le graphe de liens hypertextes, i.e. l'ensemble des liens hypertextes reliant les pages web entre elles. Les crawlers permettent d'indexer les pages web afin de les référencer et de les classer, selon différents algorithmes. Ils constituent donc la technologie principale à la base des moteurs d'indexation et de recherche (Google, Yahoo, Bing, ...) d'information sur le web.

Définition 7 : Classification hiérarchique multi-étiquette sémantique

La classification hiérarchique multi-étiquette sémantique est une architecture d'apprentissage automatique d'une ontologie, décrite de façon exhaustive dans les chapitres 3 et 4 de ce manuscrit. Les termes SHMC (semantic hierarchical multi-label classification), Semantic HMC et HMC Sémantique sont utilisés de façon interchangeable dans le document pour faire référence à cette architecture.

1.3/ PROBLÉMATIQUE

Parallèlement à l'explosion des données générées chaque jour, la popularité grandissante du domaine du traitement massif de données est accompagnée de l'intérêt croissant pour les méthodes d'exploitation de telles masses de données. Le traitement du Big Data peut être vu comme un cas d'application des domaines de l'apprentissage automatique et de la fouille de données (data mining), spécifique à ce type de données. Par conséquent, une majorité d'approches d'analyse de données issus de ces deux domaines ont été considérées pour l'exploitation du Big Data, et ont été appliquées avec succès à des domaines et des tâches variés.

Le système de recommandation de l'entreprise Actualis, développé dans les travaux de Werner (2015), demande l'intervention d'experts du domaine. Ils interviennent d'une part lors de la classification de l'information, selon la sémantique du domaine, et d'autre part dans la recherche d'information pertinentes sur le web, appuyée par des outils de veille. Ces deux aspects de leur travail sont des tâches complexes et fastidieuses compte tenu de la quantité d'information à traiter chaque jour, et pour lesquelles aucun outil spécifique n'est adapté pour faciliter la charge de travail des experts.

Les travaux de thèse développés dans ce manuscrit ont pour objet une nouvelle méthode de valorisation automatique de grands volumes de données issues du web, par une nouvelle approche de recherche et de qualification sémantique de l'information. L'approche présentée dans ce manuscrit tire parti à la fois de méthodes d'apprentissage automatique et des technologies du Big Data, qui permettent de pallier les difficultés associées au traitement de telles données, ainsi que des technologies du web sémantique, qui permettent de rapprocher une information numérique du point de vue humain, extrayant ainsi de la valeur à partir de grands volumes de données. L'objectif scientifique de ces travaux est d'apporter une contribution fondamentale au croisement des domaines de l'apprentissage automatique, de traitement massif des données, et du web sémantique. Un objectif industriel est également un des points clé de ces travaux, soutenus par l'entreprise Actualis SARL. L'approche doit permettre de développer une solution logicielle utilisable dans un cadre industriel, qui fournit une plus-value dans un domaine métier précis, i.e. la veille stratégique. Cette plus-value concerne essentiellement les tâches de recherche et de qualification de l'information, effectuées manuellement par les experts de l'entreprise.

Pour répondre à ces objectifs, nous nous intéressons à l'application de méthodes issues des domaines du web sémantique, de la classification et de la recherche d'information dans un contexte Big Data. Les fondements de ces domaines et leurs rapprochements sont introduits dans les sections suivantes, avant de décrire l'approche envisagée dans ces travaux et développée dans les chapitres suivants.

1.3.1/ PRINCIPALES APPROCHES EN APPRENTISSAGE AUTOMATIQUE

L'objectif de l'apprentissage automatique est de déduire, à partir d'observations empiriques des données, un modèle dit prédictif, ou fonction de prédiction, qui permet de réaliser une tâche trop complexe pour un algorithme classique ou des heuristiques, applicable à de nouvelles données. Il convient de différencier trois grandes "variantes" des tâches d'apprentissage automatique : l'apprentissage supervisé, non supervisé, et l'apprentissage par renforcement. Historiquement, chaque type d'apprentissage correspond à des tâches spécifiques, bien qu'en réalité les limites entre les tâches et les paradigmes d'apprentissage soient plus floues.

L'apprentissage supervisé est naturellement lié à la classification, la régression ou les séries temporelles Tollari (2006), tandis que l'apprentissage non supervisé est lié aux approches de "clustering" (regroupement). L'apprentissage par renforcement est basé sur une amélioration itérative et continue du modèle prédictif, en associant à chaque décision prise par le modèle une récompense adaptée. Le modèle est ensuite adapté de façon itérative en fonction des récompenses obtenues. L'apprentissage par renforcement ne sera que brièvement abordé dans ce manuscrit car il ne fait pas l'objet de ces travaux.

Liu (2007) définit les notions essentielles utilisées dans toute approche en apprentissage automatique. Dans le cas général, un jeu de données d'entraînement est d'abord

considéré, permettant de construire le modèle par extrapolation à partir des observations empiriques. Ensuite, un jeu de données de test, généralement de taille réduite par rapport au jeu d'entraînement, permet de tester l'efficacité du modèle sur des données différentes. En effet, la difficulté principale dans la construction du modèle est de parvenir à déduire des généralisations qui sont correctes en dehors du contexte du jeu de données d'entraînement. Lorsque le modèle est adapté uniquement aux données d'entraînement, les prédictions du modèle seront erronées pour de nouvelles données. Cette problématique est nommée "sur-apprentissage" (overfitting). Une hypothèse essentielle de l'application de méthodes d'apprentissage automatique aux Big Data est que plus les volumes de données sont grands, plus la généralisation des modèles est représentative dans le cas général.

Les travaux présentés traitent de la classification automatique de données web. Les sections suivantes s'intéressent aux méthodes d'apprentissage automatique pour la tâche de classification.

1.3.2/ APPRENTISSAGE AUTOMATIQUE ET CLASSIFICATION

Les présents travaux traitent essentiellement de l'apprentissage automatique pour la tâche de classification. Pour la classification, le jeu de données d'apprentissage est composé d'un ensemble d'"enregistrements", chacun décrit par un ensemble d'**attributs** $|A| = \{A_1, A_2, \dots, A_n\}$ où $|A|$ désigne le nombre d'attributs, i.e. la taille de l'ensemble A . La définition d'"enregistrement" change selon le type de données. En l'occurrence, il s'agit du document unitaire qui compose le jeu de données (page web, image, séquence audio...), également appelé instance, cas, ou vecteur. Dans le cas de traitement de données textuelles, les enregistrements sont communément appelées items ou documents. Les attributs, ou "features", sont également appelées traits ou caractéristiques. Ce dernier terme sera majoritairement employé dans le reste du manuscrit. Le jeu de données est également composé d'un ensemble C , composé des **attributs de classe**, i.e. les attributs utilisés pour la classification des items. Suivant l'approche, les ensembles A et C peuvent être disjoints ou non. Nous verrons dans cette partie que l'approche d'apprentissage de ces travaux considère que C est un sous-ensemble de A . Les attributs de classe sont en ensemble de valeurs discrètes, tel que $|C| = \{C_1, C_2, \dots, C_n\}$, où $|C|$ est le nombre de d'attributs de classe. Les termes étiquette et labels sont également employés pour désigner ces attributs, et seront utilisés dans ce manuscrit de façon interchangeable. Le résultat de l'apprentissage est donc une fonction de prédiction de la forme

$$A_i \rightarrow C_j | A_i \in A | C_j \in C \quad (1.1)$$

i.e. une fonction qui permet de déduire les attributs de classes d'un item à partir de ses caractéristiques. La composition des attributs A_i en entrée et des attributs de classe C_j en sortie du processus dépendent de la méthode de classification.

Toujours pour la tâche de classification, les différences entre les trois variantes d'apprentissage portent sur les propriétés intrinsèques du jeu d'entraînement / de test, en l'occurrence sur les attributs de classe C . En apprentissage supervisé, tous les enregistrements sont définis par un ensemble d'attributs $\{A_1, A_2, \dots, A_n\}$ et d'attributs de classe $\{C_1, C_2, \dots, C_n\}$, i.e tous les enregistrements sont "classifiés" préalablement à la construction du modèle de classification.

La construction d'un jeu de données supervisé de taille conséquente est naturellement

coûteuse en temps, car elle est réalisée à la main par des humains : la qualité du modèle prédictif étant fonction de la qualité du jeu de données, l'annotation manuelle des données d'entraînement par des humains est la solution privilégiée pour sa construction. Pour certaines données, il n'existe tout simplement pas d'attributs de classe, ce qui rend l'annotation manuelle impossible. Aussi, il est possible que l'utilisateur veuille découvrir des motifs dans les données sans savoir à l'avance quelles corrélations/dépendances observer, i.e. quels sont les attributs de classe. Pour ces raisons, l'apprentissage non-supervisé suppose que les classes doivent être définies de manière automatique lors de la création du modèle prédictif. Les approches de regroupement, ou "clustering", sont de fait souvent liées à la notion d'apprentissage non supervisé car elles ne considèrent pas d'attributs de classe, bien que d'autres approches telles que l'extraction de règles d'association puissent également être considérées comme des approches non-supervisées.

L'apprentissage semi-supervisé est une solution hybride, qui suppose pour la tâche de classification un jeu d'entraînement composé à la fois d'enregistrements pré-classifiés et non classifiés. Cette approche permet de palier la difficulté de construction manuelle du jeu d'entraînement, tout en conservant les attributs de classe de l'approche supervisée.

Dans ce manuscrit, les deux premières variantes d'apprentissage automatique exposées en introduction sont abordées, i.e. l'apprentissage supervisé et non-supervisé.

L'approche d'apprentissage développée dans la première partie de ces travaux considère le problème de la classification hiérarchique multi-label dans un contexte non-supervisé, un cas particulier de classification où l'ensemble de classes C est déterminé automatiquement et où des relations hiérarchiques entre les attributs de C sont également déduites à partir des données d'entraînement. La classification effectuée considère enfin la possibilité d'associer plusieurs classes de C à chaque item, i.e. plusieurs chemins de la hiérarchie de classe peuvent être attribués à un item en fonction de ses attributs.

Les chapitres 3 et 4 se focalisent sur un processus d'apprentissage automatique d'un modèle prédictif dans un contexte non supervisé, qui représente la contribution principale de ces travaux. Le chapitre 3 porte sur une approche de regroupement hiérarchique, dont la finalité est d'une part la sélection des attributs de classes (labels) à partir d'un large volume de données, et d'autre part la construction de relations hiérarchiques entre les labels. Le chapitre 4 porte sur une nouvelle méthode statistique pour la création de règles de classification, i.e. les règles d'association qui décrivent la fonction de prédiction $A \rightarrow C$. La combinaison de la hiérarchie de classes et les règles de classification forment le modèle prédictif, intégré dans une base de connaissances commune sous la forme d'une ontologie. Ce modèle prédictif est basé essentiellement sur des méthodes d'apprentissage non-supervisées. Afin de fournir une comparaison exhaustive avec l'état de l'art dans le domaine de la classification, l'apprentissage supervisé est également considéré dans ce manuscrit. Dans cette partie, l'apprentissage supervisé a pour fin la comparaison à l'état de l'art, qui n'a pas été possible dans un contexte non-supervisé. Enfin, l'apprentissage supervisé est également considéré dans la troisième partie qui se concentre sur la mise en place du processus HMC Sémantique dans le cadre de l'application à un domaine métier, afin de répondre aux besoins spécifiques de l'entreprise Actualis SARL.

1.3.3/ RECHERCHE D'INFORMATION SUR LE WEB

La recherche d'information automatique sur le web est un cas particulier du domaine de la Recherche d'Information. La Recherche d'Information étudie les méthodes permet-

tant à un utilisateur de trouver une information pertinente parmi une large collection de documents texte⁷ Manning et al. (2008) Liu (2007).

L'objectif est de répondre à des requêtes utilisateurs qui souhaitent retrouver un ou plusieurs items. Les items les plus pertinents pour une requête donnée sont alors retournés à l'utilisateur, et un classement des items selon leur pertinence peut être effectué. Le résultat de la requête est l'ensemble des items pertinents. La majorité des systèmes de recherche attribuent un score de pertinence aux résultats de la requête.

Les moteurs de recherche commerciaux tels que Google, Bing ou Yahoo permettent une recherche efficiente de l'information, basée sur une indexation de l'information. Ces moteurs recensent plusieurs milliards de pages, de qualité et de pertinence très hétérogène, ce qui implique des compromis en terme d'exhaustivité et de finesse des résultats. Ces moteurs sont donc des outils efficaces mais sont peu spécialisés.

Dans un contexte de veille stratégique tel que le contexte de l'entreprise Actualis, des problématiques spécifiques supplémentaires doivent être prises en compte. Les sources d'informations utilisées sont variées, et leur référencement par les moteurs de recherche commerciaux est hétérogène. La mise à jour des sources d'information peut varier grandement d'un source à une autre. Afin d'assurer la véracité d'une information, le croisement d'information provenant de sources multiples est parfois nécessaire, ce qui est chronophage avec des outils standards. Enfin, la découverte de nouvelles sources d'informations fiables est une tâche complexe et difficile à automatiser.

Les moteurs de recherche généralisés ne peuvent pas répondre à ces tâches, spécifiques à un domaine métier précis. Des outils spécialisés, notamment dans le contexte de la veille stratégique, ont été développés afin de palier les limitations des moteurs généralistes. Ces outils répondent à certaines de ces problématiques, mais il est parfois difficile d'adapter ces outils au processus de production d'une entreprise. Des outils existants de veille stratégique, tels que Digimind⁸, Sindup⁹, Website Watcher¹⁰, Pressedd¹¹ sont des outils puissants et personnalisables, et fournissent une solution à certaines de ces problématiques. En revanche, le croisement d'information et la découverte de nouvelles sources d'information demeurent des tâches non résolues.

Aussi, bien qu'il soit possible de paramétriser ces outils, il ne permettent pas d'intégrer des connaissances métiers spécifiques, représentées par exemple dans un thesaurus.

Certains outils reposent sur l'utilisation de robots d'indexation spécialisés, i.e. des agents autonomes qui parcourrent le web ou une partie du web à la recherche d'informations spécifiques, pour répondre aux problématiques de croisement et de découverte d'information. Google News¹² effectue par exemple cette tâche en regroupant des informations similaires issues de sites de presse nationale et internationale. La dimension du web et sa structure impliquent des importants coûts en ressources et en temps pour la recherche d'information avec ce type d'outils. Ainsi, diverses stratégies de parcours du web, basées à la fois sur sa structure, i.e. le graphe de liens hypertexte, et le contenu des pages et leur analyse ont été employées pour améliorer ces outils. L'association de méthodes issues de l'apprentissage machine et de la sémantique ont également été envisagées. Ces

7. Dans le cas de recherche d'information sur le web, l'unité de recherche est la page web

8. <http://www.digimind.com/fr>

9. <http://www.sindup.com/>

10. <http://www.website-watcher.fr/>

11. <http://www.pressedd.fr/>

12. <https://news.google.fr/>

méthodes sont décrites de façon exhaustives dans le chapitre 6.

1.3.4/ APPROCHES SÉMANTIQUES

L'apprentissage automatique permet de générer des modèles prédictifs, à partir d'analyses statistiques des données fondées sur des modèles mathématiques, notamment des modèles probabilistes. Ces approches statistiques ont cependant pour inconvénient de ne pas prendre en compte les propriétés sémantiques des données. Ces informations sémantiques sont pourtant des informations de valeur pour un utilisateur humain. Aussi, il est difficile de rapprocher un modèle établit de façon purement statistique d'un modèle représentatif du raisonnement humain.

Par opposition aux approches statistiques pour l'extraction et la découverte de connaissances à partir de données numériques, les approches symboliques sont basées sur des connaissances pré-établies et notamment les technologies du web sémantique. L'utilisation des technologies du web sémantique dans la construction d'un modèle prédictif permettent de présenter à l'utilisateur une vue compréhensible du processus, ce qui augmente la confiance de l'utilisateur dans le système. Ces approches peuvent être combinées à des méthodes de fouille de données et d'apprentissage automatique pour générer automatiquement de nouvelles connaissances, en prenant en compte l'aspect sémantique des données, et leur lien avec des connaissances établies.

Ristoski et al. (2016) décrit trois cas d'utilisation où les domaines de la fouille de données et du web sémantique se recoupent :

- Utiliser des ontologies et le Linked Data pour supporter et améliorer un processus d'extraction de connaissances.
- Utiliser des méthodes de fouille de données pour extraire des connaissances à partir du web sémantique (Semantic web mining).
- Utiliser des méthodes d'apprentissage automatique pour générer des données sémantiques et améliorer le Linked Data

Dans un contexte de traitement massif de données, les bénéfices du web sémantique sont réels. L'extraction d'informations de valeur dans des masses de données peut être améliorée en tirant partie de bases de connaissances et/ou des moteurs d'inférence. De nouveaux champs de recherche sont issus de l'application d'approches sémantiques au domaine du Big Data. Sheth (2014) nomme ce champ de recherche le Smart Data, où la combinaison efficiente de données statistiques (modèles mathématiques, apprentissage machine...) et symboliques (métadonnées, analyse de contenu, annotation et raisonnement sémantique...) permet d'extraire la valeur du Big Data et d'associer cette valeur à des connaissances établies, qu'elles soient générales ou spécifiques à un domaine.

Enfin, dans le cadre de la recherche d'information sur le web, les technologies du web sémantique ont été employées dans diverses applications, afin d'améliorer la qualification de l'information et ainsi assurer la pertinence des résultats. Cette amélioration impacte indirectement la performance des crawlers web, qui peuvent alors utiliser des bases connaissances pour une recherche efficiente et spécifique de l'information Batsakis et al. (2009) Kassim et al. (2009).

1.3.5/ VERROUS SCIENTIFIQUES

Les travaux présentés dans ce manuscrit s'articulent autour de quatre verrous scientifiques :

- La construction non-supervisée d'un modèle prédictif dans un contexte Big Data, pour la classification de textes en langage naturel
- L'utilisation et le passage à l'échelle du raisonnement logique en contexte Big Data, dans le cadre d'une classification basée sur une description sémantique de l'information
- L'intégration d'un modèle prédictif pour la découverte de sources d'information et le croisement d'information sur le web
- L'adaptation d'un modèle prédictif, décrit par une ontologie, aux évolutions des données dans un contexte web non structuré

Les deux premiers points concernent les chapitres 2 à 4, tandis que les deux derniers concernent les chapitres 5 et 6.

1.4/ CONTRIBUTIONS

Les contributions de ce travail sont triples.

Premièrement, une architecture d'analyse de données appelée HMC Sémantique, ou Classification Hiérarchique Multi-Label¹³ est décrite. L'architecture étend les travaux de Werner (2015), et pallie les limitations techniques de celle-ci. L'architecture consiste en l'apprentissage d'un modèle de classification à partir de grands volumes de données, et l'exploitation de ce modèle pour réaliser la classification de nouvelles données. Ces deux aspects du processus sont évalués, de façon exhaustive dans les chapitres 3 et 4. Les résultats obtenus démontrent premièrement la faisabilité de l'approche dans un contexte Big Data. La qualité du modèle générée est évaluée d'une part, par une évaluation lexicale pour l'apprentissage d'une hiérarchie d'attributs cibles en contexte non supervisé, et d'autre part par l'analyse des performances proches de l'état de l'art pour la tâche de classification. L'évaluation lexicale montre le potentiel de l'approche pour l'extraction d'attributs et d'attributs de classe qualitatifs en contexte non supervisé. L'évaluation de la classification montre des résultats proches de l'état de l'art pour la tâche de classification multi-label en contexte supervisé, et suggère de bonnes performances en contexte non supervisé, bien qu'il n'ait pas été possible d'effectuer une comparaison à l'état de l'art. Cette architecture est l'apport principal de ces travaux, puisque les deux autres apports reposent en partie sur celle-ci. Cette architecture a été développée en collaboration avec Rafael Peixoto, avec qui j'ai eu la chance de travailler sur cette partie du projet de recherche. En particulier, sa participation à la formalisation de l'approche a été essentielle au développement et aux publications issues du projet, notamment Hassan et al. (2015), Peixoto et al. (2015b) Peixoto et al. (2015c) Peixoto et al. (2016b) Peixoto et al. (2016c). Je le remercie également pour son esprit critique, sa rigueur scientifique, et globalement pour les échanges productifs que nous avons eu vis-à-vis de l'approche et de l'évaluation préliminaire exposée en première partie de ce document. Les chapitres 2 à 4 reflètent cette collaboration, une partie de leur contenu est donc similaire aux travaux exposés dans les travaux de thèse de Rafael Peixoto.

13. Multi-étiquette

La seconde contribution est une architecture de recherche et de croisement d'information sur le web, basée sur une qualification sémantique de l'information issue de l'architecture HMC Sémantique. Cette seconde architecture tente de répondre aux verrous scientifiques établis dans le domaine de la recherche d'information sur le web (section précédente), et apporte des éléments de réponse via la combinaison des méthodes d'apprentissage machine et des technologies du web sémantique pour la recherche d'information sur le web. Elle reflète les travaux présentés dans Hassan et al. (2017a) et Hassan et al. (2017b). L'architecture est évaluée pour la tâche de croisement d'information, et comparée à une approche de l'état de l'art. Cette évaluation montre les avantages de l'approche, qui permet une adaptation continue de la recherche d'information, améliorant ainsi la performance comparativement à l'état de l'art.

La dernière contribution concerne la mise en application de l'approche, dans un contexte industriel réel. Cette application est une des motivations initiales des travaux, soutenus par l'entreprise Actualis SARL, partenaire et pour partie financeur de ce projet de recherche. Au delà du développement d'une preuve de concept et de la résolution des verrous scientifiques sur lesquels ces travaux se sont concentrés, les problématiques réelles de mise en application ont toujours été au cœur du projet de recherche. La réalisation d'une solution logicielle fiable, qui apporte une plus-value dans un contexte métier précis est un des objectifs de ces travaux. Cette application tente de résoudre des problématiques concrètes au sein de l'entreprise, notamment dans le but d'améliorer certaines parties critiques du travail effectué par les documentalistes. Le volume de données à traiter chaque jour par les documentalistes est important, ce qui induit une possible perte d'informations à destination de leur clients.

1.5/ COMPOSITION DU DOCUMENT

Le reste de ce document est divisé en 3 parties. La partie 2 décrit l'architecture de classification hiérarchique multi-étiquette sémantique, nommée SHMC, mentionnée précédemment. Cette partie est divisée en trois chapitres. Le chapitre 2 décrit l'état de l'art du domaine de la classification hiérarchique multi-étiquette, et s'intéresse en particulier au croisement de ce domaine avec les domaines du Big Data et du web sémantique. L'architecture SHMC est ensuite décrite de façon exhaustive dans les chapitres 3 et 4. Le chapitre 3 s'intéresse à une méthode d'apprentissage automatique non-supervisé d'une taxonomie dans un contexte Big Data. Le chapitre 4 décrit notre approche de classification sémantique de données, qui comprend à la fois la génération et l'application de règles de classification pour des grands volumes de données.

La partie 3 du document décrit une architecture de recherche d'information sur le web, basée sur un crawler web sémantique adaptatif. Le chapitre 5 décrit l'état de l'art de la recherche d'information sur le web. Le chapitre 6 présente la proposition d'architecture de recherche d'information sur le web, nommée SEMXDM. Le chapitre 7 décrit la mise en place des deux architectures SHMC et SEMXDM au sein de l'entreprise Actualis. Les spécificités de mise en production de la solution, et l'intégration de celle-ci dans le processus de production de l'entreprise y sont décris.

La dernière partie du document conclue sur les contributions de ces travaux, et donne des pistes de réflexion sur les questions en suspens, et les éventuels travaux futurs.



CLASSIFICATION AUTOMATIQUE DE DONNÉES DANS UN CONTEXTE BIG DATA

2

ETAT DE L'ART : CLASSIFICATION AUTOMATIQUE BASÉE SUR LA SÉMANTIQUE

La section précédente a exposé les bases de construction d'un modèle de classification, où des instances des données, ou items, sont associés à un attribut d'un ensemble de classe. Les approches permettant de construire le modèle et d'effectuer sont variées, mais peuvent être regroupées en deux catégories : la classification plate, et la classification hiérarchique. Dans les deux cas, si la classification associe plusieurs classes à une même instance, il s'agit de classification multi-étiquette (multi-label). Ce chapitre décrit dans une première section les approches existantes pour chaque cas, et en étudie les limitations.

Les approches de classification basées sur les ontologies font l'hypothèse que les performances des méthodes de classification peuvent être améliorées via l'utilisation des technologies du web sémantique. Cet aspect est essentiel dans la majorité des domaines, car la valeur ajoutée de la classification réside dans l'adéquation avec les connaissances spécifiques au domaine, et passe donc par l'utilisation des technologies du web sémantique.

Deux cas d'utilisation des technologies du web sémantique ont été identifiés pour la tâche de classification :

- Représenter les connaissances métier, et intégrer les règles de classification au sein d'une ontologie de domaine. L'ontologie (Studer et al., 1998b) joue un rôle déterminant dans la définition des termes utilisés pour représenter la connaissance, réduisant ainsi l'écart entre la base de connaissances et l'utilisateur spécialisé dans le domaine.
- Utiliser des moteurs d'inférence (raisonnement sémantique) pour améliorer le processus de classification.

L'apport des technologies du web sémantique au domaine de la classification et leurs limitations pour ces deux cas d'utilisation sont donc étudiées dans une seconde section. L'annexe A définit les notions et les principes des logiques de description, nécessaires à la compréhension de la second section.

Une proposition d'architecture de classification sémantique de l'information est finalement formulée, afin de répondre aux limitations de l'état de l'art.

2.1/ CLASSIFICATION HIÉRARCHIQUE MULTI-ÉTIQUETTE

La tâche de classification consiste à associer un ou plusieurs attributs de classe, ou labels, à un item défini par un ensemble d'attributs ou caractéristiques (features). Lorsque les attributs de classe sont indépendants, la classification est dite "plate"¹. Dans la plupart des cas réels de classification, les labels sont dépendant les uns des autres et ne sont pas traités séparément : ils peuvent être organisés en groupes, et notamment agencés en fonction de leur généralité/specificité. Le résultat de cet agencement est une structure hiérarchique Bi et al. (2011) Cerri et al. (2014) Tsoumakas et al. (2009) Silla Jr et al. (2011).

La classification hiérarchique multi-label (hierarchical multi-label classification) est la combinaison de la classification multi-label et de la classification hiérarchique(Bi et al. (2011) Santos et al. (2009) Cerri et al. (2014)). Par définition, la classification hiérarchique multi-étiquette bénéficie des avantages de la classification hiérarchique et de la classification multi-étiquette. Dans cette approche, plusieurs chemins de la hiérarchie de classes peuvent être attribués à chaque item, qui peut donc appartenir à différentes classes d'un même niveau.

On différencie deux types de structures hiérarchiques pour l'organisation des labels : les arbres et les graphes acycliques Cerri et al. (2014). La figure 2.1 décrit les trois types de structures et leur correspondance avec les méthodes existantes de classification.

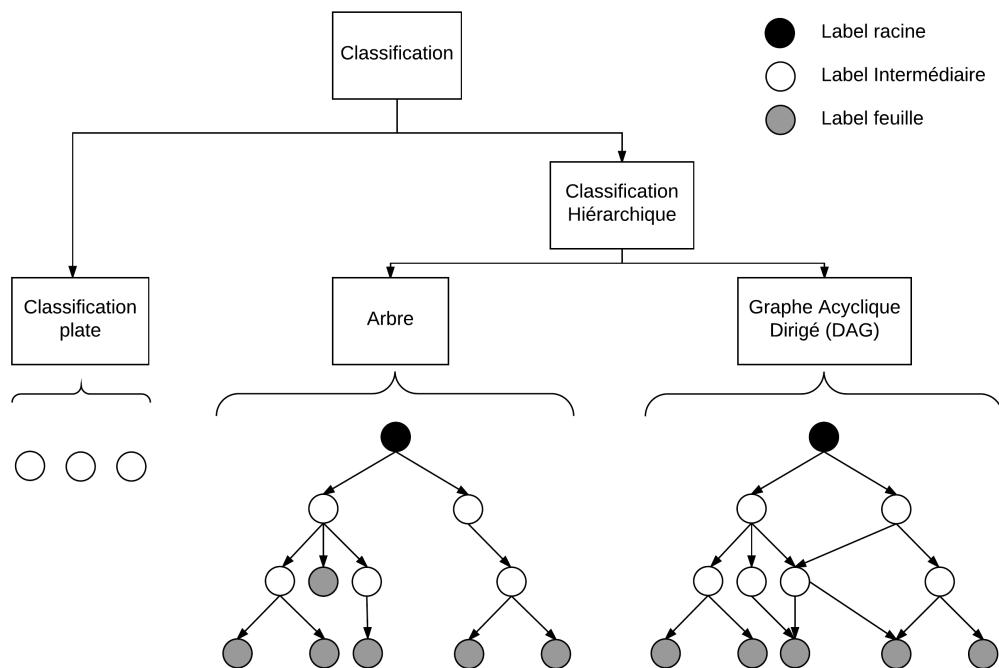


FIGURE 2.1 – Types de structure pour la classification

La figure 2.2 montre un exemple de classification d'instances pour les deux types de structures hiérarchiques.

L'appartenance (classification) d'une instance à une classe induit l'appartenance aux

1. "Flat classification"

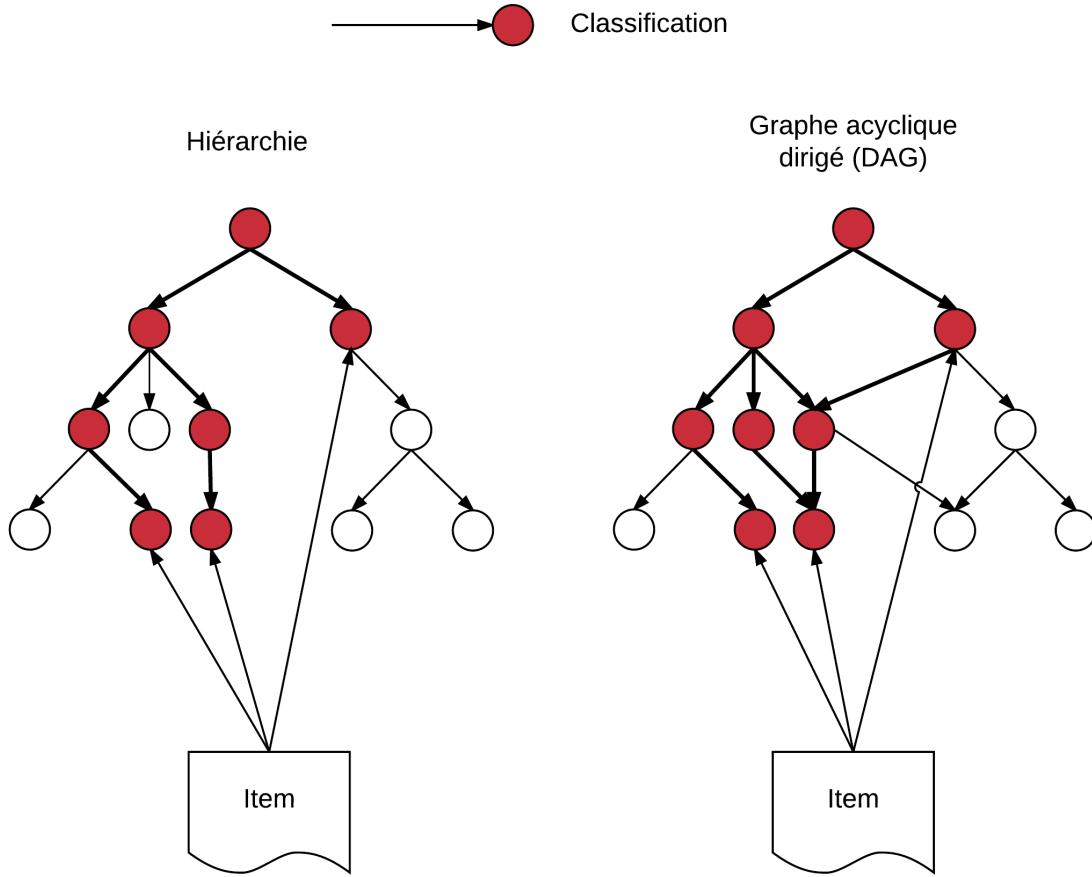


FIGURE 2.2 – Classification d'un item selon une hiérarchie et un DAG

classes parentes, i.e. assigner une classe à un item revient à assigner le chemin de la classe jusqu'au nœud racine.

On considère généralement 2 types d'approches pour résoudre le problème de classification hiérarchique, la classification locale (1) et globale (2). La différence entre classification hiérarchique locale (1) et globale (2) est généralement définie comme suit (Santos et al. (2009) Silla Jr et al. (2011) Cerri et al. (2014)) :

- L'approche locale (1), où des algorithmes de classification conventionnels sont utilisés, et définissent une fonction de prédiction différente pour chaque classe. Pour créer la fonction de prédiction associée à chaque classe (nœud du graphe), le jeu d'entraînement ne considère que les informations liées directement à la classe (nœuds parents / enfants). Divers algorithmes tels que "Local Classifier per Node" (LCN), "Local Classifier per Parent Node" (LCPN), et "Local Classifier per Level" (LCL) peuvent directement être appliqués pour générer une fonction de prédiction par nœud Cerri et al. (2014).
- L'approche globale (2) ne considère qu'une fonction de prédiction pour l'ensemble des labels. Contrairement à la classification locale, les algorithmes conventionnels de classification sont alors difficilement applicables.

L'algorithme Binary Relevance (BR) Tsoumakas et al. (2007) décrit une approche de classification locale, où un classifieur binaire est défini pour chaque label (attribut de classe). Le classifieur est défini par un vecteur binaire des attributs. Chaque classifieur

détermine la pertinence des labels pour chaque item. Une liste de labels est attribuée à chaque item en sortie. La probabilité de pertinence est ensuite utilisée pour ordonner les labels de l'item par pertinence.

Classifier chaining (CC) Read et al. (2011) définit des classifieurs binaires de façon similaire à l'approche de Tsoumakas et al. (2007). Q classifieurs distincts sont définis tels qu'ils forment une chaîne de classifieurs. L'objectif est de représenter les corrélations entre les labels par la chaîne. Le $i^{\text{ème}}$ classifieur de la chaîne définit la classification pour le label $\lambda_i \in L$ où $(1 \leq i \leq Q)$. Contrairement à l'approche de Tsoumakas et al. (2007), le résultat de la classification d'un label dépend du résultat de la classification des éléments précédents de la chaîne. Le vecteur binaire $\lambda_i + 1$ est récursivement étendu par le résultat de la classification du label λ_i . La prédiction et l'ordonnancement des labels en sortie est effectuée de la même manière que dans l'approche de Tsoumakas et al. (2007). L'approche classifier chaining améliore les performances comparativement à l'approche Tsoumakas et al. (2007), et conserve une complexité faible.

Sucar et al. (2014) définit l'approche Bayesian chain classifier (BCC). Pour chaque label L_i dans la chaîne, un classifieur bayésien est défini. TNBCC (tree naïve bayes chain classifier) et Path-BCC (Path bayes chain classifier) sont deux approches dérivées de l'approche BCC. Dans les deux approches, une structure de dépendance (par exemple un arbre) des labels est d'abord construite. Dans Path-BCC, tous les noeuds indirectement parents d'un label sont ajoutés en tant qu'attributs du vecteur du label L_i , i.e. les noeuds du chemin entre le label racine et le label L_i . Dans TNBCC, le vecteur est étendu uniquement avec le label parent.

Hierarchy Of Multi-label Classifiers (HOMER) Tsoumakas et al. (2008) est un algorithme de classification multi-label, créé spécifiquement pour de larges volumes de données. HOMER construit une hiérarchie de classifieurs multi-label, où chaque classifieur est dédié à un nombre restreint de labels. Le processus permet ainsi de monter en charge par distribution de la tâche de classification à plusieurs classifieurs.

Multi-label k-nearest neighbors (ML-kNN) Zhang et al. (2007) est une extension de l'algorithme de des k-voisins (kNN). Premièrement, pour chaque item $item_i$, les k-voisins sont identifiés en fonction de leurs attributs. Ensuite, le principe de maximum a posteriori est utilisé pour déterminer les attributs de classe de l'item $item_i$, en fonction des propriétés statistiques des attributs de classe de ses voisins, i.e la proportion de ses voisins qui correspondent à un label.

Les algorithmes Random forest of predictive clustering trees (RF-PCT) et Random forest of ML-C4.5 (RFML-C4.5) Kocev et al. (2007) sont des méthodes ensemblistes² qui utilisent respectivement les arbres PCTs et ML-C4.5 comme classifieur de base. Plusieurs classifieurs de base sont utilisés pour déterminer la pertinence des labels. Les prédictions individuelles de chaque classifieur sont ensuite combinées via un paradigme de vote, par exemple la majorité des résultats des classifieurs ou par distribution de probabilité.

Griffiths et al. (2004) se base sur le modèle génératif probabiliste Latent Dirichlet Allocation (LDA) Blei et al. (2003), pour déduire le thème de documents à partir de leurs

2. Les méthodes ensemblistes supposent la division d'un problème complexe en un agrégat de problèmes plus simples. Pour la tâche de classification, l'objectif est de diviser la tâche de classification globale en la délégant à différents classifieurs plus spécifiques. Ce type de méthodes permet de palier certaines limitations dues à l'utilisation d'un seul classifieur. La plupart des méthodes ensemblistes sont applicables à un éventail de classifieurs de base <http://blog.octo.com/les-methodes-ensemblistes-pour-algorithmes-de-machine-learning/>

attributs (termes). L'approche LDA nécessite d'approximer les paramètres d'intérêt de la distribution de dirichlet, ce qui pose un problème calculatoire lorsque l'espace des attributs est de taille importante. L'algorithme Collapsed Gibbs Sampling (CGS) de Griffiths et al. (2004) utilise alors une procédure d'échantillonnage (Markov Chain Monte Carlo, MCMC) pour définir la distribution de probabilité.

L'algorithme CGS_p Papanikolaou et al. (2015) est une variante de l'approche de Griffiths et al. (2004), basée sur une la procédure CVB0 (Collapsed Variational Bayesian inference update procedure) Asuncion et al. (2009). Les auteurs montrent que la procédure CVB0 améliore la performance de l'approche LDA pour la tâche de classification multi-label.

La plupart des approches en apprentissage automatique ne considèrent pas de dépendance (relations) entre les items, i.e. la classification est dite indépendante en rapport aux items. Dans certains cas réels, l'interdépendance des items peut également être considérée. Il s'agit alors de classification collective. Un exemple de cas d'utilisation est la classification de pages web, où chaque item est lié aux autres par ses liens hypertextes. Les liens entre les pages peuvent être utilisés conjointement avec les termes extraits des pages pour classer les pages. Les systèmes collaboratifs basés sur un graphe d'utilisateurs peuvent également tirer partie de la classification collective. Kong et al. (2011) présente une approche combinant classification collective et classification multi-label. Cette approche ne considère cependant pas de structure hiérarchique entre les classes.

L'état de l'art de la classification multi-étiquette (hiérarchique) est éprouvé. Les principales approches, décrites ci-dessus, sont basées sur des modèles probabilistes, notamment bayésiens, et des variantes de ceux-ci. Ces approches bien que performantes sont purement statistiques, et ne tirent pas partie des métadonnées ou de la sémantique éventuellement associée aux données. La section suivante fait état des approches de l'état de l'art qui se concentrent sur ce dernier aspect de la classification, par l'intégration de d'ontologies dans le processus de classification. L'ontologie Studer et al. (1998a) joue un rôle déterminant dans la définition des termes utilisés pour représenter la connaissance. L'exploitation de bases de connaissances dans un processus de classification est un moyen de rapprocher le processus de classification et les connaissances de l'utilisateur humain.

2.2/ APPROCHES BASÉES SUR LES ONTOLOGIES

Les sous-sections suivantes présentent l'état de la littérature pour les cas d'utilisation des ontologies et plus largement des technologies du web sémantique dans les systèmes de classification. Deux cas d'utilisation ont été identifiés dans l'état de l'art : la représentation des connaissances, et le mécanisme d'inférence appliqué à la tâche de classification.

2.2.1/ ONTOLOGIES ET CLASSIFICATION

Divers travaux de la littérature traitent de l'utilisation des ontologies pour la classification.

Elberrichi et al. (2012) présente une méthode en deux étapes pour améliorer la classification de documents médicaux (MeSH - Medical Subject Headings) :

- Les documents médicaux sont représentés par un vecteur sémantique. Les concepts de l'ontologie sont en premier lieu définis à partir de termes médicaux

pertinents. Un algorithme de désambiguïsation et de matching³ permet d'extraire les termes pertinents des documents médicaux, puis l'ontologie permet de retrouver les concepts correspondants qui contient un contexte syntaxique (termes associés au concept).

- Le vecteur sémantique est ensuite enrichi par addition des concepts parents (hyponymes⁴)

Leurs résultats montrent que l'utilisation d'une ontologie de domaine permet d'améliorer la performance de la méthode de classification des documents médicaux.

Aparicio et al. (2013) propose une méthode de classification semi-supervisée assistée par une ontologie. Dans ces travaux l'ontologie est utilisée pour fournir un ensemble de termes qui contextualisent une classe. A partir de ces termes les données non classées peuvent être annotées automatiquement via un ensemble de règles.

Galinina et al. (2013) définit un système de classification basé sur deux ontologies :

- une ontologie de domaine, indépendante de la méthode de classification
 - une ontologie dédiée à la méthode de classification basée sur un arbre de décision
- Costa et al. (2013) décrit un framework⁵ de description de sources d'informations (documents, pages web). Chaque source d'information est représentée par un vecteur sémantique. Les vecteurs sont composés d'un ensemble de concepts d'une l'ontologie, puis étendus via les relations entre les concepts de l'ontologie (de la même façon qu'avec des vecteurs simples tels que des vecteurs de termes).

Johnson et al. (2010) propose une méthode qui comprend une ontologie qui décrit les connaissances, et inclue les experts du domaine dans le processus de décision. Cette approche a pour objectif l'amélioration continue de l'ontologie, en prenant en compte le retour des experts du domaine. L'ontologie n'est cependant pas utilisée pour classer automatiquement de nouveaux items.

Dans Vogrincic et al. (2011), une méthode non supervisée d'apprentissage d'une ontologie à partir d'un corpus d'articles économiques est présentée. Un ensemble d'approches de classification multi-label sont ensuite appliquées pour catégoriser les articles selon les concepts de l'ontologie⁶. Les auteurs montrent selon différents critères d'évaluation, la plupart des approches sont pertinentes pour la tâche de classification basée sur une ontologie générée automatiquement.

Dans Garrido et al. (2012) un thésaurus pré-existant est utilisé à la fois pour décrire les connaissances du domaine et pour classer de nouveaux items. L'approche est comparée à une classification par une machine à vecteurs de support (SVM). Les auteurs montrent que la performance de l'approche sémantique est supérieure à l'approche SVM. Aussi, les auteurs insistent sur les avantages d'une approche sémantique par rapport à une approche statistique telle que SVM. Le modèle SVM doit être mis à jour régulièrement pour être précis, comparativement à l'approche sémantique. La classification assistée par un thesaurus est plus intuitive pour l'utilisateur.

3. Appariement

4. Terme général dont le sens inclut celui d'autres termes (ses hyponymes) des concepts qui le composent <http://www.linternaute.com/dictionnaire/fr/definition/hyperonyme/>

5. "Un framework est, comme son nom l'indique en anglais, un "cadre de travail". L'objectif d'un framework est généralement de simplifier le travail des développeurs informatiques (les codeurs si vous préférez), en leur offrant une architecture "prête à l'emploi" et qui leur permette de ne pas repartir de zéro à chaque nouveau projet." Source : <https://www.1min30.com/dictionnaire-du-web/framework>

6. On retrouve parmi les approches sélectionnées certaines des approches classiques décrites dans la section 2.1

La plupart des travaux de la littérature se concentrent sur l'amélioration du processus de classification en utilisant les ontologies, ce qui permet d'améliorer la description des items, et la performance de la classification. En revanche, ils ne tirent pas avantage des capacités des raisonneurs sémantiques pour classer automatiquement des items (Peixoto et al. (2016b)). Au delà de l'aspect descriptif de haut niveau sémantique que permet l'usage d'une ontologie, l'usage de raisonneurs sémantiques pour la tâche de classification a également été l'objet de recherches antérieures. Or, l'utilisation de raisonneurs pour la tâche de classification peut améliorer les performances du processus de classification (Moller et al. (2009)). La sous-section suivante étudie les approches de classification qui tirent partie du mécanisme d'inférence.

2.2.2/ RAISONNEMENT SÉMANTIQUE POUR LA CLASSIFICATION

Dans Fang et al. (2010), un moteur d'inférence effectue la classification de documents textes en utilisant différentes mesures de similarité. La méthode de classification est composée de 4 étapes :

- Un vecteur pondéré de termes représente chaque item. Les catégories de documents (classes) sont définies dans l'ontologie et agencées de façon hiérarchique.
- Un moteur d'inférence détermine tous les concepts de plus bas niveau (feuille) dans l'ontologie pour un item en fonction du vecteur de termes.
- Une étape de classification binaire définie ensuite pour chaque item l'ensemble des catégories auxquelles il correspond. Le résultat est en ensemble de paires <item,catégorie>.
- La similarité sémantique entre les items et les catégories est ensuite calculée par approximation en utilisant la distance Google Normalisée (Normalized Google Distance) Cilibarsi et al. (2007).

Ben-David et al. (2010) utilise les ontologies et un raisonneur pour représenter le modèle prédictif, et effectuer la classification. Une ontologie de domaine décrit les entités sur lesquelles sont basées la classification, pour effectuer et représenter la classification. Cette méthode permet de classer des items de formats et de sources variés, selon un modèle de classification unique. L'inférence est utilisée pour améliorer le résultat de la classification, mais la classification ne repose pas uniquement sur le moteur d'inférence.

2.3/ PROPOSITION : PROCESSUS DE "HMC SÉMANTIQUE"

L'intérêt de l'apprentissage automatique supporté par une approche sémantique est soutenu par divers travaux, exposés dans la section précédente. Les approches sémantiques peuvent améliorer la performance de la classification d'une part, et d'autre part permettent à l'utilisateur d'interagir avec la système de classification, et d'observer des éléments du système de façon compréhensible et intuitive. Dans le cas de l'apprentissage automatique d'ontologies à partir des données, les ontologies générées sont une vue sémantique des données basée sur des propriétés statistiques. Elles représentent la valeur extraite des données, et peuvent être réutilisées comme modèle pour des tâches ultérieures de la même façon qu'un modèle statistique standard.

Maedche et al. (2001) définit les problématiques associées à la construction automatique d'ontologies : la complexité de construction, le temps de construction, et la confiance

dans l'ontologie résultante du processus (i.e. sa qualité). Ces problématiques sont toujours présentes aujourd'hui dans un contexte Big Data. Dans les travaux de recherche précédent le projet actuel, Werner et al. (2014) utilise une ontologie de domaine dans un processus de classification de nouvelles économiques. L'ontologie est basée une hiérarchie de termes qui décrit le domaine et permet d'effectuer la classification. L'ontologie est enrichie avec les documents à classifier (instances au niveau Abox), et compare l'utilisation de raisonneurs basé sur la logique de description (DL) pour la classification, notamment Pellet (Sirin et al. (2007)), FaCT++ (Tsarkov et al. (2006)), et Hermit (Shearer et al. (2009)). Dans le cadre d'une architecture d'une classification sémantique de l'information appliquée à la classification d'articles de presse, l'approche de Werner et al. (2014) montre l'intérêt d'une telle méthode. En revanche, Werner et al. (2014) montre que les raisonneurs sémantiques basés sur la logique de description (DL) ne sont pas adaptés pour la montée en charge, et sont donc inadaptés à au contexte Big Data. Des raisonneurs à l'échelle du web (Urbani (2013)) utilisent un raisonnement basé sur des règles OWL Horst, et permettent un passage à l'échelle par parallélisation et distribution de la charge de calcul sur des clusters de machines. La montée en charge peut ainsi s'effectuer via des techniques telles que Map-Reduce. Ce type d'approche peut permettre de palier cette limitation.

L'architecture de classification hiérarchique multi-label, ou SHMC, se base sur ces méthodes et l'approche de Werner et al. (2014) pour palier les limitations de l'état de l'art, en particulier la construction d'un modèle de classification dans un contexte Big Data, et l'application du mécanisme d'inférence pour effectuer la classification selon un modèle décrit par une base de connaissances. Notre approche consiste à utiliser le raisonnement sur des règles de Horn, dont l'expressivité est limitée mais permet un passage à l'échelle dans un contexte Big Data, tout en conservant des performances élevées pour la classification. L'architecture est décrite de façon exhaustive dans les chapitres 3 et 4.

3

APPRENTISSAGE DE TAXONOMIE NON SUPERVISÉE

Le chapitre précédent a identifié les avantages et les limites des approches de classification supportées par une ontologie. D'une part, ce type d'approche permet d'effectuer une classification proche des connaissances métiers, dont la valeur est conséquemment importante pour l'utilisateur du domaine. Aussi, les moteurs d'inférence peuvent être utilisés pour d'améliorer la performance du processus de classification.

Les travaux de Werner et al. (2014) ont cependant montré les limites de ces approches en terme de passage à l'échelle. La contribution de ces travaux est une architecture de classification sémantique. Cette architecture tente de pallier les limitations de l'état de l'art. Elle s'appuie sur les technologies du web sémantique pour représenter l'information et effectuer la classification.

Elle s'inscrit dans un contexte Big Data, et tente de répondre à deux problématiques. Premièrement, l'apprentissage automatique non-supervisé d'un modèle de classification à partir de grands volumes de données, qui tire partie de ce type de données pour générer un modèle de classification à la fois efficient et statistiquement pertinent. Deuxièmement, l'utilisation de ce modèle de classification dans un contexte Big Data. L'approche doit permettre classer les nouvelles données via un moteur inférence en respectant les spécificités sémantiques du modèle, tout en palliant les problématiques de passage à l'échelle définies par Werner et al. (2014).

Dans les deux chapitres suivants, le processus d'apprentissage d'une ontologie (chapitre 3) et de classification via l'inférence sur des règles de l'ontologie est décrit et évalué (chapitre 4).

Ce chapitre s'intéresse à deux étapes du processus d'extraction de connaissances à partir des données : l'extraction des attributs (termes) et des attributs de classe (labels), et la construction automatique d'une hiérarchie de labels en contexte non supervisé. Ces étapes du processus sont évaluées en fin de chapitre. L'évaluation porte sur la qualité d'extraction des attributs, et la qualité de la hiérarchie générée. La section suivante décrit d'abord globalement l'architecture. Les éléments de l'architecture dont ce chapitre fait l'objet sont décrits dans les sections suivantes.

3.1/ ARCHITECTURE "SHMC"

Le processus nommé HMC¹ Sémantique, ou SHMC, est composé de 5 étapes (Figure 3.1) :

- **L'étape d'indexation** extrait les termes à partir des données d'entraînement (a), i.e. les items non classés, et crée un index inversé (b) des items.
- **L'étape de vectorisation** calcule les vecteurs de fréquence des items indexés, et une matrice de cooccurrence (c) à partir des vecteurs (b). Cette matrice recense l'ensemble des termes contenus dans l'index, et l'ensemble des cooccurrences entre chaque paire de termes, i.e. le nombre d'items où deux termes apparaissent simultanément.
- **L'étape de hiérarchisation** créé la hiérarchie de labels (d) à partir de la matrice de cooccurrence (c).
- **L'étape de résolution** créé les règles de classification (e) à partir de la matrice de cooccurrence (c).
- **L'étape de réalisation** peuple l'ontologie avec les nouveaux item (f). Un moteur de raisonnement permet d'inférer les labels les plus spécifiques pour chaque item à partir des règles de classification et la hiérarchie de labels. Le résultat de la classification est composé de l'ensemble des relations liant un item aux labels labels inférés (g).

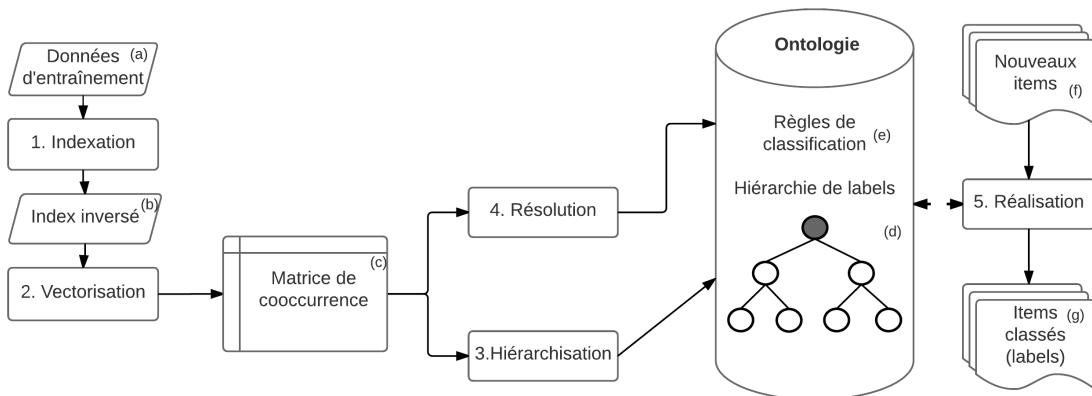


FIGURE 3.1 – Le processus HMC Sémantique

Le modèle prédictif du processus SHMC est intégré dans une ontologie noyau², d'expressivité \mathcal{ALCI} (voir annexe A, dont les axiomes en logique de description sont présentés dans le tableau 3.1).

La classe *Item* représente les items à classer. Cette classe est peuplée avec les items (documents textes) en tant qu'instances de l'ontologie. La classe *Term* représente les attributs extraits des items, et est peuplée par les termes extraits des items. La classe *Label* définit les termes les plus pertinents considérés pour la classification. Les relations *Broader* et *Narrower* décrivent la hiérarchie de subsomption, i.e. les relations de généralisation/specialisation entre les labels. La relation *hasTerm* lie les instances d'*Items* à leurs termes (instances de *Term*). Comme il est proposé dans l'approche de Werner et al. (2014), les relations *hasAlpha* et *hasBeta* lient une instance de *Label* aux instances de

1. Hierarchical Multi-Label Classification

2. L'ontologie noyau est une ontologie minimale, indispensable à la description des autres concepts de l'ontologie et de ses individus.

TABLE 3.1 – Concepts du modèle prédictif

Concepts DL	Description
$Item \sqsubseteq \exists hasTerm.Terms$	Associe un item à ses attributs (termes)
$Term \sqsubseteq asString.String$	Termes extraits des items
$Label \sqsubseteq Term$	Termes pertinents pour classer les items
$Label \sqsubseteq \forall broader.Label$	Relation de généralisation
$Label \sqsubseteq \forall narrower.Label$	Relation de spécialisation
$broader \equiv narrower^{-}$	Les relations Broader et Narrower sont inverses
$Item \sqcap Term \equiv \perp$	Items et Termes sont disjoints
$Label \sqsubseteq \forall hasAlpha.Term$	Termes composant une règle Alpha
$Label \sqsubseteq \forall hasBeta.Term$	Termes composant une règle Beta
$Item \sqsubseteq \exists isClassified.Label$	Représente la classification d'un item

Term utilisées pour créer les règles de classification pour ce même label.

Enfin, la relation *isClassified* est la relation de classification entre un *Item* et un *Label*. Le processus de création de règles et le processus de classification est décrit dans le chapitre suivant 4.

À titre d'exemple, soit un item $item_{news-politique}$ défini par deux termes $term_{referendum}$ et $term_{president}$, extraits lors de la phase d'indexation.

En logique de description, les assertions et les rôles suivants décrivent la relation entre l'item et ses termes après l'indexation :

```

Item(itemnews-politique)
Term(termpresident)
Term(termreferendum)
hasTerm(itemnews-politique, termpresident)
hasTerm(itemnews-politique, termreferendum)

```

L'objectif de la phase de hiérarchisation est de déterminer l'ensemble des *Label*, et de générer un ensemble de relations de subsomption formant la hiérarchie de labels. L'exemple suivant décrit une relation de subsomption entre un label $term_{politique}$ et un label plus spécifique $term_{politique-internationale}$:

```

Label(termpolitique-internationale)
Label(termpolitique)
broader(termpolitique-internationale, termpolitique)
narrower(termpolitique, termpolitique-internationale)

```

L'objectif de l'étape de Résolution est de générer un ensemble de règles de classifications spécifiques à chaque Label, où chaque règle est une clause de Horn³. Le processus de

3. Voir annexe A

création de règles est décrit dans le chapitre 4. L'exemple suivant décrit une règle de classification de type Beta⁴ :

$$\begin{aligned} & Item(item_{news-politique}) \wedge Term(term_{international}) \wedge Term(term_{politique}) \wedge \\ & Label(term_{politique-internationale}) \wedge hasTerm(item_{news-politique}, term_{international}) \wedge \\ & hasTerm(item_{news-politique}, term_{politique}) \Rightarrow isClassified(item_{news-politique}, term_{politique-internationale}) \end{aligned}$$

Enfin, l'objectif de l'étape de Réalisation de générer un ensemble de relations *isClassified*, i.e. les classifications de l'item, à partir des règles de classification et de la hiérarchie de labels :

$$isClassified(item_{news-politique}, term_{politique}) \quad (3.1)$$

Ce chapitre traite des trois premières étapes du processus, i.e. indexation, vectorisation et hiérarchisation. Les sections suivantes décrivent respectivement le processus d'extraction de termes (indexation), la génération des vecteurs de fréquence et de la matrice de cooccurrence (vectorisation), puis la sélection de labels et la création des relations hiérarchiques entre les labels (hiérarchisation). Les éléments relatifs à ces étapes sont surlignés en rouge dans la figure suivante.

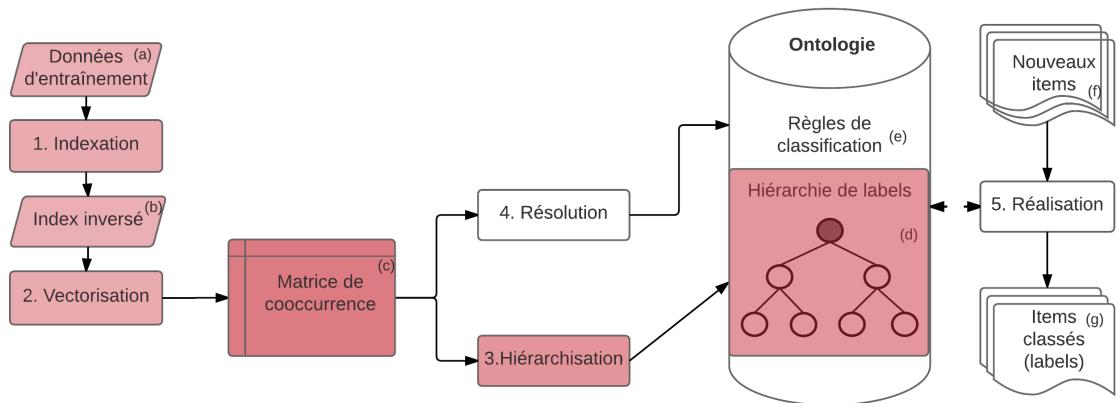


FIGURE 3.2 – Le processus SHMC : indexation, vectorisation et hiérarchisation

3.2/ INDEXATION

L'indexation est une étape obligatoire qui permet d'une part de palier *l'Analyse de Contenu Limitée*, et d'autre part de gérer une plus grande Variété de données. *L'analyse de Contenu Limitée* répond à la difficulté d'extraire automatiquement des informations fiables à partir de données non structurées (texte, image, vidéo...), notamment pour les systèmes de recherche d'information et de recommandation (Lops et al., 2011) (Bobadilla et al., 2013). Le type des données et le cas d'utilisation engendrent de nombreuses contraintes additionnelles qui doivent être prises en compte lors de l'indexation.

4. La définition des règles Alpha et Beta est donnée dans le chapitre suivant

Dans le cadre de l'apprentissage automatique, l'indexation est également la composante principale de la réduction de dimension des données initiales. Les principaux objectifs de la réduction de dimension sont les suivantes Guérif (2006) Ricci et al. (2011) :

- faciliter la visualisation et la compréhension des données
- réduire l'espace de stockage nécessaire
- réduire le temps d'apprentissage et d'utilisation
- identifier les caractéristiques pertinentes

L'indexation doit notamment réduire le bruit et la redondance des données d'apprentissage, et ainsi affiner la sélection des caractéristiques. Cette tâche est d'autant plus importante dans un contexte non supervisé, où les classes peuvent être directement affectées par les imperfections des données d'apprentissage.

Les méthodes employées lors de l'étape d'indexation peuvent être séparées entre la sélection de variables, qui consiste à choisir des caractéristiques dans l'espace de mesure, et l'extraction de caractéristiques, sélectionnées dans un espace transformé Guérif (2006).

Dans le cadre du traitement de données textuelles, l'extraction des caractéristiques consiste essentiellement en la sélection et la transformation des mots dans les données initiales. Afin de réduire l'espace des mots des données, dont la taille augmente avec la taille des données, une réduction de l'espace des mots est nécessaire. L'espace résultant de cette réduction est composé de caractéristiques communément nommées *termes*. La réduction de l'espace initial est effectué par une chaîne de traitement séquentielle, basée sur des techniques issues du traitement automatique du langage. La figure 3.3 décrit une séquence classique de traitements dans le cas de données textuelles. Différents sous-traitements d'analyse du langage peuvent être intégrés dans une chaîne, en fonction du cas d'application. Une analyse sémantique plus fine peut également être ajoutée suite aux traitements lexico-syntactiques.

La chaîne de traitement utilisée dans l'architecture SHMC répond à des contraintes spécifiques au cas d'utilisation industriel dans lequel ces travaux s'inscrivent. Elle est décrite de façon exhaustive dans la section 3.2.2.

3.2.1/ DESCRIPTION DE L'APPROCHE

L'indexation extrait les caractéristiques depuis une collection d'items et génère un index inversé des items. Dans notre cas d'utilisation chaque item est un document contenant du texte en langage naturel, et les termes sont des mots pertinents extraits du texte.

Chaque item est représenté par un vecteur v_{item_i} des termes extraits dans celui-ci, tel que :

$$v_{item_i} < \{term_1, k_1\}, \{term_2, k_2\} \dots, \{term_m, k_m\} >, \forall i \in C \quad (3.2)$$

où pour tout i , $item_i \in Item$ dans l'ontologie noyau, $k_1 \dots k_m$ est le nombre d'occurrences du terme dans l'item, et m est le nombre de termes distincts contenus dans l'item.

La création de l'index inversé permet de passer d'un vecteur de termes pour chaque item à un vecteur d'items v_{term_i} pour chaque terme tel que :

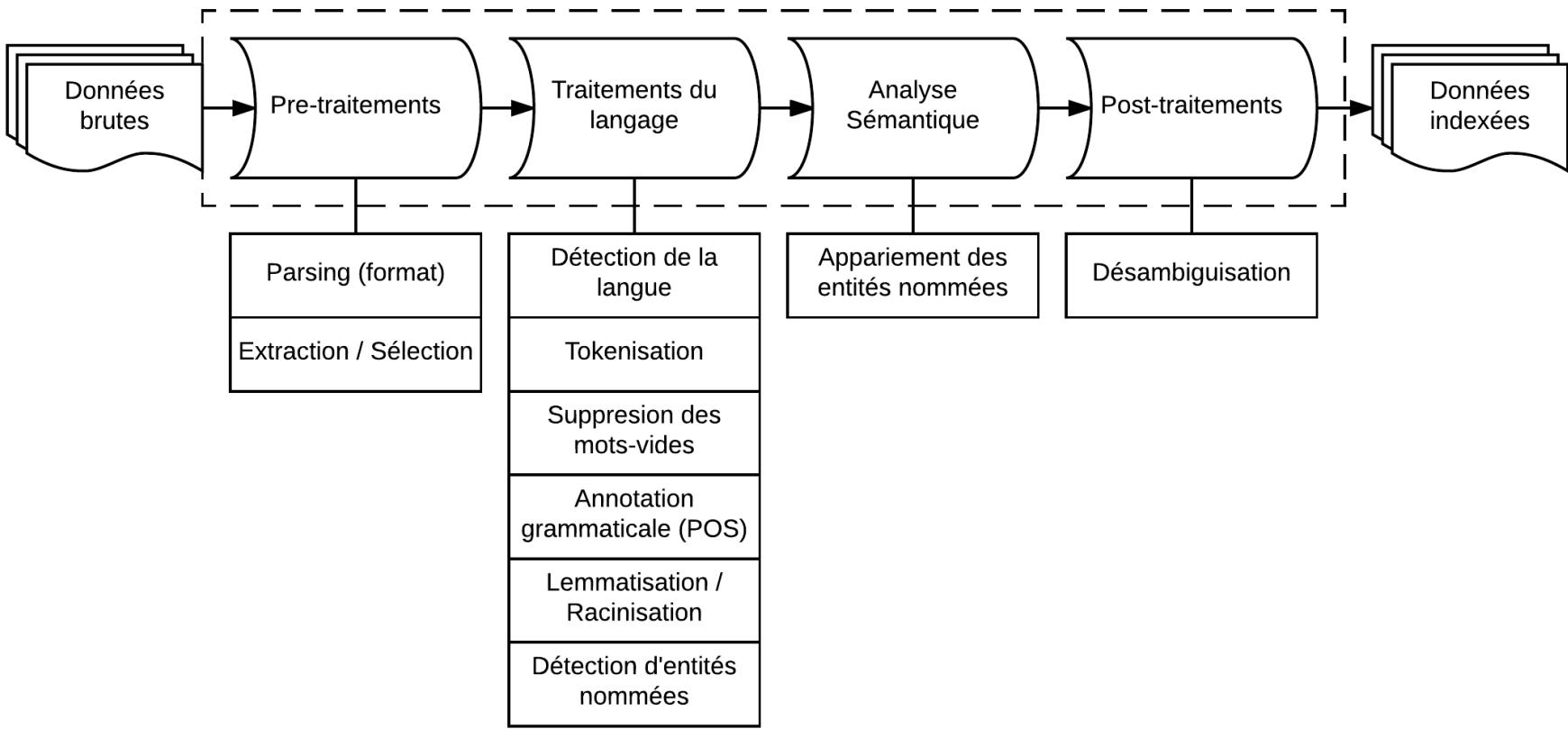


FIGURE 3.3 – Exemple de chaîne de traitement

$$v_{term_j} < item_1, item_2, \dots, item_n > \quad (3.3)$$

où pour tout j , $term_j \in Term$ dans l'ontologie noyau, $< item_1, item_2, \dots, item_n >$ sont les items où $term_j$ apparaît au moins une fois.

Le résultat de l'indexation est un index inversé (3.1.b) des items illustré par la figure 3.4.

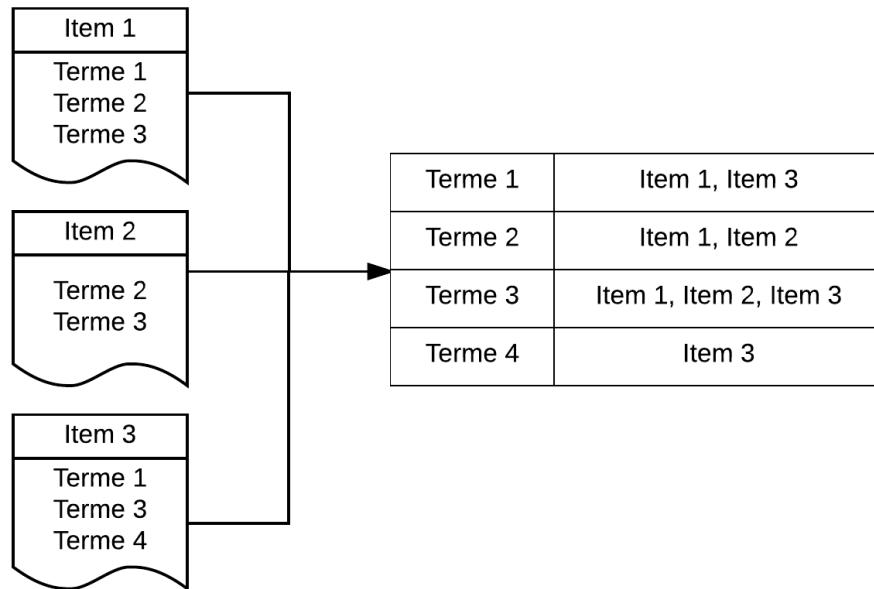


FIGURE 3.4 – Exemple d'index inversé

De la même façon que dans une base de données relationnelle indexée⁵, l'index inversé permet un accès direct et rapide à tout item de la collection à partir d'un ou plusieurs attributs (termes). L'index est utilisé comme base de construction de la matrice de cooccurrence (3.1.c) des termes dans l'étape suivante du processus (Vectorisation).

3.2.2/ IMPLÉMENTATION

Cette section présente l'implémentation de la phase d'Indexation.

Le processus complet est implémenté en Java, car un grand nombre de technologies et de bibliothèques nécessaires dans le processus sont libres compatibles avec Java. Dans les trois premières étapes du processus (indexation, vectorisation et hiérarchisation), des technologies et paradigmes de calcul dans un contexte distribué sont utilisés, en particulier le modèle MapReduce. MapReduce Dean et al. (2008) est un modèle de programmation distribué dont l'objectif est le traitement de larges volumes de données, dont l'implémentation la plus reconnue est intégrée au framework Hadoop, que nous utilisons dans notre implémentation. Ce modèle est basé sur deux fonctions, une fonction `map` qui génère reçoit en entrée un tuple clé/valeur, et renvoie un autre tuple clé/valeur intermédiaire. La seconde fonction `reduce` regroupe toutes les valeurs intermédiaires associées à une même clé. L'annexe B décrit les principes du paradigme MapReduce et décrit brièvement l'écosystème Hadoop.

5. <http://sql.sh/cours/index>

L'indexation effectue des pré-traitements (parsing) pour extraire le contenu des items et créer l'index. Dans le cas de données textuelles semi-structurées et/ou non-structurées, comme c'est le cas pour des données issues du web (pages web, flux rss, fichiers pdf/word), chaque type d'item requiert un parser spécifique afin de récupérer des informations pertinentes et réduire ainsi *l'Analyse de Contenu Limitée*. Suite à ces pré-traitements, le cas général en traitement automatisé du langage est d'appliquer une chaîne de traitements lexico-syntactiques qui permet de réduire en partie le bruit (termes inutiles/redondants), et d'améliorer l'extraction la représentation des items.

Suite à la phase de pré-traitement, la chaîne de traitement du langage est appliquée au texte extrait des items. La chaîne de traitement intégrée à la phase d'indexation est composée des étapes suivantes :

- Tokenisation : la tokenisation est le processus de décomposition d'un flux de texte en mots, phrases ou entités appelés tokens. Ce traitement est généralement le premier dans la chaîne, ainsi les traitements suivants ne considèrent les tokens plutôt que le texte dans son ensemble.
- Suppression des mots-vides : certains termes apparaissent très fréquemment dans un corpus de texte et sans apporter d'information utile. Ces termes sont considérés comme du bruit et peuvent être supprimés. Une liste de mots vides fixe est généralement utilisée, construite en fonction de la langue et du domaine.
- Lemmatisation : la lemmatisation consiste à regrouper les termes d'un même famille, en transformant les formes fléchies vers leur lemme (exemple : bleues ⇒ bleu). Contrairement à la racinisation (stemming), la lemmatisation conserve une morphologie syntaxique valide (un lemme est nécessairement un mot existant).
- Calcul des collocations : les collocations, ou n-grams, sont des termes qui cooccurrent régulièrement et de façon statistiquement significative, i.e. leur probabilité d'apparition simultanée est élevée comparativement au nombre d'items. Une mesure d'association statistique permet d'évaluer la pertinence des collocations. Dans notre cas, la fonction de vraisemblance (Loglikelihood ratio) Dunning (1993) est utilisé. Une valeur minimale est définie pour détecter les cooccurrences (voir section évaluation à la fin de ce chapitre).

Apache Lucene/Solr⁶ est utilisé pour intégrer cette chaîne de traitements, à l'exclusion du calcul des collocations. L'implémentation de l'analyseur syntaxique est décrite de façon exhaustive dans l'annexe D.

Le calcul de collocations est effectué via l'algorithme de collocations⁷ de la librairie Mahout The Apache Software Foundation (2013). Mahout est une librairie d'apprentissage automatique libre basée sur le framework Hadoop. La librairie Mahout et un serveur Solr sont déployés sur un cluster de machines Hadoop (<https://hadoop.apache.org/>), ce qui permet de répartir la charge de calcul de l'indexation.

Le résultat de l'indexation est l'index inversé des items, stocké dans le système de fichier distribué HDFS⁸ d'Hadoop⁹.

6. <http://lucene.apache.org/solr/>

7. <https://mahout.apache.org/users/basics/collocations.html>

8. Hadoop Distributed File System : https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

9. Voir annexe B

3.3/ VECTORISATION

L'étape de vectorisation a pour objectif de construire une matrice de cooccurrence des termes extraits lors de l'étape d'indexation. La construction d'une matrice de cooccurrence est une phase commune en apprentissage automatique, et est souvent le point de départ nécessaire à l'utilisation d'autres algorithmes d'apprentissage statistique, i.e. la plupart des mesures d'association basés sur l'étude de la corrélation et de la dépendance de variables aléatoires. De fait, ces méthodes sont utilisées pour des types très variés de données. Pour des données textuelles, les variables aléatoires peuvent être des termes ou des expressions plus complexes comme des collocations ou des entités nommées.

Lin et al. (2010) donne une définition formelle d'une matrice de termes : la matrice de cooccurrence d'un corpus de documents est une matrice carrée de taille $n \times n$, où n est le nombre de termes uniques dans le corpus. Une cellule $m_{i,j}$ contient le nombre de cooccurrences entre deux termes $term_i$ et $term_j$, i.e. le nombre d'apparitions simultanées des deux termes dans un contexte donnée. Le contexte considéré pour détecter une cooccurrence peut être une phrase, un paragraphe, document complet, ou un voisinage (fenêtre) en nombre de mots. Le nombre de cooccurrences est grandement affecté par le choix du contexte, et permet d'influer sur le nombre total de cooccurrences détectées, et par conséquent sur le coût calculatoire de la matrice.

3.3.1/ DESCRIPTION DE L'APPROCHE

La Vectorisation crée deux types de vecteurs de fréquence de termes pour les items indexés Salton et al. (1988) : un vecteur de fréquence des termes pour chaque item, ou term-frequency noté TF (1), et un vecteur de fréquence des termes dans l'ensemble du corpus, ou document-frequency noté DF (2). Une matrice de cooccurrence de termes est construite à partir de ces vecteurs. La matrice permet d'analyser les corrélations entre les termes extraits de la collection d'items, en l'occurrence leur proportion conditionnelle, ou fréquence relative. Les étapes suivantes du processus, hiérarchisation et résolution, exploitent la matrice de cooccurrence afin de générer le modèle prédictif.

Les deux types de vecteurs sont définis comme suit :

1. Term-frequency (pour chaque item), $\mathcal{V}_{tfidf}^i = \{(t, tfidf_{i,t,C})\}$, où t est le terme, i est l'item, C est la collection de document et $tfidf_{i,t,C}$ est la valeur TF-IDF Salton et al. (1988) du terme t dans l'item i .
2. Document frequency (collection de documents), $\mathcal{V}_{df} = \{(t, df_{t,C})\}$ où t est le terme, C est la collection d'items, et $df_{t,C}$ est le nombre d'items de la collection C où t apparaît.

La matrice de cooccurrence, notée cfm , représente la cooccurrence entre toute paire de termes ($term_i, term_j$) dans la collection de documents C , et est définie comme suit :

$$cfm(term_i, term_j) = \left| \left\{ item_{i,j} \in C | item_{i,j} \in vector_{term_i} \wedge item_{i,j} \in vector_{term_j} \right\} \right| \quad (3.4)$$

où $item_{i,j}$ est un item, $vector_{term_i}$ est le vecteur issu de l'index inversé pour le terme $term_i$, et $vector_{term_j}$ est le vecteur correspondant pour le terme $term_j$.

TABLE 3.2 – Matrice de cooccurrence de taille 3×3

	$Term_1$	$Term_2$	$Term_3$
$Term_1$	95	70	80
$Term_2$	70	80	60
$Term_3$	80	60	90

La matrice de cooccurrence cfm est une matrice symétrique de taille $n \times n$ où n est le nombre de termes différent dans l'index inversé. La diagonale de la matrice $cfm(term_i, term_i)$ représente le nombre d'occurrences total (document frequency) du terme $term_i$ dans la collection C . Par conséquent $cfm(term_i, term_i)$ est la valeur maximale de la ligne de la matrice correspondant au terme $term_i$. Le tableau 3.2 montre un exemple de matrice de taille 3×3 où $cfm(Term_1, Term_2) = cfm(Term_2, Term_1) = 70$

3.3.2/ IMPLÉMENTATION

Le calcul des vecteurs de fréquence V_{df} et $(item)V_{tfidf}^{item}$ pour un grand volume de données est une tâche coûteuse en temps et en ressources. La construction de la matrice est également une étape très coûteuse en temps et en ressources (voir 3.3.1). Le nombre de cooccurrences de termes est en effet fonction de la taille de la collection qui est importante dans un contexte Big Data. Un nombre de documents plus élevé engendre un nombre de termes plus grand, ce qui augmente de façon quadratique la taille de la matrice et le nombre de cooccurrences. De fait, les algorithmes de comptage des cooccurrences sont quadratiques ($O(n^2)$). Nous proposons donc de distribuer le processus sur un cluster de machines en utilisant un paradigme de calcul distribué, i.e. MapReduce. Pour générer les vecteurs de fréquence, les algorithmes de vectorisation¹⁰ issus de la librairie MapReduce Mahout The Apache Software Foundation (2013) sont utilisés. Deux implémentations sont communément utilisées pour compter l'ensemble des cooccurrences, i.e. l'algorithme des **paires** et l'algorithme des **stripes**. Ces deux algorithmes sont adaptés au paradigme MapReduceLin (2013), et utilisent les mêmes principes statistiques pour générer la matrice avec un résultat identique.

Pour chaque item, la fonction Map enregistre toutes les paires de termes $(term_i, term_j)$ dans un voisinage et comptabilise une cooccurrence. L'approche des paires enregistre individuellement toutes les cooccurrences, tandis que l'approche stripes regroupe et enregistre l'ensemble des cooccurrences pour chaque terme pour les émettre au reducer.

Ces deux approches sont complémentaires : l'algorithme des stripes est plus efficient en terme d'entrée/sorties sur le disque, car plus de données transitent directement en mémoire pour être traitées au niveau du reducer. En revanche, un goulot d'étranglement mémoire peut apparaître quand les données dépassent l'ordre du gigaoctet Lin et al. (2010). Dans notre cas d'utilisation, l'algorithme des stripes a été utilisé car plus adapté à la taille des données et aux spécifications matérielles dans nos expérimentations. Des optimisations supplémentaires de l'approche existent mais n'ont pas été implémentées par manque de temps.

Dans l'approche stripes, un tuple clé/valeur $(key_i, value_i)$ est émis par la fonction map, tel

10. <https://mahout.apache.org/users/basics/creating-vectors-from-text.html>

que :

- key_i est un terme $\in Term$

- $value_i$ est la liste de termes cooccurents de taille n : $(term_0, term_1, \dots, term_{n-1})$

où n est le nombre de termes dans l'item courant, car la fenêtre définie dans notre cas d'utilisation est le document complet.

Conformément au paradigme MapReduce, les tuples sont regroupés par clé key_i et la fonction reduce est exécutée pour chaque tuple émit ayant la même clé key_i . La fonction reduce regroupe l'ensemble des valeurs $value_i$ de chaque tuple, puis compte les cooccurrences. La sortie de la fonction reduce est une liste de tuples $(key_o, value_o)$ où key_o est une cooccurrence représentée par la concaténation de deux termes, et $value_o$ est la valeur absolue de la cooccurrence.

La section suivante décrit comment les informations de la matrice de cooccurrence sont exploitées dans l'étape de hiérarchisation pour déterminer l'ensemble de classes du modèle prédictif et leurs relations hiérarchiques.

3.4/ HIÉRARCHISATION

La hiérarchisation sélectionne les termes pertinents à partir des vecteurs de termes et définit ainsi les concepts qui appartiennent à la hiérarchie. Deux méthodes pour construire automatiquement une hiérarchie de classes (ou taxonomie) sont utilisées dans la littérature (De Knijff et al., 2013)(Meijer et al., 2014) :

- La méthode de subsomption qui construit les relations de généralisation - spécialisation en se basant sur la cooccurrence des concepts
- Le regroupement hiérarchique qui consiste à regrouper les concepts les plus proches les uns des autres.

La méthode de subsomption est en l'occurrence adaptée à notre solution, de par sa performance. Les vecteurs générés lors de la vectorisation sont utilisés en entrée du processus de hiérarchisation. La sortie du processus est la hiérarchie de labels, composée des relations de subsomptions entre les labels.

3.4.1/ DESCRIPTION DE L'APPROCHE

La hiérarchisation détermine les labels (classes) et les relations hiérarchiques entre les labels. Les labels sont un sous-ensemble des termes : les termes les plus pertinents sont désignés comme labels, selon l'approche décrite dans Feldman et al. (1998). Cette méthode exploite se base sur les vecteurs de fréquence calculés lors de la vectorisation. Pour chaque terme $term_j$, la proportion $P_C(term_j)$ d'items de C où apparaît $term_j$ est définie par :

$$P_C(term_j) = \frac{df_C(term_j)}{|C|} \quad (3.5)$$

où $df_C(term_j)$ est la fréquence (document frequency) de $term_j$ dans l'ensemble du corpus C , et $|C|$ est la taille du corpus C .

Pour l'ensemble des termes dont la proportion $P_C(term_j)$ est supérieure à un seuil (IT),

i.e. $P_C(term_j) \geq IT$, où $term_j \in Term$, l'ensemble ω_{IT} est défini tel que :

$$\omega_{IT} = \{term_j \in Term | P_C(term_j) \geq IT\} \quad (3.6)$$

Les termes appartenant à cet ensemble sont classés comme *Label* dans l'ontologie, tel que $Label \sqsubseteq Term$.

Pour construire les relations hiérarchiques entre les labels, nous adoptons la méthode de subsomption pour sa performance. La matrice de cooccurrence est utilisée pour construire les relations hiérarchiques, en suivant l'approche de Sanderson et al. (1999). Soit deux labels x et y , x subsume y (figure 3.5(A)), i.e. $x < y$ si :

$$(P_C(x|y) = 1) \wedge (P_C(y|x) < 1) \quad (3.7)$$

où :

- $P_C(x|y)$ est la proportion conditionnelle, i.e. le nombre d'items de C communs à x et y , en fonction du nombre d'items où y apparaît tel que :

$$P_C(x|y) = \frac{x \cap y}{y} = \frac{cfm(x,y)}{df_y} \quad (3.8)$$

- $P_C(y|x)$ est la proportion conditionnelle inverse, i.e. le nombre d'items de C communs à x et y , en fonction du nombre d'items où x apparaît :

$$P_C(y|x) = \frac{x \cap y}{x} = \frac{cfm(x,y)}{df_x} \quad (3.9)$$

Dans Sanderson et al. (1999) les auteurs remarquent que beaucoup de labels ne sont pas inclus par les conditions définies ci-dessus, et proposent d'assouplir la première condition $P_C(x|y) = 1$ pour les inclure (Figure 3.5 (B)). La subsomption est alors redéfinie par :

$$(P_C(x|y) \geq st_1) \wedge (P_C(y|x) < 1) \quad (3.10)$$

où $st_1 \in [0, 1]$ est le seuil de subsomption pour la première condition. De Knijff et al. (2013) propose également d'assouplir la seconde condition $P_C(y|x) < 1$ (Figure 3.5 (C)). La définition de la subsomption est alors :

$$x < y = (P_C(x|y) \geq st_1) \wedge (P_C(y|x) < st_2) \quad (3.11)$$

où $st_1 \in [0, 1]$ est le seuil pour la première condition, et $st_2 \in [0, 1]$ le seuil pour la seconde condition tel que $st_1 \geq st_2$. De façon verbeuse, si x apparaît avec une proportion supérieure à st_1 dans les documents où y apparaît, et y apparaît avec une proportion supérieure à st_2 dans les documents où y apparaît, alors x subsume y , i.e. $x < y$.

L'application de cette méthode à l'ensemble des labels définit un graphe acyclique dirigé, i.e. la hiérarchie de labels qui est ensuite intégrée à l'ontologie noyau (voir A).

3.4.2/ IMPLÉMENTATION

Le choix des labels est effectué via l'application de la méthode de recherche d'information définie par Feldman et al. (1998), en fonction de la proportion $P_C(term)$ de chaque terme $term \in Term$ et de la collection C . Cette étape ne requiert pas de calcul spécifique car elle est effectuée par le serveur Solr. La proportion est accessible par requête au serveur.

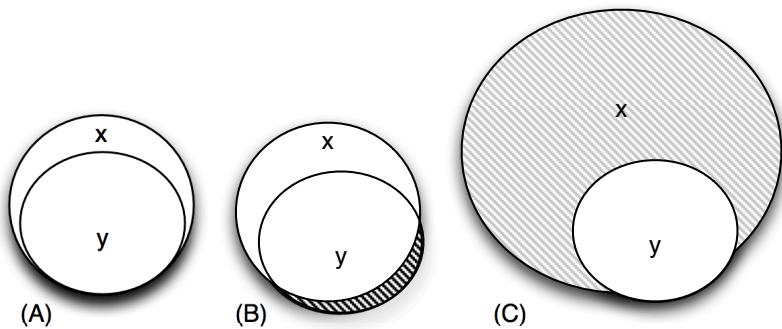


FIGURE 3.5 – Diagrammes d'Euler de la méthode de subsomption

Étant basée sur l'exploitation de la matrice de cooccurrence, le calcul des relations hiérarchiques peut être une tâche coûteuse. L'algorithme de subsomption a donc été implémenté via le paradigme MapReduce.

La matrice de cooccurrence cfm peut être vue comme un ensemble de paires $\langle (term_i, term_j), P(term_i|term_j) \rangle$, où $term_i$ et $term_j$, où $P(term_i|term_j)$ est la proportion conditionnelle des items communs à $term_i$ et $term_j$ par rapport à la proportion $P(term_j)$.

L'ensemble des paires $\langle (term_i, term_j), P(term_i|term_j) \rangle$ est donnée comme entrée de la fonction map, où le couple $(key, value)$ est défini comme suit :

- key est un tuple $(term_i, term_j)$ où $term_i$ et $term_j$ sont des termes issus de la phase de vectorisation
- $value$ est la proportion $P(term_i|term_j)$

Le seuil IT est appliqué par dans fonction map. Si chaque terme est inclus dans ω_{IT} , la fonction map émet un couple $((label_i, label_j), P(label_i|label_j))$ où $(label_i, label_j)$ sont les labels de l'ensemble ω_{IT} correspondants aux termes initiaux $term_i$ et $term_j$.

La fonction reduce regroupe les tuples en sortie par $label_i$, ce qui permet de vérifier l'ensemble des relations de subsomption potentielles des couples $(label_i, label_j)$, en fonction de la relation 3.11. Le résultat de la fonction reduce est l'ensemble des tuples $\langle (label_i, label_j) \rangle$, où $label_j$ est un potentiel parent de $label_i$.

L'algorithme de subsomption est déployé sur le cluster Hadoop. La librairie OWL-API(<http://owlapi.sourceforge.net/>) permet de générer les relations hiérarchiques à partir des tuples $\langle (label_i, label_j) \rangle$ issus de la phase de réduction. La hiérarchie de labels ainsi générée est stockée sur HDFS puis intégrée à la base de connaissances. L'utilisation de cette hiérarchie de labels dans le processus de classification est décrit dans le chapitre 4.

3.5/ ÉVALUATION QUANTITATIVE

Cette évaluation initiale a pour objectif de valider la conception de l'approche, et de montrer que le processus a été implémenté avec succès, en décrivant de façon indicative le résultat du processus d'apprentissage décrit dans cette section, i.e. la taxonomie générée automatiquement.

L'environnement de test et les données utilisées sont d'abord décrites, suivis par les

résultats de l'évaluation et une discussion.

3.5.1/ JEUX DE DONNÉES

Le jeu de données utilisé dans cette évaluation est composé d'articles issus de la version française de Wikipedia (articles en langage naturel non structurés). Le jeu de données est décomposé en 3 parties de taille variable, pour étudier l'impact de la taille des données sur le processus. Les spécificités du jeu de données sont présentées dans le tableau 3.3.

TABLE 3.3 – Jeux de données wikipedia

Jeu de données	Cardinalité (items)	Taille (GB)
Wikipedia 1	174900	1.65
Wikipedia 2	407000	2.21
Wikipedia 3	994000	5
Wikipedia 4	2788500	11

3.5.2/ ENVIRONNEMENT DE TEST

Le processus est déployé sur un cluster Hadoop distant via le service cloud "Google Compute Engine"¹¹. Le cluster est composé de 4 nœuds (1 maître, 3 esclaves). Chaque nœud correspond à une instance, i.e. une machine virtuelle dédiée disposant de ressources matérielles propres. Le tableau 3.4 décrit les spécifications des instances. On s'intéresse entre autres aux coûts en terme de ressources matérielles et de temps de calcul dans cette évaluation. Pour analyser ces performances, les interfaces natives du framework Hadoop ainsi que les interfaces de service Ambari¹² sont utilisées.

TABLE 3.4 – Spécifications matérielles

Ressource	Description
CPU (par nœud)	2.5GHz Intel Xeon E5 v2 (Ivy Bridge)
RAM (par nœud)	7.5GB
Espace disque (par nœud)	500GB

3.5.3/ PARAMÉTRAGE

Chaque étape du processus est dépendante d'un ensemble de paramètres, qui ont un impact important sur la création du modèle prédictif. A des fins de reproductibilité des résultats, le tableau 3.5 décrit l'ensemble des paramètres déterminants. Comme indiqué dans la description de l'approche, certains des paramètres concernent les algorithmes de la librairie Mahout (support minimum, taille des collocations, fréquence minimum et

11. <https://cloud.google.com/compute/>

12. Apache Ambari, <https://ambari.apache.org/>

maximum, ratio de vraisemblance). L'outil "seq2sparse"¹³ permet de calculer les collocations de termes en conjonction avec un analyseur Lucene personnalisé qui intègre les traitements automatiques du langage implémentés (section 3.2.2).

TABLE 3.5 – Paramétrage du processus

Paramètre	Étape	Valeur
Minimum support	Indexation	1750
Taille des Collocations (n-grams)	Indexation	2
Maximum document frequency	Indexation	90%
Minimum document frequency	Indexation	1
Log-Likelihood Ratio	Indexation	17500
Seuil de détection des labels	Hiérarchisation	2%
Seuil <i>st2</i> (bottom)	Hiérarchisation	10%
Seuil <i>st1</i> (top)	Hiérarchisation	90%

3.5.4/ RÉSULTATS

Ces résultats évaluent le potentiel de passage à l'échelle des 3 premières étapes du processus (Indexation, Vectorisation et Hiérarchisation). Les données suivantes sont recueillies pour les 3 sous-parties du jeu de données :

- Temps d'exécution total de chaque étape par jeu de données
- Nombre de termes extrait par jeu de données
- Nombre de labels extrait par jeu de données
- Nombre de relations hiérarchiques extraites par jeu de données

La figure 3.6 présente le nombre de termes extraits en fonction de la taille du jeu de données. Le nombre de termes extraits augmente naturellement avec la taille du jeu de données. La croissance du nombre de termes en fonction de la taille des données est supposée limitée, car le nombre de mots d'une langue est limité par définition. Les dictionnaires français, formes fléchies exclues, recensent entre 30000 et 100000 mots en moyenne Malherbe (1983). D'autres langues recensent plus de mots (les dictionnaires anglais comptent en général 200000 mots). La taille des jeux de données utilisés ne permet pas cependant d'observer cette limite. La suppression des mots vides et la lematisation permettent de limiter grandement la croissance du nombre de termes extraits, et donc de réduire la taille de la matrice de cooccurrence. Ce point est important pour réduire le coût calculatoire de la matrice.

Le nombre de labels extract pour chaque jeu de données est présenté dans la figure 3.7. Les labels étant un sous-ensemble des termes, le nombre de termes impacte le nombre de labels. Cependant, due à l'approche utilisée pour extraire les labels, nous pouvons observer que le nombre de labels diminue lorsque la taille du corpus augmente. En effet, l'approche de sélection des labels considère un seuil statique (2%, tableau 3.5), non adapté à l'augmentation de la taille des données. Pour améliorer la sélection des labels de façon non supervisée en suivant l'approche de Feldman et al. (1998), il serait nécessaire de changer dynamiquement le seuil *IT*. Cette adaptation dynamique n'a pas

13. <https://mahout.apache.org/users/basics/creating-vectors-from-text.html>

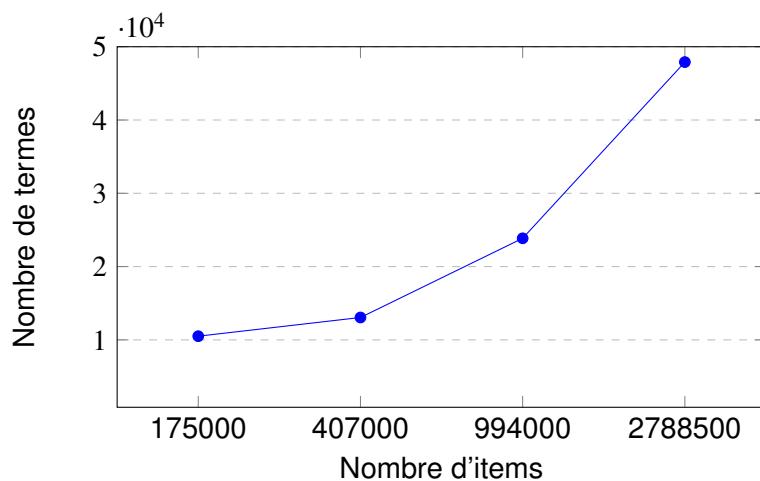


FIGURE 3.6 – Nombre de termes issus de l'apprentissage

été considérée compte tenu du nombre de variables sur lequel le processus s'appuie, ce qui rend une évaluation exhaustive de l'ensemble des variables complexe. Aussi, l'optimisation de l'approche de sélection des labels ne fait pas partie de nos objectifs principaux. Une autre solution est d'opter pour une approche de sélection des labels différente, par exemple une approche linguistique ou sémantique (sélection de groupes nominaux, verbes, entités nommées...), en fonction du cas d'utilisation réel. La qualité du vocabulaire (labels) est évalué de façon qualitative dans la section suivante.

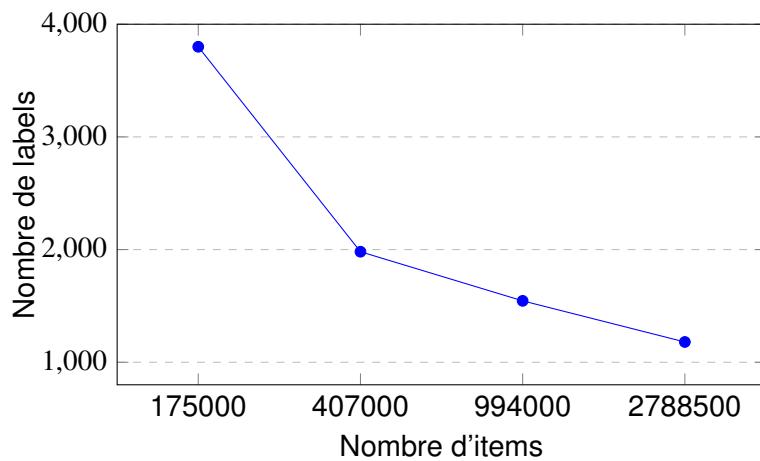


FIGURE 3.7 – Nombre de labels issus de l'apprentissage

Le nombre de relations hiérarchiques en fonction de la taille des données est présenté en figure 3.8. Du fait du nombre réduit de labels extraits avec l'augmentation de la taille des données, le nombre de relations de subsomption diminue en conséquence. Tout comme pour les labels, une évaluation subjective de la qualité des relations extraites est présentée dans la section suivante.

Enfin, le tableau 3.6 présente le temps d'exécution des différents 3 étapes présentées dans ce chapitre. Certaines sous-étapes pertinentes sont également détaillées. Comme mentionné en section 3.3.1, les résultats montrent que le point critique de la construc-

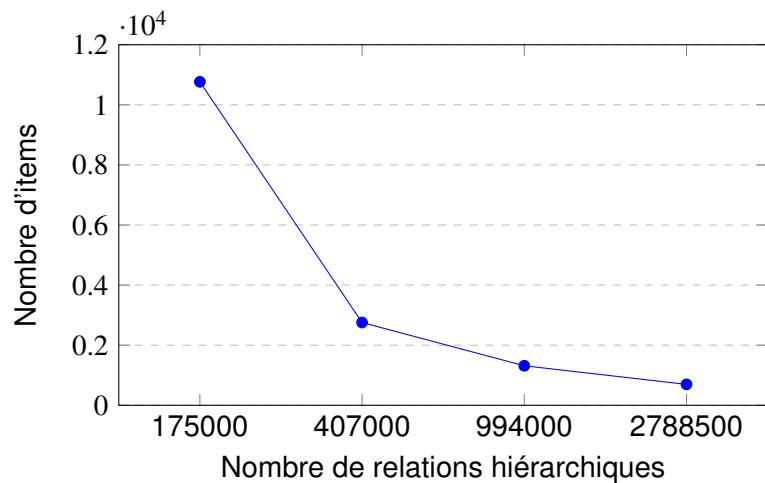


FIGURE 3.8 – Nombre de relations hiérarchiques issus de l'apprentissage

tion du modèle est la génération de la matrice de cooccurrence (sous-processus de la vectorisation) implémentée via l'approche stripes (3.3.1). Cette étape est très coûteuse en temps quelque soit la taille des données. Il est nécessaire de relativiser ces résultats pour différents points :

1. L'environnement de test, notamment la taille du cluster de machines est une limitation importante. Lin et al. (2010) évalue à la fois l'impact de la taille des données et de la taille du cluster sur l'exécution de l'algorithme stripes, et montre que la montée en charge par augmentation des ressources matérielles pallie en grande partie l'augmentation de la taille des données.
2. De par sa conception orientée batch¹⁴, le temps de calcul n'est pas la priorité numéro une de l'architecture.
3. Enfin, un combiner¹⁵ a été ajouté dans l'implémentation actuelle de l'architecture, ce qui améliore le temps de calcul de cette phase. Le taux d'amélioration de l'ajout du combiner n'a pas été analysé pour ce jeu de données¹⁶.

3.6/ ÉVALUATION QUALITATIVE

Cette section se focalise sur l'évaluation qualitative des étapes de l'architecture décrites dans ce chapitre. Les ontologies sont des structures de connaissances complexes et hétérogènes. Par conséquent, il existe de multiples approches pour évaluer la qualité

14. Le Batch processing, ou "traitement par lots", est le paradigme de calcul standard de MapReduce, où l'ensemble des données est traité de façon séquentielle par une série de sous-processus (jobs). Le processus est exécuté à un instant t fixe, et possède un début et une fin par opposition au paradigme de traitement par flux (streaming).

15. Un combiner permet de regrouper les valeurs intermédiaires émises par la fonction map au niveau du nœud où elle est exécutée, et donc d'envoyer moins de données au reducer https://www.tutorialspoint.com/map_reduce/map_reduce_combiners.htm

16. Le gain en temps de calcul varie selon les cas d'utilisation. L'approche stripes agrège un nombre important de valeurs au niveau du reducer, et bénéficie donc grandement du combiner. L'usage d'un combiner dans le cas de l'algorithme wordcount peut par exemple diviser jusqu'à deux le temps de calcul <https://blog.cloudera.com/blog/2009/12/7-tips-for-improving-mapreduce-performance/>

TABLE 3.6 – Temps d'exécution

Jeu de données	Indexation	Création vecteurs	Création matrice	Hiérarchisation
Wikipedia 1	15 min	45 min	6h	6 min
Wikipedia 2	23 min	56 min	9h30	8 min
Wikipedia 3	52 min	1h52	14h40	7 min
Wikipedia 4	2h20 min	4h35	35h	7min

d'une ontologie Porzel et al. (2004). Les méthodes d'évaluation globales d'une ontologie sont généralement séparées en 4 catégories Brank et al. (2005) Raad et al. (2015) :

- Gold Standard : cette approche se base sur la comparaison de l'ontologie à évaluer avec une ontologie de référence
- Task-based : cette approche évalue la performance de l'ontologie dans le cadre d'une tâche spécifique. Cette approche est employée dans le chapitre 4 pour évaluer notre approche pour la tâche de classification.
- Domain coverage : approche basée sur la comparaison de la couverture de l'ontologie à des données spécifiques au domaine
- Expert-based : approche qui demande l'intervention d'experts pour évaluer manuellement l'ontologie

La complexité et la taille des ontologies peut rendre difficile l'utilisation de ces méthodes pour évaluer une ontologie dans son intégralité. Certaines approches se concentrent sur l'évaluation séparée des différentes "couches" de connaissances de l'ontologie Brank et al. (2005) Wong et al. (2012). Ce type d'approche est particulièrement adapté dans le cas de l'apprentissage automatique d'une ontologie, où les différentes couches de l'ontologie sont clairement identifiables, notamment lorsque chaque couche est générée par un algorithme spécifique. L'évaluation peut alors porter sur ces couches de connaissances telles que définies dans Brank et al. (2005) :

1. Lexique / Vocabulaire : cette couche inclut les concepts de l'ontologie et leur définition
2. Hiérarchie / Taxonomie : cette couche inclut les relations hiérarchiques entre les concepts
3. Autres types de relations : cette couche inclut tous les autres types de relations intégrées dans l'ontologie
4. Contexte / application : cette couche inclut l'analyse des bénéfices de l'ontologie dans un contexte donné. Elle peut par exemple concerner son lien avec d'autres ontologies, ou son usage dans une application donnée. Ce type d'évaluation recoupe les évaluations globales de type "Task-based".
5. Syntaxique et Structurelle : ces évaluations concernent respectivement l'étude du langage formel qui décrit l'ontologie, et les critères de construction relatifs à sa construction et à sa maintenance spécifiques au domaine. L'évaluation de cette couche requiert généralement l'intervention d'experts, et n'est pas applicable dans le cas d'une ontologie générée automatiquement à partir de grands volumes de données.

Dans le cas du processus SHMC, l'ontologie est un modèle prédictif dont l'objectif est la classification d'items. L'ontologie est constituée d'une part d'une hiérarchie de labels, dont la construction est décrite dans ce chapitre, et de règles de classification (chapitre suivant). Compte tenu de l'étude des méthodes d'évaluations, nous considérons deux types

d'évaluation du processus. Cette section présente une évaluation qualitative par couches de la hiérarchie de labels (ou taxonomie), dans un contexte non supervisé, basée sur les méthodes d'évaluation des ontologies décrites ci-dessus. Une seconde évaluation globale de type "Task-based" est présentée dans le chapitre suivant, et s'intéresse aux performances du processus pour la tâche de classification hiérarchique multi-label. Cette seconde évaluation est également considérée comme une évaluation qualitative d'une couche supplémentaire de l'ontologie, i.e. les règles de classification, et est basée sur des métriques d'évaluation du domaine de l'apprentissage automatique.

Pour évaluer la hiérarchie de labels, l'évaluation est divisée en 3 parties : une évaluation lexicale (vocabulaire), une évaluation hiérarchique, et une étude de la corrélation entre ces deux premières évaluations.

3.6.1/ ÉVALUATION LEXICALE

Pour évaluer la couche lexicale, i.e. les labels extraits à partir des données, les plupart des approches effectuent une comparaison à un lexique de référence. La précision lexicale (Lexical Precision, ou LP) et le rappel lexical (Lexical Recall, LR) sont alors utilisés comme mesures d'évaluation Meijer et al. (2014). La précision lexicale représente la proportion de labels extraits correspondant à un label du lexique de référence :

$$LP(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_C|} \quad (3.12)$$

où O_C est l'ontologie évaluée, O_R l'ontologie de référence, C_C la liste des concepts (labels) de O_C , et C_R la liste des concepts de O_R .

Le rappel lexical est la proportion de labels valides (présents dans le lexique de référence) comparativement au nombre total de labels valides, i.e. la taille du lexique de référence.

Des mesures de similarité de chaînes de caractères sont souvent utilisées pour effectuer cette tâche Brank et al. (2005). Dans le cas de la comparaison de labels issus d'une ontologie, des frameworks dédiés tels que LinkQL Hassanzadeh et al. (2009) et SilkVolz et al. (2009) ont été développés dans le cadre de l'appariement d'ontologies du Linked Data. Ces frameworks intègrent des algorithmes de comparaison de chaînes et de pré-traitements syntaxiques, et sont adaptés à la comparaison d'un grand nombre de triplets.

3.6.1.1/ PROTOCOLE D'ÉVALUATION

La chaîne de traitement du langage présentée dans la section précédente réduit l'espace des termes extraits depuis le corpus de données. Les termes et collocations de termes les plus fréquents constituent les labels, i.e. les concepts pertinents identifiés. Nous définissons la précision et le rappel lexical comme le rapport :

$$LP(O_{SHMC}, O_R) = \frac{|C_{SHMC} \cap C_R|}{|C_{SHMC}|} \quad (3.13)$$

où O_{SHMC} est l'ontologie évaluée, O_R est l'ontologie de référence (DBpedia ou Wordnet), C_{SHMC} la liste des concepts (labels) de O_{SHMC} , et C_R la liste des concepts de O_R .

Notre système extrait des termes, concepts et expressions communes de façon non supervisée. Nous proposons de comparer les attributs extraits par le système à des ressources terminologiques généralistes. Notre chaîne de traitement étant spécifiquement dédiée au français, des ressources terminologiques en français sont utilisées.

DBpedia structure les métadonnées de Wikipedia sous la forme d'un graphe de connaissances. Étant donné que les données utilisées pour générer la taxonomie sont issues de Wikipedia, le choix de DBpedia comme ressource d'appariement est évident. La version française de DBpedia est nommée DBpediaFr¹⁷.

Wordnet est un graphe de connaissances lexicales, qui représente de façon exhaustive les sens de termes et concepts communs. cette ressource est largement utilisée dans la littérature notamment pour évaluer le sens d'un terme. Cette ressource est complémentaire de DBpedia, laquelle représente globalement des concepts plus complexes. Les traductions françaises de la version officielle de Wordnet sont basées sur le Wolf, une traduction manuelle de Wordnet Sagot et al. (2008). Une autre traduction automatisée a été publiée par Pradet et al. (2014), mais ne fait pas partie du Wordnet standard, elle n'a donc pas été utilisée.

Les versions françaises de DBpedia et Wordnet sont incomplètes comparativement à leur équivalent anglais, par conséquent certaines traductions de concepts sont manquantes, ce qui impacte les appariements possibles.

Comme d'autres travaux récents de la littérature tels que Meijer et al. (2014), l'ontologie de référence (DBpedia et Wordnet) est beaucoup plus large que l'ontologie issue du processus d'apprentissage (hiérarchie de labels). Cette particularité induit nécessairement un rappel lexical faible, qui devient une mesure peu pertinente. Par conséquent seule la précision lexicale est évaluée. Le framework Silk Volz et al. (2009) est utilisé pour effectuer la comparaison des ontologies¹⁸. Les labels sont comparés via une distance de levenstein de 0 (comparaison exacte) entre les chaînes de caractères correspondants aux labels. La désambiguïsation contextuelle des labels n'est pas prise en compte.

3.6.1.2/ RÉSULTATS

Le tableau 3.7 présente les résultats de l'évaluation lexicale. L'ensemble des labels est comparé à trois types de classes : les catégories et entités de DBpedia, et les synsets¹⁹ de Wordnet. Les proportions du tableau 3.7 sont exprimées en rapport au nombre total de labels de la hiérarchie, composée de 5368 labels. La précision lexicale totale est également présentée, par la concaténation des 3 ensembles de référence (DBpedia /wordnet) et déduplication (suppression des doublons).

Les résultats de l'évaluation lexicale soulignent 3 caractéristiques du processus SHMC en contexte non supervisé :

1. Le processus permet d'extraire des labels pertinents correspondants à des concepts simples du lexique Wordnet. La comparaison étant basée sur une distance de levenstein de 0 (comparaison exacte), cette évaluation suggère que la

17. <http://fr.dbpedia.org/>

18. Ce framework, initialement développé pour l'appariement d'ontologies du Linked Data, est adapté à la comparaison d'ontologies ayant une dimension élevée

19. Le synset est le concept principal de Wordnet. Il représente un concept (voir <https://wordnet.princeton.edu/>)

TABLE 3.7 – Précision lexicale basée sur Wordnet et DBpedia

Ressource	Précision lexicale
DBpediaFr	
Catégories	18.4%
DBpediaFr	
Entités (instances)	12.8%
Wordnet	27.6%
DBpedia + Wordnet (dédupliqué)	40.4%

chaîne de traitement du langage sur laquelle est basée l'extraction des labels est efficace pour la normalisation de texte en langage naturel en français.

2. L'approche hybride statistique / lexicale permet l'extraction de termes complexes pertinents, telles que les entités et catégories issues de DBpedia. Cette extraction est permise notamment par le calcul des collocations (n-grams). D'après nos observations empiriques, les entités nommées (villes, personnes, organisations) constituent la majorité de ces entités. Ce critère est important pour l'application industrielle et sera évoquée dans le chapitre 7.
3. Les résultats sont mitigés, en effet 60% des labels extraits ne correspondent pas à des entités de référence après comparaison exacte. Il faut noter que le protocole d'évaluation et les données de référence ne permettent pas un recouvrement optimal des labels, et dégradent les résultats. D'après nos observations, certains termes extraits sont pertinents, mais une comparaison exacte ne permet pas un rapprochement avec les données de référence. Par exemple, le label pertinent "skieur alpin" est manquant dans DBpedia. Une classe sémantiquement proche est représentée par le label (propriété *rdfs:label*) "skieurs_alpins_par_nationalité"²⁰. Cette classe est de surcroît complexe à désambiguïser (une comparaison de chaîne partielle ne permettrait pas sa détection).

Du bruit persiste dans les termes extraits (3). Des efforts supplémentaires doivent être entrepris pour améliorer à la fois la paramétrisation du processus, et la chaîne de traitements du langage. Ces résultats sont néanmoins encourageants, compte tenu du protocole d'expérimentation, qui affecte négativement les résultats (appariement exact des termes et peu de ressources terminologiques).

La pertinence des termes extraits de façon non-supervisée constitue un avantage pour la construction d'une hiérarchie de labels à partir de grands volumes de données.

3.6.2/ ÉVALUATION HIÉRARCHIQUE

La hiérarchie de labels issue du processus SHMC peut être vue comme un ensemble de relations sémantiques entre des paires de concepts. Une méthode d'évaluation courante

20. http://dbpedia.org/page/Category:Male_alpine_skiers_by_nationality

de ces relations est la "semantic cotopy" De Knijff et al. (2013) Dellschaft et al. (2006), qui mesure la proportion de relations partagées avec une ontologie de référence.

Compte tenu du contexte Big Data et traitement massif de données de notre approche, et des données utilisées dans notre évaluation, il n'a pas été possible d'utiliser cette méthode. Nous proposons d'utiliser la distance Google normalisée (Normalized Google Distance ou NGD) Cilibrasi et al. (2007) pour cette évaluation, qui est une mesure de distance sémantique²¹.

La NGD de deux termes x et y , notée $NGD(x, y)$, est une approximation de la distance (écart) sémantique entre x et y . Elle est basée sur le nombre de résultats retournés par un moteur de recherche généraliste, en fonction de la cooccurrence des deux termes dans l'index sur lequel est basé le moteur.

La distance Google Normalisée est définie comme :

$$NGD(x, y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x, y)}{\log M - \min\{\log f(x), \log f(y)\}} \quad (3.14)$$

où M est le nombre total de pages indexées, x et y sont les termes, et $f(x), f(y)$ correspondent respectivement au nombre de résultats retournées par le moteur pour x et y .

L'avantage principal de cette mesure est qu'aucune ressource lexicale ou sémantique n'est nécessaire pour mesurer la distance. La théorie sur laquelle est basée la mesure NGD est développée de façon exhaustive dans Cilibrasi et al. (2007). Bien que la NGD soit symétrique et ne permette donc pas d'analyser la direction d'une relation ($NGD(Outil, Tournevis) = NGD(Tournevis, Outil)$), la méthode de subsomption appliquée à de large volumes de données assure en théorie la cohérence de la direction des relations.

Une distance sémantique locale est d'abord calculée pour chaque relation de la hiérarchie de labels. Une mesure globale est ensuite calculée par moyenne sur l'ensemble de la hiérarchie.

3.6.2.1/ PROTOCOLE D'ÉVALUATION

La section précédente a présenté la distance Google, qui requiert pour son calcul la taille de l'index du moteur de recherche, notée M .

Cette valeur n'est pas fixe dans le temps, et n'est pas rendue publique avec exactitude par la plupart des moteurs de recherche (Google, Bing, Yahoo...). Cependant, une approximation de la valeur est suffisante Cilibrasi et al. (2007), tant que M a un ordre de grandeur important ($M \gg f(x)$). Le moteur de recherche Bing²² a été utilisé pour effectuer cette évaluation. Les requêtes à l'index ont été effectuées durant le mois de novembre 2015. L'estimation de M est basée sur une approximation correspondant à la date des tests, basée sur van den Bosch et al. (2015) : $M \approx 13 \cdot 10^9$ ²³.

21. La distance sémantique est l'inverse de l'appartenance sémantique, qui définit la similarité sémantique de deux concepts. Cette similarité concerne tout type de relation sémantique, où les concepts partagent un sens commun (synonymie, homonymie, hyperonymie), ou une relation fonctionnelle / associative (méronymie, antonymie)

22. <https://www.microsoft.com/france/bing/>

23. Une approximation de la taille de l'index à une date donnée peut être trouvée à l'adresse suivante <http://www.worldwidewebsize.com/>

D'après Cilibrasi et al. (2007), la distance est comprise entre 0 et 1 dans le cas général. Certains cas particuliers sortent de l'intervalle $[0, 1]$ lorsque la différence d'ordre de grandeur des fréquences est élevé. Une distance de 0 est le cas idéal, i.e. les deux termes sont identiques. Une distance supérieure ou égale à 1 correspond à un recouvrement nul des deux termes. Une distance sémantique locale est calculée pour chaque couple de termes $(term_i, term_j)$ liés par une relation de subsomption. Cette distance est notée $NBD_{local}(term_i, term_j)$.

Une distance globale pour l'ensemble des relations hiérarchiques, notée NBD_{global} , est calculée par moyenne des distances locales :

$$NBD_{global} = \frac{\sum NBD_{local}(term_i, term_j)}{|R|} \quad (3.15)$$

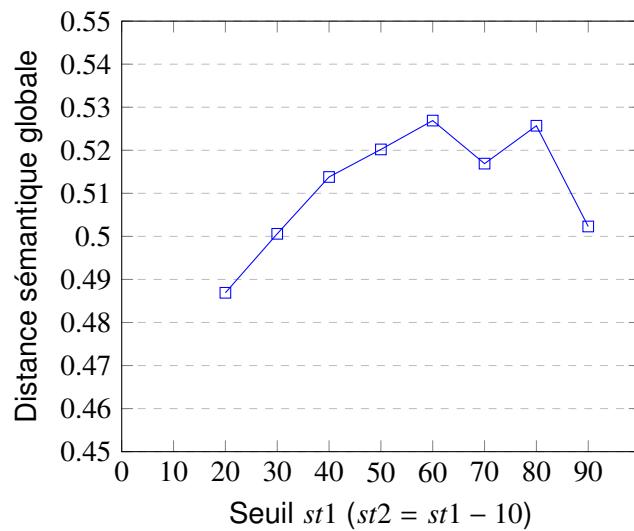
où $|R|$ est le nombre total de relations hiérarchiques, et $(term_i, term_j) \in C_{SHMC}$.

Les résultats présentés dans la sous-section suivante présente la distance sémantique globale obtenue. L'impact des paramètres de hiérarchisation influent grandement sur les relations hiérarchiques, l'impact des valeurs des seuils $st1$ and $st2$ sont donc étudiés.

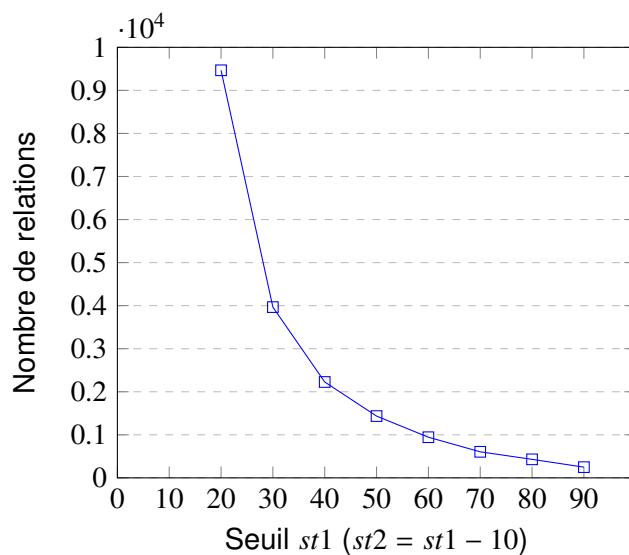
3.6.2.2/ RÉSULTATS

Les seuils $st1$ et $st2$ (voir tableau 3.5) varient respectivement de 0.20 à 0.90, et de 0.10 à 0.80 ($st2 = st1 - 10$). Le pas d'augmentation est de 0.10. La figure 3.9 montre les variation de la distance sémantique globale pour chaque couple de valeurs $st1$ et $st2$. La figure 3.10 montre le nombre de relations hiérarchiques correspondant pour chaque valeur de $st1$.

FIGURE 3.9 – Distance sémantique globale en fonction du seuil $st1$



La distance globale optimale (minimum) est obtenue pour les valeurs $st1 = 20$ et $st2 = 10$. Les résultats semblent indiquer que les relations issues de proportions conditionnelles élevées impactent négativement la qualité globale des relations. Cependant, la différence

FIGURE 3.10 – Nombre de relations hiérarchiques en fonction du seuil $st1$ 

globale reste faible (comprise dans l'intervalle [0.48, 0.53]). La qualité globale des relations est améliorée en parallèle du nombre de relations. Ce point est encourageant : une hiérarchie plus large permet de capturer un nombre de relations plus important sans pour autant dégrader les qualité moyenne des relations sémantiques.

3.7/ CONCLUSION

Cette section a décrit l'approche d'apprentissage d'une taxonomie dans un contexte Big Data, et son intégration au sein d'une ontologie pour la construction d'un modèle de classification hiérarchique multi-label. Les résultats ont démontré la faisabilité de l'approche d'apprentissage de la taxonomie dans un contexte Big Data, et les capacités d'extraction de caractéristiques pertinentes à partir de grands volumes de données. L'évaluation des temps de calcul a montré que le point critique du processus est la construction de la matrice de cooccurrence, dont le coût calculatoire est élevé. La montée en charge horizontale, i.e. l'augmentation des ressources matérielles par ajout du nombre de machines, permet cependant de palier cette limitation, comme indiqué dans la littérature Lin (2013). En outre, des optimisations ont par la suite été apportées au processus (notamment l'ajout d'un combiner), et d'autres solutions ont été proposées pour optimiser cette étape, i.e. la réduction de la fenêtre de voisinage pour la détection des cooccurrences.

Les limitations des méthodes d'extraction de labels et de construction de relations hiérarchiques pertinentes ont été soulignées. Une évaluation du processus pour la tâche de classification est présentée dans le chapitre suivant, afin de compléter ces premiers résultats par une évaluation de la performance de l'approche pour la tâche de classification et comparaison à la littérature.

4

CLASSIFICATION BASÉE SUR UN MOTEUR DE RAISONNEMENT

Ce chapitre présente d'une part la méthode d'apprentissage de règles de classification (résolution), et d'autre part la méthode de classification par raisonnement sémantique (réalisation) du processus SHMC. Suite à la description complète de ces deux composantes du processus, une évaluation du processus pour la tâche de classification est présentée dans la section 4.3. Les éléments surlignés en rouge de l'architecture concernent directement le processus d'apprentissage de règles (la hiérarchie de labels est dans une moindre mesure impliquée dans le processus de classification, voir section 4.2.1).

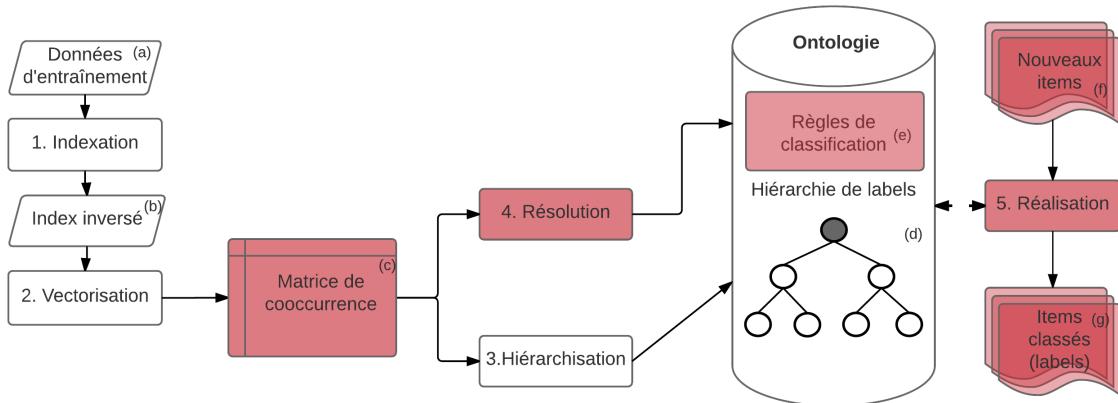


FIGURE 4.1 – Le processus SHMC : résolution et réalisation

4.1/ RÉSOLUTION

La résolution crée les règles de l'ontologie utilisées pour la classification des items par rapport à la taxonomie, à partir de la matrice de cooccurrence de termes issue des données d'apprentissage. L'approche et son implémentation sont décrites dans la sous-section suivante.

Le processus de création des règles utilise une méthode de seuils (Werner et al., 2014) pour sélectionner les termes les plus pertinents pour chaque concept de la hiérarchie, et inclue ces termes dans la définition de la règle. La principale différence avec (Werner et al., 2014) réside dans le fait de transcrire les règles au format SWRL (Semantic Web

Rule Language), plutôt que de les traduire en tant que contraintes logiques de l'ontologie. Le principal intérêt des règles SWRL est de réduire la charge de calcul du raisonneur, donc d'améliorer la performance du système. Un ensemble conséquent de règles SWRL simples (i.e. courtes et peu complexes) est généré lors de cette phase afin de classer les items.

4.1.1/ DÉFINITIONS

Définition 8 : Raisonneur sémantique

Un raisonneur sémantique, ou raisonneur, est un logiciel qui implémente des algorithmes de raisonnement déductif, selon une logique de description donnée. Les raisonneurs et les moteurs d'inférence répondent aux mêmes principes. Les raisonneurs fournissent en revanche un plus grand nombre de mécanismes. Ils font partie intégrante des technologies du web sémantique, et respectent les standard définis par le W3C^a. Les raisonneurs sont notamment intégrés dans différents logiciels de gestion de bases de données de triplets (triple-stores), et permettent d'inférer de nouvelles connaissances à partir d'ontologies existantes contenues dans la base.

a. Le World Wide Web Consortium, ou W3C, est un organisme de standardisation à but non lucratif, chargé de promouvoir la compatibilité des technologies du World Wide Web. <https://www.w3.org/>

Définition 9 : Triple Store

Un triple store est une base de données conçue pour le stockage et la récupération de données RDF^a. Un triplestore ne stocke des triplets de type <Subject, Predicate, Object>, tels que définis dans une logique de description. La base de données représente ainsi un graphe RDF formé de l'ensemble des triplets. Le triple store permet par la suite de requêter le graphe RDF de la même manière qu'une base de données relationnelle. Le langage standard de requête de données RDF est SPARQL^b.

a. Ressource Description Framework, <https://www.w3.org/RDF/>
b. <https://www.w3.org/TR/rdf-sparql-query/>

4.1.2/ DESCRIPTION DE L'APPROCHE

La résolution génère en partie du le modèle prédictif, en déterminant pour chaque label un ensemble de règles de classification, qui constituent les conditions pour qu'un label $label_1$ soit attribué à un item $item_1$. Les règles sont générées à partir de la matrice de cooccurrence définie lors de la vectorisation. Chaque règle est composée d'un ensemble de termes nécessaires pour qu'un item soit classé avec le label correspondant.

Comme indiqué dans la description de l'ontologie noyau (tableau 3.1), les relations *hasAlpha* et *hasBeta* définissent les règles de classification pour chaque label. Deux seuils sont définis pour créer ces relations :

- **Un seuil alpha** (α) tel que $\alpha < P_C(term_i|term_j)$, où $term_i \in Label$ et $term_j \in Term$.
- **Un seuil beta** (β) tel que $\beta \leq P_C(term_i|term_j) \leq \alpha$, où $term_i \in Label$ et $term_j \in Term$.

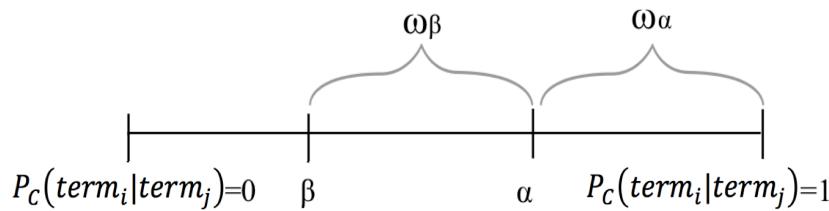


FIGURE 4.2 – Ensembles Alpha et Beta

Ces deux seuils sont définis par l'utilisateur dans l'intervalle $[0, 1]$ avant la création des règles. En fonction des seuils, deux ensembles de termes sont identifiés (figure 4.2) :

- L'ensemble Alpha ($\omega_\alpha^{(term_i)}$) est défini pour chaque label par :

$$\omega_\alpha^{(term_i)} = \{term_j | \forall term_j \in Term : P_C(term_i|term_j) > \alpha\} \quad (4.1)$$

i.e. l'ensemble des termes $term_j$ qui co-occurrent avec $term_i \in Label$, et dont la proportion conditionnelle est supérieure à α .

- L'ensemble Beta ($\omega_\beta^{(term_i)}$) est défini pour chaque label par :

$$\omega_\beta^{(term_i)} = \{term_j | \forall term_j \in Term : \beta \leq P_C(term_i|term_j) \leq \alpha\} \quad (4.2)$$

i.e. l'ensemble des termes $term_j$ qui co-occurrent avec $term_i \in Label$ et dont la proportion conditionnelle est comprise entre les seuils β et α .

Pour chaque terme $term_j$ compris dans $\omega_\alpha^{(term_i)}$, une relation *hasAlpha* est créée entre $term_i$ et $term_j$. De la même façon, pour chaque terme $term_j$ compris dans $\omega_\beta^{(term_i)}$, une relation *hasBeta* entre $term_i$ et $term_j$ est créée.

En fonction de la cardinalité des ensembles Alpha et Beta déterminés pour chaque item, quatre cas sont possibles pour catégoriser un item :

- Beta Vide (1) : $|\omega_\alpha^{(term_i)}| > 0 \wedge |\omega_\beta^{(term_i)}| = 0$
- Alpha vide (2) : $|\omega_\alpha^{(term_i)}| = 0 \wedge |\omega_\beta^{(Label_i)}| > 0$
- Alpha et Beta non vides (3) : $|\omega_\alpha^{(term_i)}| > 0 \wedge |\omega_\beta^{(term_i)}| > 0$
- Alpha et Beta vides (4) : $|\omega_\alpha^{(term_i)}| = 0 \wedge |\omega_\beta^{(term_i)}| = 0$

Pour les trois premières catégories, la création des règles est définie ci-dessous (la quatrième catégorie (4) ne génère aucune règle).

Dans le cas Beta Vide (1), seul l'ensemble ω_α est utilisé pour la création de règles. La condition de classification est alors :

$$\begin{aligned} \forall label \forall item \exists term : hasTerm(item, term) \wedge term \in \omega_\alpha^{label} \\ \rightarrow isClassified(item, label) \end{aligned} \quad (4.3)$$

i.e. si l'item possède au moins un terme de $\omega_\alpha^{(term_i)}$ il est classé avec le terme $term_i$, où $term_i \in Label$. Une règle est créée pour chaque label unique respectant cette condition, puis sérialisée au format SWRL (voir section suivante).

Un exemple de règles SWRL pour le cas Beta Vide est présenté dans le tableau 4.1, pour $|\omega_\alpha^{(term_i)}| = |\{t_1, t_2\}|$.

Dans le cas Alpha vide (2), seul l'ensemble ω_β est utilisé. Les items sont classés avec un label via la condition suivante :

$$\begin{aligned} \forall label \forall item : |\{\forall term : hasTerm(item, term) \wedge term \in \omega_\beta^{label}\}| \geq \delta \\ \rightarrow isClassified(item, label) \end{aligned} \quad (4.4)$$

TABLE 4.1 – Exemples de règle Alpha

Règles Alpha
$Item(?it), Term(?t_1), Label(?t_1), hasTerm(?it, ?t_1) \rightarrow isClassified(?it, ?t_1)$
$Item(?it), Term(?t_2), Label(?t_2), hasTerm(?it, ?t_2) \rightarrow isClassified(?it, ?t_2)$

TABLE 4.2 – Exemples de règle Beta

Règles Beta
$Item(?it), Term(?t_1), Term(?t_2), Label(?t_3), hasTerm(?it, ?t_1), hasTerm(?it, ?t_2) \rightarrow isClassified(?it, ?t_3)$
$Item(?it), Term(?t_1), Term(?t_3), Label(?t_2), hasTerm(?it, ?t_1), hasTerm(?it, ?t_3) \rightarrow isClassified(?it, ?t_2)$
$Item(?it), Term(?t_2), Term(?t_3), Label(?t_1), hasTerm(?it, ?t_2), hasTerm(?it, ?t_3) \rightarrow isClassified(?it, ?t_1)$

i.e. si l’item possède un nombre de termes de $\omega_{\beta}^{(term_i)}$ supérieur ou égal à δ , il est classé avec $term_i$, où $term_i \in Label$. L’ensemble des combinaisons de termes possibles est alors considéré pour la création des règles SWRL : une règle est créée pour chaque combinaison de termes $term_j \in \omega_{\beta}^{(term_i)}$ ou le nombre de termes combinés est $\delta = \lceil |\omega_{\beta}^{(term_i)}| * p \rceil$, et $0 \leq p \leq 0.5$. Par exemple, pour $|\omega_{\beta}^{(term_i)}| = |\{t_1, t_2, t_3\}| = 3$ et $p = 0.5$, tel que $\delta = \lceil 3 * 0.5 \rceil = 2$, les règles SWRL présentées dans le tableau 4.2 sont générées. L’ensemble des règles Beta est la combinaison C_n^m de m termes d’un ensemble de n éléments. Dans notre approche, n est le nombre de termes possibles $|\omega_{\beta}^{(term_i)}|$, et m le nombre minimum de termes δ qui composent chaque règle (par exemple, $C_{20}^{10} = 184756$). Pour limiter le nombre total de règles générées, on fixe $n \leq 10$. Les termes sont sélectionnés par classement dans l’ensemble $\omega_{\beta}^{(term_i)}$, en utilisant la proportion conditionnelle $P_C(term_j | term_i)$ comme fonction de classement.

On note que les règles décrites par un nombre de termes supérieur à δ ne nécessitent pas d’être intégrées au modèle, car redondantes avec la combinaison de δ termes qui suffit à effectuer la classification.

Dans le cas Alpha et Beta non vides (3), les deux ensembles de termes sont considérés pour la création de règles. Les règles alpha et beta alors déduites des deux ensembles sont définis de la même façon que dans les cas (1) et (2). Une modification est cependant faite dans le calcul des combinaisons de termes, car les règles beta sont par définition moins pertinentes que les règles alpha. On définit alors $q = p * 2$, tel que $\delta = \lceil |\omega_{\beta}^{(term_i)}| * q \rceil$, avec $0 \leq q \leq 1$ et $q = p * 2$.

Enfin, le cas Alpha et Beta vide (4) où la cardinalité des ensembles est zéro n’induit pas de création de règles.

L’ensemble des règles définies pour l’ensemble des labels sont sérialisées en SWRL et intégrées à la base de connaissances. La section 4.2 décrit comment les règles sont utilisées pour effectuer la classification.

4.1.3/ IMPLÉMENTATION

La Résolution tire partie du framework MapReduce afin de distribuer les calculs, de la même façon que lors de la construction de la matrice de cooccurrence 3 et de la hiérarchie de labels. La définition des règles de classification pour un concept (label) de l'ontologie est indépendante des autres concepts, ce qui permet de répartir la génération des règles par concept.

On suppose dans cette description que les seuils α et β sont définis par l'utilisateur (les valeurs effectives de α et β seront étudiées dans les expérimentations). Le processus de création de règles est divisé en un sous-processus pour chaque label $label_i \in Label$. Dans chaque sous-processus, les ensembles $\omega_{\alpha}^{(label_i)}$ et $\omega_{\beta}^{(label_i)}$ sont déterminées à partir de la matrice de cooccurrence, puis les règles sont créées indépendamment pour chaque label.

De façon similaire à l'algorithme de subsomption, la méthode de création de règles exploite la matrice de cooccurrence, et est une tâche intensive. La méthode est donc implémentée de façon distribuée via le paradigme mapReduce. Un job MapReduce dédié détermine les règles à partir de la matrice de cooccurrence.

L'ensemble des tuples de la matrice $< (term_i, term_j), P(term_i|term_j) >$ constitue l'entrée de la fonction map. Les couples (*key, value*) sont définis tels que :

- *key* est un tuple $(term_i, term_j)$ où $term_i$ et $term_j$ sont des termes identifiés durant la vectorisation.
- *value* est la proportion $P(term_i|term_j)$

Dans la phase map, les seuils α et β sont appliqués à chaque tuple en entrée $< (term_i, term_j), P(term_i|term_j) >$ où $term_i \in \omega_{IT}$ ou $term_j \in \omega_{IT}$. La fonction map émet une liste de tuples $< (RuleType, label_i), term_j >$ tel que :

- *RuleType* est un descripteur du type de règle (alpha ou beta)
- *label_i* est le label conséquent (tête) de la règle
- *term_j* est un terme du corps de la règle pour le label *label_i*. Il s'agit soit du terme unique d'une règle alpha, soit d'un des termes composant une règle beta.

Les tuples sont triés par clé (i.e. $(RuleType, label_i)$) selon le paradigme MapReduce, puis la fonction reduce regroupe les tuples dont la clé est identique, c'est à dire par label. La fonction reduce émet en sortie un ensemble $\omega_{\alpha}^{(term_i)}$ et $\omega_{\beta}^{(term_i)}$ (potentiellement vides) pour chaque *label_i*. Les règles sont serialisées au format SWRL en utilisant la librairie OWL-API¹ puis intégrées à la base de connaissances.

La section suivante décrit le processus de classification, basé sur le modèle prédictif, i.e. la hiérarchie de labels et les règles de classification (voir figure 4.1).

4.2/ RÉALISATION

La réalisation effectue la classification de nouveaux items selon le modèle prédictif, généré à partir de toutes les étapes précédentes du processus SHMC. Le modèle final est composé de la hiérarchie de labels issue de la hiérarchisation, et des règles de classification issues de la résolution. Une ontologie regroupe l'ontologie noyau définie dans la section 3, et le modèle prédictif. Cette ontologie est intégrée dans un triple store pour effectuer la classification de nouveaux items au sein de l'ontologie.

1. <http://owlapi.sourceforge.net/>

La réalisation peuple l'ontologie avec les nouveaux items, et effectue leur classification en accord avec la taxonomie et les règles de classification.

L'ontologie est d'abord peuplée avec les nouveaux items au niveau Abox (assertion). Les règles SWRL générées lors de la résolution sont utilisées afin d'associer des étiquettes aux items. Le moteur d'inférence utilise ensuite les règles et la hiérarchie pour inférer les concepts correspondants aux caractéristiques des items pour les classer. La classification des items est ainsi multi-étiquette et hiérarchique.

4.2.1/ DESCRIPTION DE L'APPROCHE

La réalisation est divisée en deux sous-étapes : le peuplement de l'ontologie en tant qu'instances de la classe *Item* (Abox), puis leur classification en fonction de leurs attributs (termes) et du modèle prédictif.

Chaque item est d'abord décrit par l'ensemble des termes extraits de celui-ci lors de son indexation. Le processus d'indexation des nouveaux items est similaire à l'indexation des items des données d'entraînement. Chaque item est décrit par un ensemble de termes pertinents $\omega_{\gamma}^{(item_i)}$ tel que :

$$\omega_{\gamma}^{(item_i)} = \{term_j | \forall term_j \in Term \wedge \gamma < tfidf_{(item_i, term_j, C)}\} \quad (4.5)$$

où γ est le **seuil de pertinence**, $\gamma < tfidf_{(item_i, term_j, C)}$, $term_j \in Term$, $item_i \in Item$. La valeur de $tfidf$ est calculée de la même façon que lors de la vectorisation (voir section 3.2).

L'étape de classification attribue à chaque item les labels appropriées en fonction du modèle prédictif décrit par la hiérarchie et les règles de classification. Des implémentations de raisonneurs sémantiques basés sur des algorithmes d'inférence (tableau, résolution) tels que Pellet Sirin et al. (2007), FaCT++ Tsarkov et al. (2006), ou Hermit Shearer et al. (2009) permettent d'effectuer une inférence complète des axiomes implicites pour des ontologies ayant une expressivité élevée, telle que OWL2 SROIQ(D). Les travaux de Werner (2015) ont montré que dans un contexte réel où le nombre de règles est élevé, ces moteurs ne permettent pas de passer l'échelle et de gérer un volume important de données.

Notre approche consiste à utiliser le raisonnement sur des règles de Horn, dont l'expressivité est limitée mais permet un passage à l'échelle dans un contexte Big Data. Le raisonnement sur des règles applique de façon exhaustive un ensemble de règles à un ensemble de triplets, pour inférer les axiomes implicites Urbani et al. (2011). Dans le cas de la tâche de classification, ces axiomes sont les classifications (labels) des items représentés par un ensemble d'attributs (triplets).

Le moteur d'inférence infère les labels les plus spécifiques de la hiérarchie de labels pour chaque nouvel item. Les règles de classification et la hiérarchie de subsomption sont utilisées par le raisonneur pour cette tâche. Le résultat est une classification hiérarchique multi-label des items. Pour inférer l'intégralité des labels en fonction de la hiérarchie de labels, la règle SWRL suivante est ajoutée à l'ontologie :

$$\begin{aligned} & Item(?item), Label(?labelA), Label(?labelB), broader(?labelA, ?labelB), \\ & isClassified(?item, ?labelA) \rightarrow isClassified(?item, ?labelB) \end{aligned} \quad (4.6)$$

Les règles de classification peuvent ensuite être appliquées selon deux types d'inférence, i.e. le chaînage avant (matérialisation) ou chaînage arrière. À ces deux types d'inférence

correspondent deux stratégies de classification : la classification avant la requête et la classification au moment de la requête.

La classification avant la requête (figure 4.3) est effectuée par un moteur d'inférence qui détermine par chaînage avant l'inférence de toutes les assertions implicites² de la base de connaissance (matérialisation). Les règles d'inférence sont appliquées sur la totalité de l'ontologie, jusqu'à ce que toutes les assertions soient matérialisées dans la base. Une fois la matérialisation terminée, les requêtes effectuées sur la base permettent de retrouver les classifications rapidement. Cette méthode est peu adaptée dans un système où l'ontologie change fréquemment (le raisonnement doit alors être ré-exécuté entièrement).

La classification au moment de la requête est effectuée par le moteur d'inférence par chaînage arrière (figure 4.4), qui applique les règles uniquement sur les données concernées par la requête. Contrairement au chaînage avant, le raisonneur est activé par la requête. Le moteur applique les règles uniquement sur les axiomes nécessaires pour répondre à la requête et déterminer de la classification. L'inconvénient est que les requêtes demanderont un temps d'exécution relativement long, indépendamment de leur fréquence ou de leur complexité.

Appliquer les règles uniquement sur une partie des données, i.e. les items à classer, représente un avantage dans un contexte Big Data, en particulier gérer la vitesse des données. En revanche, le désavantage est la nécessité d'activer le moteur d'inférence à chaque requête, qui est impacté par le volume et l'expressivité des règles.

Ces deux stratégies d'inférence sont compatibles avec le processus SHMC, cependant il faut adopter l'une ou l'autre en fonction du cas d'utilisation.

4.2.2/ IMPLÉMENTATION

L'implémentation de la réalisation est dépendante du triple store donnant accès à l'ontologie et aux individus (items). L'application des règles de classification par raisonnement sur l'ontologie lors de la requête (query time) est coûteux en temps de calcul. Les deux méthodes de raisonnement décrites ci-dessus ne sont pas implémentées dans tous les triples stores (AllegroGraphe, Stardog, Virtuoso...). Aussi, tous les triples stores ne sont pas libres de droit.

Dans notre cas d'utilisation, l'ajout de nouveaux items doit permettre leur classification, i.e. nous nous intéressons essentiellement au cas de la classification d'items à un instant t donné. La solution appropriée est donc d'utiliser un raisonnement par chaînage arrière. Des travaux récents s'intéressent au passage à l'échelle des méthodes basées sur le chaînage arrière, par optimisation des requêtes (notamment par réécriture des requêtes) d'une part et par utilisation du framework MapReduce d'autre part Urbani et al. (2012). Ces méthodes peuvent grandement améliorer la performance de notre approche, néanmoins elles ne sont pas intégrées dans la plupart des triple stores commerciaux. Nous ne considérons donc pas leur utilisation dans ces travaux.

En réponse aux problèmes de performance du chaînage arrière identifiés dans Farias et al. (2015), une approche de sélection des règles a été intégrée au processus de classification pour restreindre le raisonnement aux règles strictement nécessaires à l'exécution

2. I.e. le calcul de la fermeture de l'ontologie, ou "ontology closure"

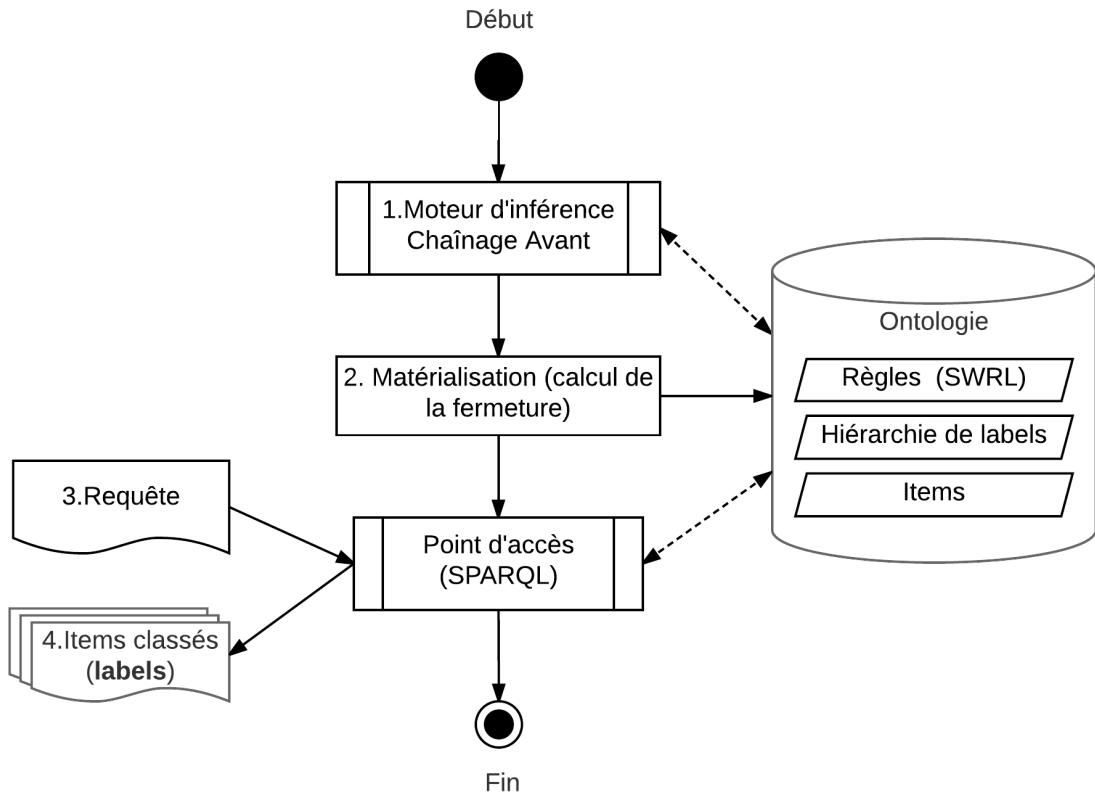


FIGURE 4.3 – Classification avant la requête et chaînage avant

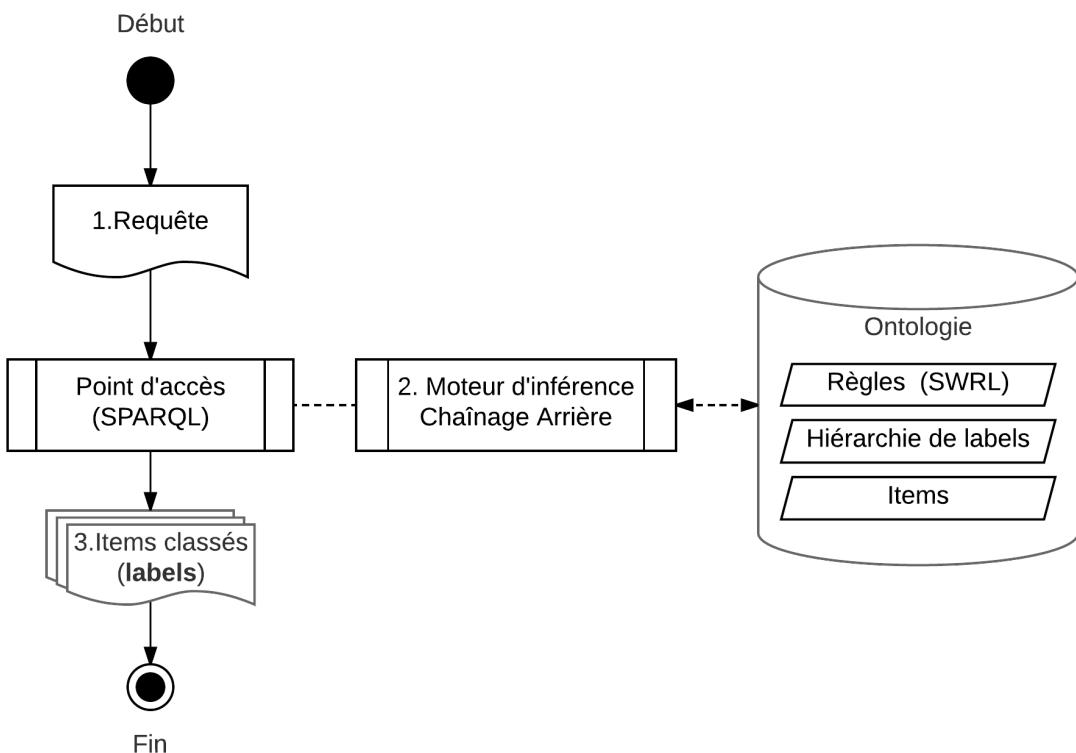


FIGURE 4.4 – Classification au moment de la requête et chaînage arrière

de chaque requête. Deux types de requête sont identifiés : (1) retrouver tous les items classés pour un label fixé, et (2) retrouver tous les labels correspondant à un item fixé.

Pour retrouver tous les items classés avec un label $label_i$ (1), seules les règles où le conséquent (tête) est $label_i$ sont considérées (i.e. $isClassified(?item, label_i)$) et activées dans l'ontologie, selon la méthode décrite dans Farias et al. (2015). Pour le second type de requête, seules les règles où au moins un terme $term_i \in \omega_{\gamma}^{(item_i)}$ dans le corps de la règle (i.e. $(hasTerm(?item, term_i))$) sont activées (toutes les règles ou un des termes extraits de l'item apparaît).

La librairie OWL-API est utilisée pour la conversion des items et de leurs attributs et leur insertion dans l'ontologie (Abox). Un triple-store performant pour de grands volumes de données nommé Stardog (<http://docs.stardog.com>) est utilisé pour intégrer l'ensemble des éléments composant l'ontologie, i.e. le modèle prédictif et les instances d'items (Tbox+Abox). Stardog est également utilisé pour effectuer la classification par chaînage arrière, Stardog étant compatible avec l'inférence sur des règles SWRL et le chaînage arrière. L'algorithme de sélection de règles est développé en java, qui interagit avec Stardog par requêtes pour optimiser les performances de la classification.

4.3/ ÉVALUATION

Cette section décrit les évaluations introduites dans le chapitre précédent, i.e. l'analyse de la performance de l'approche SHMC pour la tâche de classification hiérarchique multi-label. Trois évaluations sont présentées : une preuve de concept qui décrit de façon indicative le fonctionnement du processus, une évaluation de la tâche de classification dans un contexte Big Data non-supervisé, et une évaluation qualitative en contexte supervisé, qui permet la comparaison de l'approche à la littérature.

4.3.1/ MÉTHODE D'ÉVALUATION DE LA CLASSIFICATION

Pour l'évaluation de la tâche de classification en contexte supervisé et non-supervisé, des métriques standard basées-étiquette³ sont utilisées pour évaluer l'approche Semantic HMC : la micro précision moyenne, et le micro rappel (recall) moyen. Ces mesures sont calculées de la même façon que Madjarov et al. (2012) :

$$Precision_{micro} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad (4.7)$$

$$Rappel_{micro} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \quad (4.8)$$

et la macro précision moyenne / le macro rappel (recall) moyen, tel que :

$$Precision_{macro} = \frac{1}{n} \sum_i^n \left(\frac{TP_i}{TP_i + FP_i} \right) \quad (4.9)$$

3. Les méthodes basées étiquette calculent la précision, rappel et F-mesure pour chaque label puis effectuent une moyenne globale, en opposition aux méthodes basées documents/items, où le calcul local est effectué par document puis moyenné sur l'ensemble des documents

$$Rappel_{macro} = \frac{1}{n} \sum_i^n \left(\frac{TP_i}{TP_i + FN_i} \right) \quad (4.10)$$

où TP_i est le nombre de vrais positifs, FP_i le nombre de faux positifs, et FN_i le nombre de faux négatifs pour le $Label_i$. Une métrique d'évaluation complémentaire est la F1-mesure, i.e. la moyenne harmonique de la précision et du rappel où le poids de chaque composante est égal. Les micro et macro F1-mesure sont définies de façon similaires par la relation :

$$F1 = \frac{2 * (Precision * Rappel)}{(Precision + Rappel)} \quad (4.11)$$

4.3.2/ ÉVALUATION DU PASSAGE À L'ÉCHELLE

Cette sous-section présente l'évaluation préliminaire du processus, qui analyse ses performances dans un contexte Big Data. L'environnement de test, le jeu de données utilisés et les paramètres spécifiques à l'évaluation sont décrits, suivis des résultats expérimentaux obtenus.

4.3.2.1/ JEUX DE DONNÉES

Le jeu de données utilisé dans cette évaluation est identique au jeu de données présenté dans le chapitre précédent (section 3.5.1). Il s'agit d'articles issus de la version française de Wikipedia (textes non structurés). Le jeu de données est composé de 3 parties de taille variable, pour étudier l'impact de la taille de la taille des données sur le processus. Il s'agit des 3 premières parties du jeu de données utilisé dans le chapitre précédent. Les résultats du chapitre précédent ont montré que le point critique du processus est la vectorisation, nous avons donc limité cette étude aux 3 premières parties du jeu de données à des fins de simplification. En outre, l'algorithme de résolution a la même complexité que l'algorithme de hiérarchisation, et la réalisation dépend grandement de la technologie utilisée et des ressources disponibles. Pour ces raisons, les temps de calcul ne rentrent pas dans le cadre de cette évaluation.

Comme indiqué dans le chapitre précédent, le processus est dirigé par un ensemble de seuils dont l'impact sur la génération du modèle prédictif est important. Les valeurs de seuils utilisées dans cette évaluation sont identiques aux paramètres présentées dans le tableau 4.3 (section 3.5.4). Les valeurs restent fixes pour les différents jeux de données. La matrice de cooccurrence et la hiérarchie de labels résultant des expérimentations décrites en section 3.5.4 sont utilisés comme données d'entrée pour ce processus. Les spécificités de ces données d'entrée sont présentées dans le tableau 4.4. Les données générées par les phases précédentes correspondent aux jeux de données de cette évaluation.

4.3.2.2/ RÉSULTATS

L'objectif de ces premiers résultats est d'observer le passage à l'échelle des deux dernières étapes du processus : résolution et réalisation. On étudie alors le nombre de règles de classification issues de la résolution (1), et le nombre de relations *isClassified* (classi-

TABLE 4.3 – Paramètres par défaut

Paramètre	Étape	Valeur
Seuil Alpha	Résolution	90
Seuil Beta	Résolution	80
Seuil de classement termes (n)	Résolution	5
p	Résolution	0.25
Seuil de pertinence (γ)	Réalisation	2

TABLE 4.4 – Description du modèle initial (pré-résolution)

Jeu de données	Wiki 1	Wiki 2	Wiki 3
Nombre de termes	10973	13053	23859
Nombre de labels	3680	1981	1545
Nombre de relations	10765	2754	1315

fications) issues de la réalisation (2). Pour les tests de classification (2), le jeu de données de test utilisé pour la classification est identique au jeu de données d'entraînement.

Le chapitre précédent a montré que le nombre de labels décroît en fonction de la taille des données, dû aux seuils fixes de la hiérarchisation. Le nombre de règles étant fonction du nombre de labels, la figure 4.5 montre que le nombre de règles (α et β) décroît également.

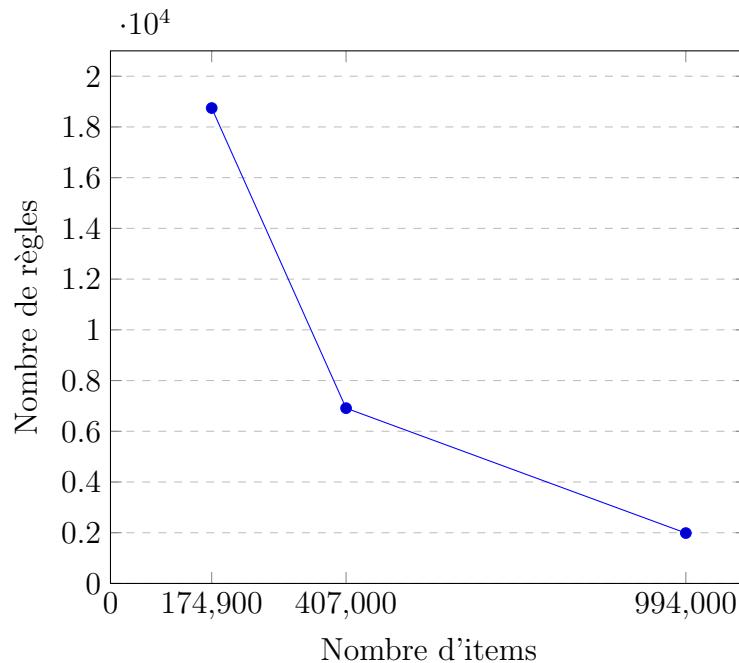


FIGURE 4.5 – Nombre de règles en fonction du jeu de données

Le nombre de classifications est présenté dans la figure 4.6. On observe que le triple store Stardog effectue un nombre croissant d'inférences (classifications) en fonction du nombre d'instances (items) ajoutés à l'ontologie, conformément à nos attentes.

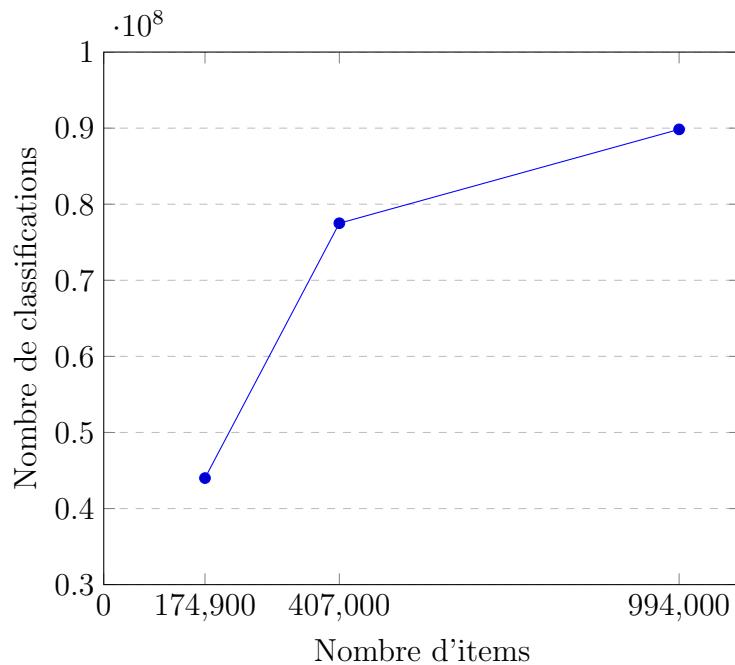


FIGURE 4.6 – Nombre de classifications (relations *isClassified*) en fonction du jeu de données

4.3.3/ ÉVALUATION QUALITATIVE EN CONTEXTE NON-SUPERVISÉ

Les figures 4.7.a et 4.7.b montrent la variation de la précision globale, rappel global et F1-mesure globale quand les paramètres de la hiérarchisation *st1* et *st2* varient. Chaque point correspond à une valeur de *st1* et *st2* représentés respectivement sur en abscisse inférieure et supérieure. La figure 4.7.c montre la variation de la F1-mesure globale lorsque *st1* et *st2* varient simultanément. De façon similaire, les figures 4.8.a et 4.8.b montrent la variation en Précision, Rappel, et F1-mesure en fonction des seuils de la Résolution *alpha* et *beta*.

Les seuils de hiérarchisation et de résolution impactent respectivement le nombre de labels dans la hiérarchie et le nombre de labels possédant au moins une règle. Pour illustrer l'impact des seuils, les figures 4.7.c montre l'impact des seuils de hiérarchisation sur la hiérarchie de labels, et la figure 4.8.c, montre l'impact des seuils de résolution sur les règles de classification. Le nombre de labels est normalisé pour permettre la comparaison à la F1-mesure.

4.3.3.1/ VARIATION DES SEUILS *st1* ET *st2*

Dans la figure 4.7, les seuils de résolution sont fixés à leur valeur par défaut (tableau 4.3), i.e *alpha* = 90 et *beta* = 10. La figure 4.7.b montre que le seuil *st1* impacte fortement la précision globale qui augmente grandement en fonction du seuil *st1*. Le seuil *st2* a en revanche un impact limité sur la précision ou le rappel (figure 4.7.a). A partir des figures 4.7.a et 4.7.c, on observe une corrélation entre le gain de précision et le nombre de labels dans la hiérarchie : le nombre réduit de labels semble impacter la précision et par extension la F1-mesure. Par conséquent, le gain de précision ne semble pas être

pertinent.

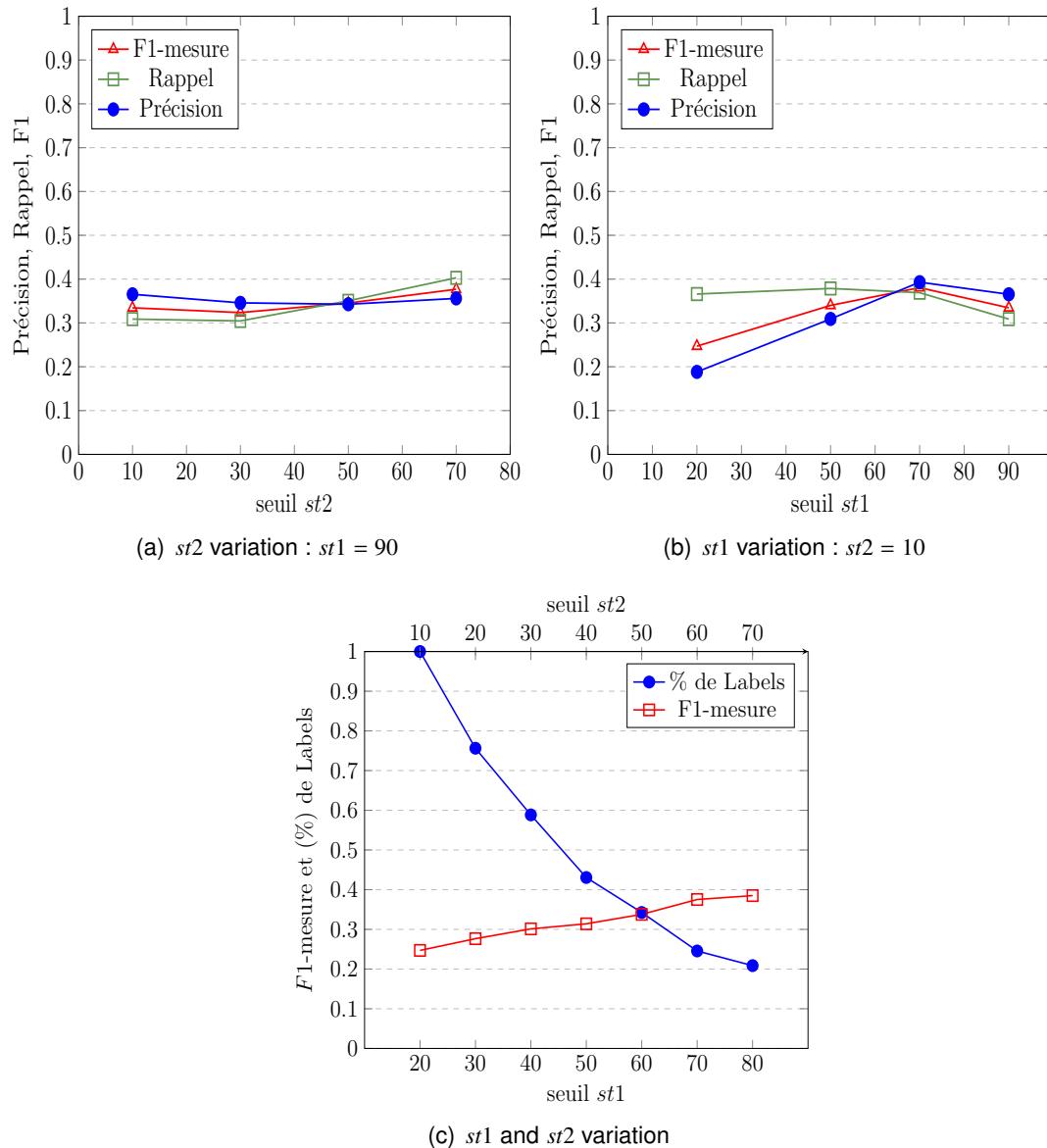
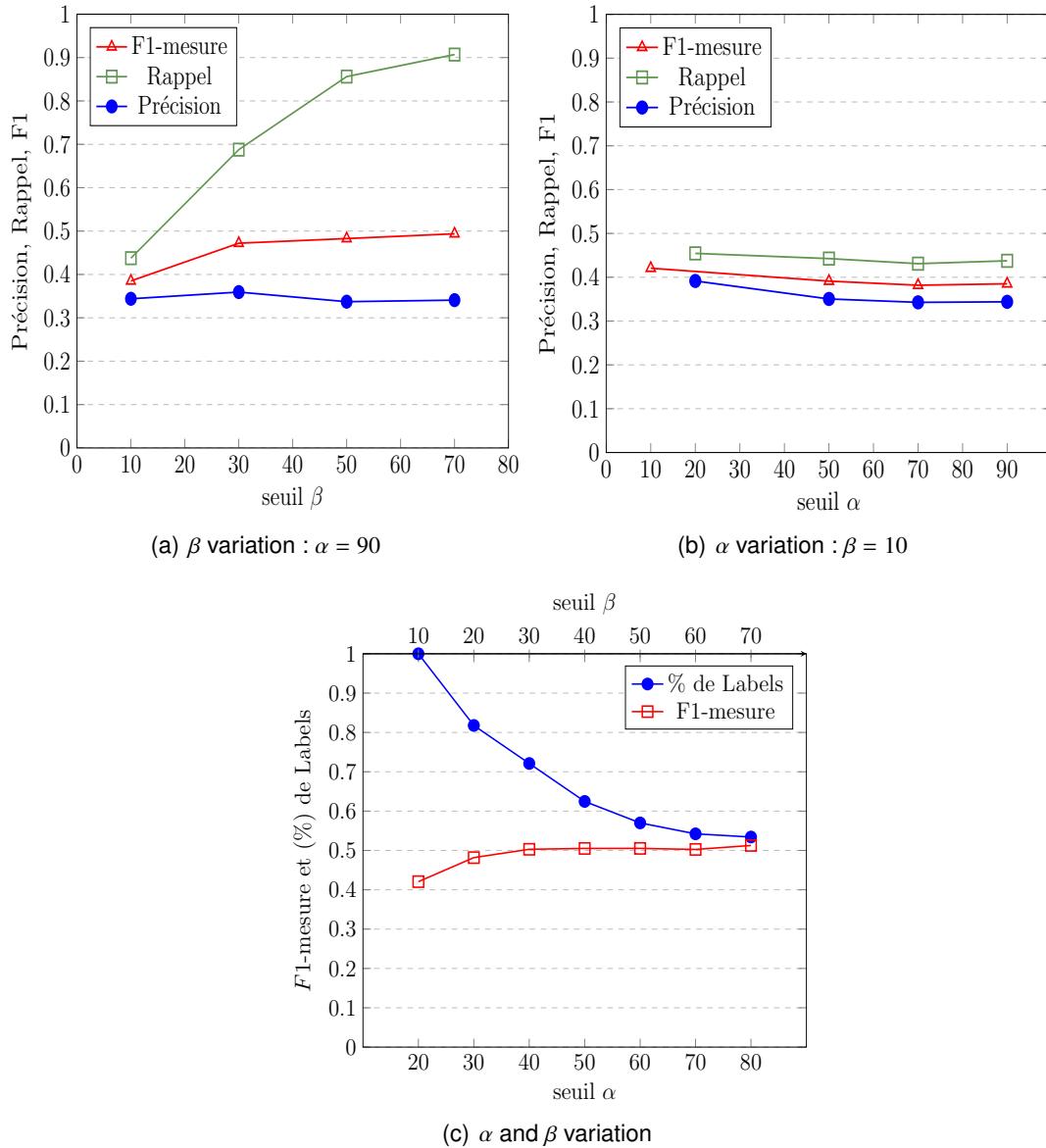


FIGURE 4.7 – Variation des seuils de hiérarchisation. $\text{Alpha} = 90$, $\text{Beta} = 10$ (valeurs par défaut)

4.3.3.2/ VARIATION DES SEUILS DE RÉSOLUTION

La figure 4.7.c montre que la F1-mesure globale maximale est 0.512, obtenue pour $st1 = 80$, $st2 = 70$. Pour observer l'impact des seuils de résolution, on fixe les seuils $st1$ et $st2$ à ces valeurs optimales. La figure 4.8 présente l'évolution du nombre de labels possédant au moins une règle, comparé au la F1-mesure globale lors de la variation des seuils de résolution. Chaque point correspond à une valeur de α et β sur l'axe des abscisses supérieur et inférieur. La figure 4.8.c montre l'évolution de la F1-mesure lorsque les deux seuil croient simultanément : l'évolution de la F1-mesure est alors similaire variation de $st1$

FIGURE 4.8 – Variation des seuils de résolution. $st1 = 80$, $st2 = 70$ (valeurs par défaut)

et $st2$ présentée dans la figure 4.7.c. Le nombre de labels avec des règles de classification décroît de façon linéaire alors que la F1-mesure augmente. L'impact de la variation de F1-mesure est cependant moins significative que lors de la variation de seuils $st1$ et $st2$. En outre, la figure 4.8.c montre un gain de F1-mesure important suite à la première variation ($\beta = 20$ et $\alpha = 30$). Le gain en F1-mesure correspond à une perte de 20% de la proportion des labels avec au moins une règle de classification. Les variations suivantes n'améliorent pas significativement la F1-mesure, alors que le nombre de labels avec des règles décroît.

4.3.3.3/ INTERPRÉTATIONS

D'après nos observations, les labels ayant une fréquence élevée dans le corpus tendent à produire des classifications plus précises, d'où une F1-mesure plus importante pour des valeurs plus hautes des seuils. Les seuils de hiérarchisation ($st1$ et $st2$) ont plus d'impact sur la précision de la classification, tandis que les seuils de résolution (α et β) ont plus d'impact sur le rappel.

Une conclusion subjective de ces résultats est que maximiser la F1-mesure globale ne semble pas être la meilleure façon d'optimiser le modèle prédictif : l'analyse exhaustive de la variation des seuils effectuée est efficace pour maximiser les mesures employées (précision, rappel, F1-mesure), mais ne tient pas compte des répercussions importantes sur le modèle prédictif. En effet, le coût de l'amélioration de la F1-mesure induit une baisse importante du nombre de relations hiérarchiques, et de règles de classification. En particulier, le gain en F1-mesure lors de l'augmentation des seuils de résolution est faible comparativement à la baisse du nombre de labels avec des règles.

La perte de relations hiérarchiques et de règles de classification induit nécessairement un manque d'options pour classer un item. Un équilibre doit être trouvé entre la maximisation des mesures, et un nombre de règles / relations hiérarchiques adapté. Cet équilibre est dépendant du cas d'utilisation réel. Une mesure de pondération, similaire à la F1-mesure mais appliquée également aux cardinalités des relations hiérarchiques et des règles pourrait permettre de trouver un tel équilibre.

4.3.4/ ÉVALUATION QUALITATIVE EN CONTEXTE SUPERVISÉ

Cette section présente une évaluation qualitative du processus Semantic HMC, qui étend les résultats précédents, et se concentre sur la précision pour la tâche de classification dans un contexte supervisé. L'objectif de cette évaluation est de comparer l'approche Semantic HMC avec l'état de l'art en classification hiérarchique multi-label. En effet, en raison du jeu de données utilisé, l'évaluation qualitative en contexte non-supervisé présentée ci-dessus ne permet pas de comparer directement l'approche à la littérature.

Pour comparer l'approche à la littérature, deux jeux de données d'apprentissage supervisé issues du projet Mulan⁴ sont utilisés dans cette évaluation. Le jeu de données Delicious est composé de données textuelles pré-étiquetées issues du site de bookmarking collaboratif du même nom (Tsoumakas et al. (2008)). Le choix de ce jeu de données se base sur le fait qu'il contient très peu de caractéristiques (500 mots) comparé au nombre d'étiquettes (983), ce qui rend la classification difficile (Papanikolaou et al. (2015)). Le jeu de données est divisé en un jeu d'entraînement (12910 items) utilisé pour apprendre l'ontologie, i.e. la hiérarchie d'étiquettes et les règles, et un jeu de test (3181 items).

Le jeu de données Bibtex est composé de données textuelles pré-étiquetées issues de métadonnées d'articles scientifiques au format bibtex(Tsoumakas et al. (2008)). Les résultats obtenus avec ce jeu de données sont inédits. Le motif pour l'utilisation de ce jeu de données est la différence importante avec le jeu de données Delicious par rapport au ratio de caractéristiques (features, i.e. termes) sur le nombre de classes (étiquettes). Comparativement au jeu de données Delicious, le nombre de caractéristiques est élevé comparé au nombre de classes (1836 caractéristiques pour 159 classes/étiquettes). Le

4. <http://mulan.sourceforge.net/datasets-mlc.html>

jeu de données est divisé en un jeu d'entraînement (4880 items) et un jeu de test (2515 items).

4.3.4.1/ DESCRIPTION

Les paramètres (seuils) du processus Semantic HMC peuvent avoir un impact important sur les résultats. Les seuils $st1$ et $st2$ de la hiérarchisation influent sur la création de relations hiérarchiques entre paires de concepts, tandis que les seuils alpha et beta de la résolution influent sur la création des règles de classification. Des seuils bas engendrent un nombre de règles/relations hiérarchiques élevé au détriment de leur qualité, et inversement. Une étude exhaustive de l'impact des combinaisons de seuils possibles sur les métriques permet de déterminer les seuils les plus appropriés en fonction des résultats obtenus. Le tableau 4.5 décrit l'ensemble des valeurs utilisées pour l'évaluation ci-après. Pour comparer de façon objective les résultats des deux jeux de données, les paramètres sont identiques dans les deux tests de l'évaluation qualitative.

TABLE 4.5 – Valeurs des paramètres pour l'évaluation

Paramètre	Étape	Valeur
Seuil $st1$ (top)	Hiérarchisation	50
Seuil $st2$ (bottom)	Hiérarchisation	40
Seuil Alpha	Résolution	20
Seuil Beta	Résolution	10
Classement des termes (n)	Résolution	5
p	Résolution	0.25
Seuil de Pertinence (γ)	Réalisation	2

4.3.4.2/ RÉSULTATS

La figure 4.9 est un exemple de relations hiérarchiques obtenues à partir du jeu de données Delicious, qui illustre le résultat de la phase de hiérarchisation appliquée à un ensemble de classes prédéfinies.

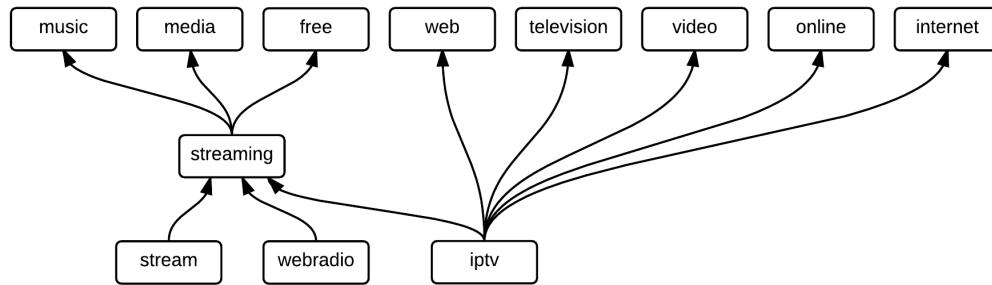


FIGURE 4.9 – Portion de la hiérarchie de labels automatiquement générée à partir du jeu de données Delicious

Le tableau 4.6 montre les résultats de l'évaluation pour les métriques standard basées étiquettes sur les deux jeux de données. La moyenne harmonique (F1-mesure) est également reportée et sera ensuite utilisée pour comparer les résultats à d'autres approches.

TABLE 4.6 – Résultats de l'évaluation (Delicious/Bibtex)

Jeu de données	Pondération	Précision	Rappel	F1-mesure
Delicious	Micro	0.284	0.740	0.410
	Macro	0.068	0.178	0.098
Bibtex	Micro	0,199	0,741	0,314
	Macro	0,188	0,444	0,264

TABLE 4.7 – Comparaison aux approches de la littérature (Delicious/Bibtex)

Algorithme	Delicious		Bibtex	
	Macro F1	Micro F1	Macro F1	Micro F1
SHMC	0.098	0.410	0,264	0,314
CGS_p	0.104	0.297	0.258	0.363
TNBCC	0.088	N/A	0.189	N/A
Path-BCC	0.084	N/A	0.212	N/A
BR	0.096	0.234	0.307	0.457
CC	0.100	0.236	0.316	0.462
HOMER	0.103	0.339	0.266	0.429
ML-kNN	0.051	0.175	0.065	0.206
RFML-C4.5	0.142	0.269	0.016	0.123
RF-PCT	0.083	0.248	0.055	0.230

Le tableau 4.7 compare les valeurs de macro F1-mesure et micro F1-mesure obtenues avec différentes approches de l'état de l'art décrites dans Madjarov et al. (2012), Sucar et al. (2014), Papanikolaou et al. (2015). Ces approches ont été décrites dans façon exhaustive dans le chapitre 2.

Pour le jeu de données Delicious, le tableau 4.7 montre que l'approche Semantic HMC est légèrement plus performante que les autres approches pour la micro F1-mesure, et que la macro F1-mesure est comparable aux autres approches. Les résultats sur le jeu de données Bibtex montrent que l'approche est moins performante que les meilleures approches de l'état de l'art (Binary relevance, Classifier Chain et HOMER). Comparativement aux approches considérées, SHMC reste au niveau de la moyenne de macro F1-mesure (0.19) et de micro F1-mesure (0.32).

4.4/ CONCLUSION

Cette section a présenté une approche d'apprentissage de règles de classification à partir dans un contexte Big Data, l'intégration de ces règles dans un modèle prédictif au sein d'une ontologie, et l'utilisation du modèle prédictif pour la tâche de classification hiérarchique multi-étiquette. Le processus est décrit de façon exhaustive, et est évalué pour la tâche de classification. Une première évaluation a montré la faisabilité de l'approche, qui pallie notamment les limitations techniques des raisonneurs DL identifiées par Werner (2015), et permet la classification d'importants volumes de données en temps raisonnable.

La performance du modèle prédictif généré par l'approche a ensuite été évaluée à la fois en contexte non-supervisé et en contexte supervisé, et comparé à des approches de la littérature dans le second cas. L'évaluation qualitative en contexte non supervisé basée sur la méthode de validation croisée a montré des performances encourageantes de l'approche proposée. L'évaluation permet en outre d'optimiser le paramétrage du processus en fonction de la performance du modèle généré, ce qui est essentiel pour l'optimisation du processus et l'amélioration continue du modèle.

Suivant les critères sélectionnés, l'évaluation qualitative en contexte supervisé montre que la performance l'approche SHMC est comparable aux autres approches de l'état de l'art pour la tâche de classification multi-étiquette. Elle est plus performante que les autres approches selon certains critères d'évaluation pour le jeu de données Delicious. L'orientation de traitement massif de données de l'approche SHMC conserve des performances proches de l'état de l'art pour la tâche de classification supervisée, ce qui montre les avantages de l'approche.



RECHERCHE D'INFORMATION SUR LE WEB PAR CLASSIFICATION SÉMANTIQUE

5

ETAT DE L'ART : RECHERCHE D'INFORMATION SUR LE WEB

Ce chapitre s'intéresse aux méthodes existantes adaptées aux tâches de veille stratégique identifiées dans le premier chapitre, pour lesquelles les outils existants de veille stratégique sont inadaptés, notamment le croisement et la découverte d'information (voir section 1.3.3). Ces méthodes s'inscrivent à l'intersection de deux domaines, la recherche d'information (RI) et la fouille du web.

La fouille du web (web mining) concerne l'application de méthodes ou techniques issues du champ de la fouille de données dans un contexte web. Les objectifs sont semblables aux objectifs courants du data mining : découverte d'informations pertinentes / de valeur, ou de motifs récurrents dans les données. Le web mining est divisé en trois catégories : l'analyse du graphe web (web graphe mining), l'analyse des comportements utilisateurs (web usage mining), et l'analyse du contenu d'items web (web content mining) Chau et al. (2003).

L'analyse du graphe web s'intéresse à l'exploitation des liens hypertextes entrants et sortant des documents web. L'application principale de cette analyse est la construction d'index et de moteurs de recherche Kleinberg (1999), à partir d'algorithmes tels que le PageRank Page et al. (1999).

L'analyse des comportements utilisateur concerne l'exploitation des logs utilisateurs, et est en général restreinte à un site web spécifique Chau et al. (2003). L'analyse de ces logs permet de découvrir des motifs, pouvant être utilisés pour améliorer l'expérience utilisateur, par exemple pour proposer du contenu spécifique à ses besoins dans le cadre de systèmes de recommandation Werner (2015).

Enfin, l'analyse de contenu (web content mining) web est basée sur l'exploitation du contenu des pages web de formats variés (texte mais aussi contenu structuré, métadonnées, images, vidéos, etc.). L'analyse de contenu web permet alors la recherche d'information sur le web. Les objectifs de la recherche d'information sur le web sont entre autres l'extraction d'informations pertinentes Tolle et al. (2000), la découverte de ressources Chakrabarti et al. (1999) Cho et al. (1998) et la catégorisation (classification, clustering) de documents web Kohonen et al. (2000).

De par la nature de la recherche d'information sur le web, les techniques et les méthodes issues du domaine de la recherche d'information et de la fouille de données y sont applicables, en les adaptant aux spécificités du web. La section suivante s'intéresse aux méthodes couramment employées dans le domaine de la fouille de données sur le web,

ainsi qu'aux problématiques techniques spécifiques à ce domaine. Une seconde section s'intéresse à un ensemble de techniques spécifiques, les crawlers ciblés, identifiés comme une solution existante aux problématiques abordées ci-dessus.

5.1/ FONDEMENTS DE LA RECHERCHE D'INFORMATION

La recherche d'information automatique sur le web est un cas particulier du domaine de la Recherche d'Information. La Recherche d'Information étudie les méthodes permettant à un utilisateur de trouver une information pertinente parmi une large collection de documents texte¹ Manning et al. (2008) Liu (2007).

La Recherche d'Information concerne la découverte mais aussi l'organisation, le stockage, et la distribution de l'information. La majorité de ces systèmes considèrent le document (item) comme unité de recherche, et une collection (corpus) comprenant l'ensemble des documents. L'objectif est de répondre à des requêtes utilisateurs qui souhaitent retrouver un ou plusieurs items. Les items les plus pertinents pour une requête donnée sont alors retournés à l'utilisateur, et un classement des items selon leur pertinence peut être effectué. Certains types de requêtes sont récurrents Liu (2007) :

- Requête mot-clé : l'utilisateur fournit un ou plusieurs mots clés représentatifs des items à rechercher. Ces mots clés peuvent être normalisés, par une chaîne de traitements pour permettre une recherche plus efficiente dans l'index. Dans le cas où plusieurs mots clés composent la requête, la plupart des systèmes considèrent par défaut un classement des items retournés en fonction du nombre de termes correspondant à la requête. Dans certains systèmes, l'ordre des termes peut également avoir un impact sur le classement.
- Requête booléenne : une extension des requêtes mots clés consiste à moduler ces requêtes par des opérateurs booléens ("et", "ou" et "non" booléens) pour raffiner la requête. Les items correspondant aux conditions booléennes sont alors retournés.
- Requête par expression (séquence de mots) : ce type de requête prend en entrée une séquence de mots complète. La convention de syntaxe est d'entourer l'expression par des doubles cotes (exemple : "phrase à chercher"). Les items retournés sont ceux où la séquence complète apparaît (recouvrement exact).
- Requête par proximité : une requête par proximité est une version relaxée d'une requête par expression, où la distance entre les termes de la requête dans le document est proportionnelle au score d'un item. Certains systèmes proposent de définir dans la requête une distance maximale entre les termes de la requête.
- Requête-item (full-document) : la requête est construite à partir d'un item existant dans le système de recherche d'information, où l'item est par exemple stocké dans une base de données. La requête est alors composée des termes extraits de cet item lors de son indexation, ou d'un sous-ensemble de ces termes.
- Question en langage naturel : ce type de requête traite des questions en langage naturel émises par l'utilisateur. La question est alors interprétée et transformée en requête.

Le résultat de la requête est l'ensemble des items pertinents contenus dans l'index. La majorité des systèmes de recherche attribuent un score de pertinence aux résultats de la requête. Les résultats sont triés par ordre décroissant de pertinence et présentés à

1. Comme il a été introduit dans le chapitre 2, l'unité de base qui forme la collection de documents peut varier suivant l'application. Dans le cas de recherche d'information sur le web, l'unité est la page web

l'utilisateur.

5.1.1/ RECHERCHE D'INFORMATION SUR LE WEB

La recherche d'information sur le web est un cas particulier de la recherche d'information. Bien que les grands principes de la recherche d'information soient applicables au web, certaines problématiques sont propres à la recherche d'information sur le web, parmi lesquelles les plus importantes sont la taille du web, la grande hétérogénéité des données et les problèmes liés à l'accès aux ressources.

La taille du web est la caractéristique la plus critique. Les moteurs de recherche tels que Google, Bing ou Yahoo recensent plusieurs milliards de pages, de qualité et de pertinence très hétérogène. Ces moteurs de recherche doivent permettre d'assurer à la fois la pertinence ainsi que l'exhaustivité des résultats à l'utilisateur, avec un temps de réponse acceptable. La maintenance des index est une problématique à part entière. Certains contenus du web évoluent de façon constante, et la qualité d'une information dépend souvent de sa proximité dans le temps. D'autres sources d'information n'évoluent que rarement, mais la qualité des informations qu'elles réfèrent est fixe dans le temps.

Les différents formats et structures variés des pages web sont un frein supplémentaire à une exploitation automatisée. Les crawlers doivent prendre en compte la multiplicité des formats. Cette tâche est cependant facilitée par le développement de frameworks dédiés tels que Apache Tika², pour une majorité de types de contenu structuré. En revanche, l'extraction de contenu pertinent des pages web est particulièrement difficile, étant donné la grande variété des structures des pages (figure 5.1). Dans la littérature, l'extraction de contenu pertinent est nommée "Boilerplate removal", i.e. suppression du contenu indésirable. Plusieurs approches récentes ont amélioré l'état de l'art dans ce domaine, notamment Boilerpipe Kohlschütter et al. (2010), un framework libre d'extraction originaire de google, JusText Pomikálek (2011), ou GoldMiner Endrédy et al. (2013).

Enfin, le respect de l'accès (fréquence des requêtes) aux serveurs web et la taille du web rendent difficile la découverte d'informations mal référencées par les moteurs de recherche. Les robots d'indexation, ou crawlers, sont à la base de la construction de tout système de recherche d'information sur le web. Un robot d'indexation est "*un robot logiciel utilisé par les moteurs de recherche pour parcourir le réseau et les sites web afin d'archiver les pages web au sein des index de référencement*"³. Les crawlers parcourent le web en utilisant les liens hypertexte (hyperliens) entre les pages, afin de construire un index de sous-parties du web. La section suivante décrit les caractéristiques communes à la majorité des crawlers web actuels.

5.1.2/ CARACTÉRISTIQUES D'UN CRAWLER WEB

Le parcours du web (web crawling) est un processus itératif. À chaque étape du processus, un ensemble défini d'items, appelé frontière du crawl, est placé dans une file pour être téléchargés (téléchargés)⁴. Les items sont téléchargés en fonction de leur priorité (i.e. leur pertinence potentielle) dans la frontière. Typiquement, un nombre n d'items

2. <https://tika.apache.org/>

3. www.definitions-marketing.com/definition/robot-d-indexation/

4. Le fetching consiste à télécharger les items de la frontière pour en extraire le contenu



FIGURE 5.1 – Contenu pertinent en vert. Contenu peu pertinent en rouge.

sont fetchés à chaque cycle du crawl, à partir desquels de nouveaux liens sont extraits. Le nouvel ensemble de liens connus est utilisé dans le cycle suivant pour générer une nouvelle frontière. La figure 5.2 décrit le processus cyclique d'un crawler générique.

Manning et al. (2008) divise les caractéristiques communes des crawlers en deux groupes. Le premier est composé des caractéristiques strictement nécessaires communes à tout crawler :

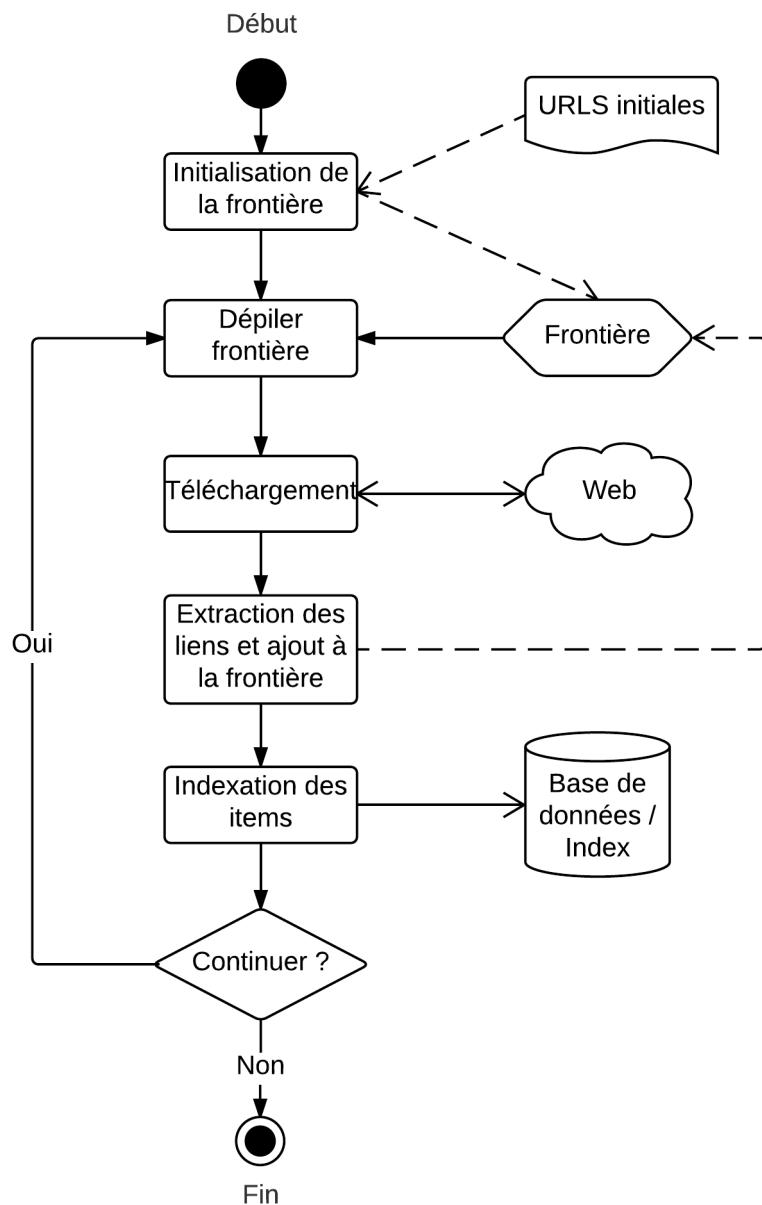


FIGURE 5.2 – Description d'un crawler générique

- Robustesse : la gestion des erreurs d'accès retournées par certains serveurs ne doivent pas impacter le fonctionnement du crawler. Aussi, un crawler web doit éviter les pièges à crawlers, i.e. les "spider traps", ou "tar pits". Certains serveurs web génèrent, intentionnellement ou non, des pièges à crawler, i.e. des enchaînement de pages web dont le nombre de variations d'URLs est virtuellement infini, par exemple en modifiant les paramètres dans l'URL.
- Politesse / éthique : les serveurs web ont des règles explicites et implicites qui régulent leur utilisation, en particulier le nombre d'accès par seconde autorisé pour une même adresse IP. Ces règles permettent d'assurer le bon fonctionnement du serveur web, qui peut être détérioré en cas de crawling automatisé abusif. Dans le cas où le nombre de requêtes au serveur est trop élevé, l'activité du crawler

constitue effectivement une attaque par déni de service. Le respect du "protocole" robot.txt permet d'assurer le bon fonctionnement du serveur.

Le second groupe se compose des caractéristiques additionnelles que nous retrouvons dans un majorité d'implémentations de crawler :

- Parallélisation : Le "crawling" est une tâche coûteuse en temps et en ressources, qui est généralement parallélisé. La parallélisation permet d'augmenter les performances du crawler en temps de recherche, tout en respectant les mesures de politesse vis-à-vis des serveurs. Un crawler doit alors pouvoir s'exécuter de façon distribuée, en répartissant la charge de travail à différentes machines. Cette caractéristique est notamment indispensable pour palier les limitations liées à la politesse citées ci-dessus. Des crawlers parallèles sont communément utilisés dans la littérature, pour tout type de stratégie de parcours. Plusieurs instances d'un même crawler, ou plusieurs processus légers (threads) effectuent le crawl parallèlement. Les problèmes de concurrence entre processus sont généralement évités en maintenant une base commune contenant la frontière et les items déjà fetcheds souvent appelée "crawl database", accompagnée d'une base de données des liens déjà parcourus (link database).
- Passage à l'échelle : un crawler devrait permettre le téléchargement d'un nombre important de pages d'une variété importante, tolérer les erreurs de téléchargement des pages ou de format des items. Il doit également résister aux éventuelles boucles entre les différentes pages, par exemple les pages générées de façon dynamique.
- Performance et efficience : un crawler devrait faire un usage efficient des ressources système mises à sa disposition (processeur, stockage et bande passante).
- Qualité : l'utilité d'un document web dépend grandement du contexte. Un crawler doit être capable de sélectionner des documents pertinents pour l'utilisateur. D'une façon générale, la qualité des documents web est très hétérogène, et difficile à identifier.
- Récence (Freshness) : beaucoup d'applications qui exploitent des données du web considèrent que la "durée de vie" d'un document web est limitée, i.e. les documents web deviennent obsolètes après un temps dépendant du contexte. Aussi, le contenu d'une URL varie à intervalle régulier. Un crawler doit prendre en compte cette dimension éphémère des documents web.
- Modularité : la modularité du crawler est importante pour l'adapter à des cas d'utilisation précis, modifier la stratégie de parcours, et intégrer des composants externes.

Quelque soit l'approche considérée dans sa conception, un crawler suit systématiquement deux étapes pour atteindre un objectif de recherche d'information : l'indexation des items, qui permet par la suite la recherche ou la recommandation. Dans le cas des crawlers génériques dont l'objectif est de recenser un maximum d'informations du web, l'indexation est effectuée indépendamment du besoin final de l'utilisateur, et doit permettre de retrouver une large variété de documents. C'est le cas des moteurs de recherche commerciaux tels que Google ou Bing, qui sont particulièrement appropriés pour une utilisation basique. En revanche, des outils alternatifs sont souvent requis pour répondre à des besoins spécifiques. Les robots d'indexations peuvent alors être paramétrés pour effectuer des tâches spécifiques. Par opposition aux crawlers génériques, les crawlers ciblés (focused crawler) ont pour objectif de restreindre la portée⁵ du crawler à des items

5. Focus, ou scope

qui correspondent à un besoin défini pour l'utilisateur. La section suivante présente les approches existantes de crawling ciblé.

5.2/ CRAWLERS CIBLÉS

L'objectif des crawlers ciblés est de maximiser le nombre d'items pertinents en fonction d'un besoin de l'utilisateur, et de minimiser le nombre d'items non pertinents téléchargés. La restriction de la portée du crawler peut être thématique, par exemple pour restreindre la recherche à l'économie, la politique, ou le sport. Il s'agit alors de crawlers thématiques (topical-crawlers). Il peut également s'agir de rechercher des pages ayant un référencement faible ou fort, ou des données/métadonnées spécifiques comme des données RDF.

La restriction de la portée du crawler répond à la fois à une problématique technique (1) et une problématique utilisateur (2) :

- (1) comme défini dans la section précédente, les caractéristiques indispensables de politesse et de robustesse d'un crawler contraignent une utilisation "gloutonne" abusive. Cette problématique est d'autant plus importante dès lors qu'une source d'information est unique ou difficilement remplaçable, donc de valeur élevée. La perte d'accès à certaines ressources réduit l'exhaustivité des résultats ultérieurs.
- (2) le rapport entre la quantité et la qualité des items retournés par le crawler est l'aspect le plus important pour l'utilisateur. La rapidité de retour des résultats est également un facteur déterminant pour ce dernier. Un crawler doit donc être efficient lors de la recherche de documents. Pour répondre aux exigences de l'utilisateur, parcourir et indexer le web de façon exhaustive n'est pas nécessaire.

Pour restreindre les décisions du crawler à un thème, les crawlers ciblés utilisent différentes stratégies pour assigner une priorité à chaque lien de la frontière de crawl. Les crawlers ciblés maximisent ainsi la proportion de pages pertinentes et limitent la quantité totale de données fetchées Chang et al. (2006). Le contenu des pages déjà fetchées et les liens entre ces pages peuvent être utilisées pour déterminer la priorité des pages non fetchées. Les crawlers ciblés se basent sur les conjectures lien-contenu (link-content conjecture) et lien-cluster (link-cluster conjecture) Liu (2007).

La conjecture lien-contenu tire partie des métriques de similarité issues de la recherche d'information (par exemple le modèle vectoriel), pour représenter et comparer différentes pages. Si une page référence plusieurs pages pertinentes dont le contenu est similaire, alors il s'agit d'un hub⁶ pertinent, et les liens sortant de cette page doivent être fetchés en priorité. Cette stratégie également nommée "sibling locality" Aggarwal et al. (2001) a été appliquée pour des crawlers ciblés, et aussi pour des crawlers thématiques notamment pour la veille stratégique et concurrentielle Pant et al. (2004). L'inverse est également vrai. Plusieurs liens pertinents pointant vers une même source suggèrent qu'il s'agit d'une source pertinente, dont les liens entrants et sortants peuvent être priorisés.

La conjecture lien-cluster suppose que la distance des pages entre elles au sein du graphe web peut être exploitée. Chaque lien est un arc orienté du graphe, entrant ou sor-

6. Un hub est une page web qui référence un grand nombre de pages d'autres domaines. L'inverse d'un hub est une "autorité" (authority), i.e. une page fortement référencée par d'autres pages. Les notions de "hub" et "d'autorités" ont été introduites par Kleinberg et al. (1999) pour l'algorithme HITS, à l'origine d'autres algorithmes d'indexation du web tel que PageRank

tant. Les liens sont une source d'information importante pour déterminer la thématique d'une page. La distance en nombre de liens entre deux pages peut donner une indication de rapprochement sémantique Menczer (2004) entre les deux pages. Ce rapprochement peut parfois être plus pertinent qu'un rapprochement lexical uniquement basé sur le contenu Chakrabarti et al. (1999), notamment car les biais d'extraction de contenu et de qualification de l'information sont évités.

La figure 5.3 illustre comment le graphe web peut être exploité via ces conjectures.

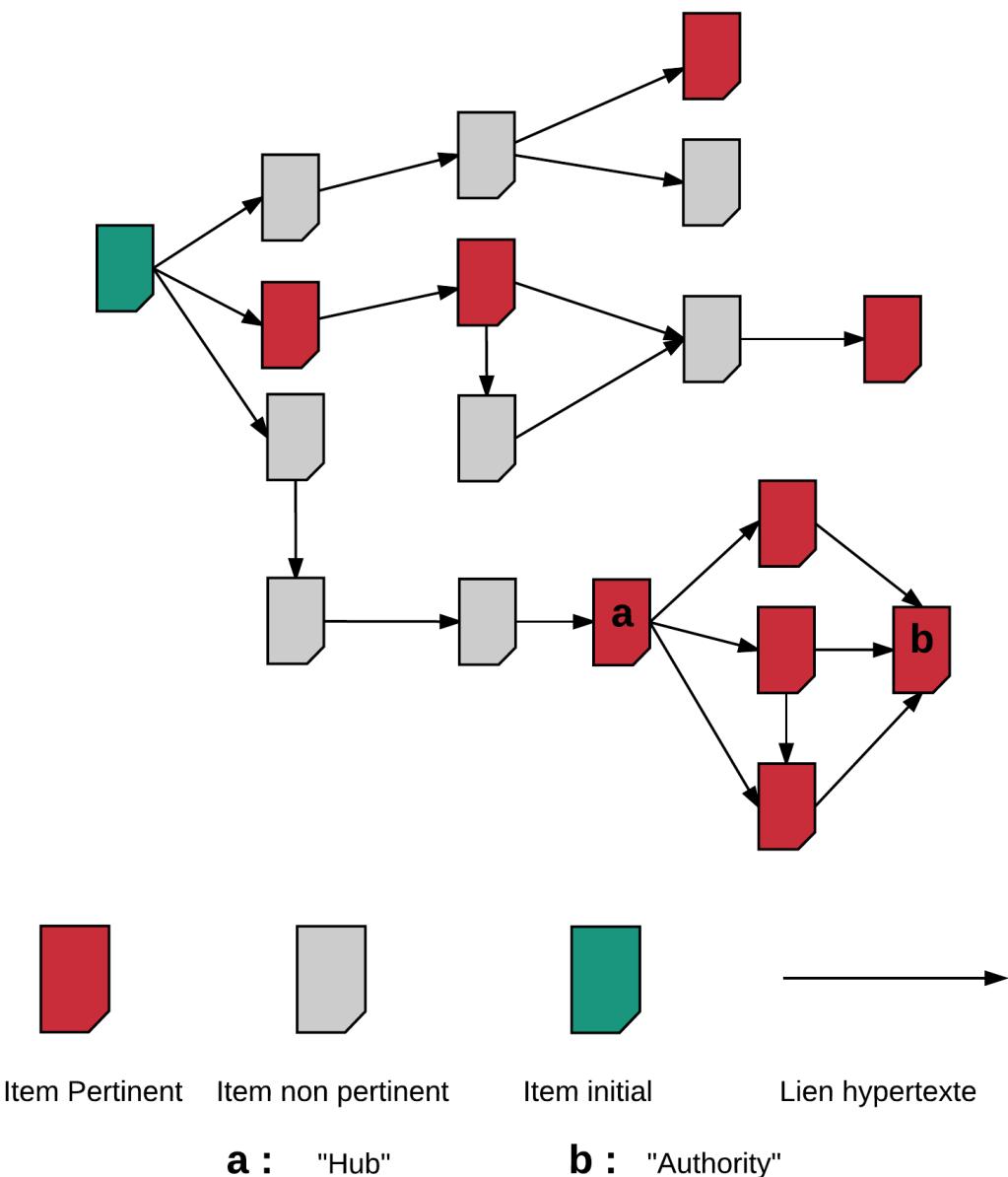


FIGURE 5.3 – Exemple de distribution de pertinence des pages

Liu (2007) utilise la similarité cosinus pour évaluer la distance lexicale entre deux pages aléatoires via un crawler breadth-first⁷. Les auteurs observent un rapprochement lexical

7. Une approche breadth-first est une implémentation naïve d'un crawler qui consiste à parcourir de façon

significatif jusqu'à une distance de 3 liens, et montrent que la distance en nombre de liens et la distance lexicale sont corrélées. Liu (2007) montre également que les conjectures lien-contenu et lien-cluster sont plus ou moins efficaces en fonction du thème (sport, news, politique...) pour lequel un crawler thématique est utilisé. Les crawlers thématiques semblent être globalement plus performants pour certains domaines tels que les nouvelles de presse. Aussi, chacune des deux conjectures est plus ou moins importante en fonction du thème. Cet aspect est logique dans le sens où certains types de contenus profitent plus du référencement mutuel avec d'autres ressources. Les sites de commerce par exemple n'ont pas grand intérêt à référencer directement leurs concurrents, ce qui rend la conjecture lien-cluster inefficace.

Dans les sections suivantes, les principales approches de crawling ciblé de la littérature sont revues, puis deux types d'approche spécifiques sont abordés. Les approches basées sur les ontologies, et les approches basées sur l'apprentissage automatique. Les limitations de ces approches sont ensuite analysées, pour expliquer le positionnement de la nouvelle architecture proposée dans le chapitre suivant.

5.2.1/ STRATÉGIES INITIALES DE PARCOURS

Les stratégies initiales de parcours sur lesquelles se basent les crawlers ciblés ont été établies à partir des conjectures lien-cluster et lien-contenu. Les approches plus récentes étendent ces stratégies initiales.

La majorité des crawlers ciblés et thématiques utilisent un ensemble de pages initiales nommées seed pages pour commencer à rechercher des items similaires. Les pages initiales sont généralement fournies par l'utilisateur, ou sélectionnées parmi les réponses d'une requête à un moteur de recherche généraliste McCown et al. (2007), par exemple en utilisant le thème de la page initiale comme requête Kraft et al. (2003) Pant et al. (2004) Bao et al. (2008).

Des pages initiales de bonne qualité peuvent être des pages correspondant à un thème recherché, ou des pages qui mènent à une page pertinente directement ou indirectement (distance en nombre de liens faible). La frontière du crawl, i.e. l'ensemble des pages candidates à fetcher, est alors composée des liens sortants extraits des pages initiales.

A partir des pages initiales, un crawler ciblé emploie ensuite une méthode pour assigner une priorité à chaque lien, en sélectionner de façon itérative des liens potentiellement pertinents.

L'approche Fish-Search de De Bra et al. (1994), assigne des valeurs de priorité binaires (1 = pertinent, 0 = non pertinent) aux pages candidates, en fonction de la similarité lexicale entre les items et une requête (mots clés). Toutes les pages candidates ont la même priorité dans la file d'attente du crawler.

Les stratégies de type Best-First calculent la similarité entre la page initiale et les nouvelles pages par application du modèle vectoriel (VSM Salton et al. (1975)). Les items sont représentés par un vecteur de termes en fonction de leur contenu, et la distance entre deux items est calculée par une mesure de distance vectorielle telle que la similitude cosinus.

Les approches initiales de type Best-First ne considèrent que la fréquence des termes

dans chaque document, pour calculer leur pertinence, et ne tirent pas partie de la fréquence inverse pour prendre en compte la pertinence des termes dans le corpus. Prendre en compte la fréquence inverse demande en effet de recalculer les vecteurs de termes à chaque cycle de crawl. Aussi, les fréquences ne sont pas représentatives de la pertinence des termes au début du crawl, car la taille du corpus est trop petite.

Shark-Search Hersovici et al. (1998) est une variante de la stratégie Best-First ou la fonction qui détermine la priorité des pages est plus complexe. L'approche est basée sur le modèle VSM (Vector Space Model). La priorité assignée prend en compte le contenu de la page courante, le texte d'ancrage, le texte entourant chaque lien, ainsi que la priorité des pages parentes (pages pointant vers la page courante dont les liens sont extraits).

La stratégie Best-N-First de Menczer et al. (2004) est une généralisation de l'approche Best-First, où les N pages candidates dont les valeur de priorité sont les plus élevées sont fetchées à chaque cycle de crawl. De la même façon, Aggarwal et al. (2001) propose une approche dite de crawling intelligent, où la calcul de la priorité d'un lien dépend du contenu de la page parente, de l'URL correspondant au lien hypertexte, et des pages "sœurs" (pages situées à la même distance en nombre de liens). Le crawler est alors capable de recherche des informations sans entraînement préalable.

D'après Menczer et al. (2004), les approches basées sur la stratégie Best-First/Best-N-First sont à la fois simples, et plus performantes que les approches Shark-Search, Fish-Search, ou les approches naïves de type Breadth-search. Cette stratégie est donc souvent utilisée en comparaison aux nouvelles approches.

Tout comme dans un système de recherche d'information standard (hors web), ces stratégies initiales sont impactées par le biais de l'analyse de contenu. Elles font une estimation de la similarité des items via des modèles standard de la recherche d'information (modèle vectoriel dans le cas des approches Best-First). Comme dans tout système de recherche d'information, la similarité est limitée car elle est seulement lexicale. Deux items sont rapprochés uniquement s'il partagent des termes communs, sans prise en compte de leur sémantique (synonymie par exemple).

La section suivante décrit un ensemble d'approches qui étendent les approches de crawling ciblé, qui tentent de pallier ces problèmes via l'intégration de méthodes issues du web sémantique.

5.2.2/ CRAWLERS SÉMANTIQUES

Les crawlers sémantiques se basent sur des méthodes de recherche d'information basés sur des modèles sémantiques. La plupart de ces modèles reposent sur une base de connaissance (taxonomie, ontologie) pour représenter et comparer plus précisément les items. Des modèles sémantiques dérivés du modèle vectoriel tels que Varelas et al. (2005) (Semantic Similarity Retrieval Model) et Hliaoutakis et al. (2006) sont plus performants. En outre, ces méthodes sont adaptées aux crawlers thématiques Batsakis et al. (2009).

Chakrabarti et al. (1999) propose une méthode de crawling basée sur un modèle de classification. La construction du modèle de classification est basée sur une taxonomie (taxonomie Yahoo⁸). Le modèle permet de classer de nouveaux items issus du crawler

8. yahoo.com

selon les concepts de la taxonomie. Le système est composé de 3 modules : un crawler, un classifieur et un distillateur. Le classifieur évalue la pertinence des pages en fonction de leur classification dans la taxonomie. En fonction de son score de pertinence, les liens sortants d'un item sont priorisés dans la file d'attente. Périodiquement, le distillateur identifie les hubs en utilisant l'approche de Kleinberg (1999). Les hubs pertinents sont alors revisités. Les résultats de Chakrabarti et al. (1999) montrent que la pertinence moyenne est comprise entre 30% et 50%, et est constante en fonction du temps. Comparativement, un algorithme breadth-first sur le même échantillon produit une similarité moyenne proche de 0.

Ehrig et al. (2003) propose un algorithme de similarité des items basé sur une ontologie. Les concepts de l'ontologie sont extraits des items après une chaîne de pré-traitements. Différentes mesures de similarité sémantiques sont utilisées pour évaluer la pertinence des items en fonction des concepts de l'ontologie, en tirant partie des relations sémantiques entre les concepts. Comparativement à une approche de classification binaire basée sur des mots clés, les auteurs montrent que la taux d'items pertinents (harvest rate) est amélioré.

Une des difficultés principales dans la conception d'un crawler est la gestion du ratio exploration sur exploitation. L'exploration concerne la capacité du crawler à étendre sa recherche d'information de façon très large, sans nécessairement trouver d'information pertinente le plus rapidement possible. L'exploitation concerne la capacité du crawler à chercher des données pertinentes à partir d'autres données déjà considérées comme pertinentes. Prioriser uniquement l'exploration ou l'exploitation ne permet pas au crawler de trouver des informations pertinentes efficacement. Un équilibre doit être trouvé pour permettre au crawler de trouver un maximum d'informations, tout en conservant un taux d'informations pertinentes le plus élevé possible⁹.

Les crawlers sémantiques ont été efficacement utilisés pour pallier les problèmes d'analyse de contenu de la même façon que dans un système de recherche d'information standard (hors web). Ces crawlers sont performants pour l'exploitation, mais pas pour l'exploration. En outre, ils ne tirent pas partie de l'expérience acquise au cours du temps pour s'améliorer au cours du temps, et sont incapables d'exploiter le graphe web pour optimiser les temps de recherche.

La section suivante décrit des approches basées sur des méthodes d'apprentissage automatique appliquées aux crawlers ciblés, afin de pallier ces limitations.

5.2.3/ APPROCHES BASÉES SUR L'APPRENTISSAGE AUTOMATIQUE

Pour optimiser les chances de trouver des pages pertinentes, des méthodes d'apprentissage automatique peuvent être appliquées au crawler pour orienter sa recherche. Un ensemble d'items pertinents est alors utilisé comme jeu de données d'entraînement. Pour un crawler thématique, il s'agit d'items correspondant au thème désiré.

L'apprentissage peut également tirer partie du graphe web, via l'analyse des chemins, suite de liens menant aux items pertinents. Les premières approches de crawlers basées sur l'apprentissage automatique utilisent des méthodes de classification pour identifier les pages pertinentes, telles que les classifieurs bayésiens Chakrabarti et al. (1999). D'autres

9. Des métriques spécifiques au domaine de la recherche d'information sur le web permettent d'évaluer le taux d'information pertinentes d'un crawler, et seront introduites à la fin de cette partie

approches sont basées sur des arbres de décisions Li et al. (2005), la logique de premier ordre Xu et al. (2007), les réseaux de neurones artificiels, ou les machines à vecteurs de support Pant et al. (2005). D'après Pant et al. (2005), les machines à vecteurs de support performent mieux que les autres approches.

Bergmark et al. (2002) décrit une optimisation de l'approche best-first, nommée ‘tunneling’, qui a notamment pour objectif de prendre en compte la difficulté de trouver des items pertinents éloignés entre eux (nombre de liens non pertinents important entre deux ressources pertinentes). L'objectif n'est pas toujours de minimiser le nombre d'items fetchés. Le crawler explore plus longuement qu'un crawler focalisé standard, même lorsque des liens non pertinents sont trouvés. Leurs résultats montrent que l'approche permet de retrouver des items pertinents à une distance éloignée les uns des autres. Les auteurs montrent qu'utiliser le chemin menant à des pages pertinentes est un point clé de la conception d'un crawler ciblé.

Dans Pant et al. (2006), les machines à vecteurs de supports (Support Vector Machine) sont utilisées pour définir la stratégie du crawler, en fonction du contenu des pages et du graphe web. Les auteurs montrent qu'utiliser à la fois le contenu de pages et les liens est plus efficace que le contenu ou les liens seuls.

Diligenti et al. (2000) introduit la notion de graphe contextuel (context-graph), où les liens entrants des pages pertinentes sont suivis en sens inverse pour retrouver les pages menant aux pages pertinentes. Un classifieur est ensuite construit pour chaque ensemble des pages à différentes distances des pages pertinentes jusqu'à une valeur de seuil. A chaque distance en nombre de liens correspond une couche du graphe. Les classificateurs attribuent un score aux nouvelles pages, qui sont associées à la distance correspondant au score maximum parmi l'ensemble des couches du graphe. A la fin du cycle de crawl, la frontière est mise à jour avec les liens extraits de ces pages, puis la priorité des nouveaux liens est déterminée en fonction du score de la page parente. La priorité est inversement proportionnelle à la distance de la page parente.

Dans Chakrabarti et al. (2002), deux modèles de classification sont construits pour classer les pages. Un premier classifieur détermine de façon binaire la pertinence d'une page en fonction de la taxonomie ODP¹⁰, puis un second classifieur évalue la probabilité que la page mène à d'autres pages pertinentes.

Chen (2007) propose une approche basée sur un algorithme génétique, qui tente d'apprendre à partir des séquences de liens menant vers des items pertinents. L'approche est hybride afin d'améliorer le ratio exploration / exploitation : le crawler est guidé alternativement de façon naïve (breadth first) et par l'algorithme génétique.

Lu et al. (2016) a récemment proposé une approche basée sur un classifieur bayésien. Le classifieur est entraîné à partir du contenu des items : le modèle vectoriel est utilisé pour représenter chaque item, et les termes sont pondérés par leur valeur TFIDF. Un algorithme d'assignation de priorité des liens ("LPE", pour Link priority evaluation) est décrit : la priorité de chaque lien extrait est calculée en fonction du contexte du lien. Les pages web sont d'abord découpés en block, puis la similarité du block englobant chaque lien avec le thème recherché permet d'assigner une priorité au lien. Les auteurs observent un taux de récolte amélioré en comparaison à l'approche Best-N-First.

L'apprentissage automatique et les ontologies corrigent certaines limitations des crawlers ciblés. Les sections précédentes ont souligné la complémentarité de ces deux types d'op-

10. <https://www.dmoz.org/>

timisations. La section suivante définit les limitations de ces deux types d'approches, et étudie des approches hybrides, dont l'objectif est de combiner ces deux approches pour optimiser la stratégie de recherche d'un crawler ciblé.

5.2.4/ APPROCHES HYBRIDES

Les approches basées sur les ontologies sont dépendantes de l'ontologie utilisée. Cette dépendance induit deux limitations Dong et al. (2014) :

- Les ontologies conçues par des experts du domaine représentent la conceptualisation d'un domaine, mais peuvent ne pas correspondre aux connaissances du domaine du monde réel.
- Les ontologies de domaine évoluent uniquement par mises à jour manuelles, alors que les données évoluent de façon constante et dynamique.

Ces limitations peuvent rendre les ontologies inadaptées pour représenter des données du monde réel, notamment du web Dong et al. (2014). Pour résoudre ces limitations, des approches de la littérature s'intéressent à l'intégration de méthodes d'apprentissage automatique avec des méthodes de crawling sémantique ciblé, pour tirer partie des avantages des deux approches.

Dans Chakrabarti et al. (2002) la pertinence des pages et la priorité de la frontière sont évaluées séparément par deux modèles. Le modèle de pertinence est un modèle de classification binaire basé sur une taxonomie. Le modèle d'évaluation de la priorité est construit à partir d'items du web. Pour évaluer les liens potentiellement pertinents, un ensemble de liens entrant vers des items pertinents sont utilisés. Le contenu de la page parente et le contexte (termes) entourant les liens permettent d'entraîner le modèle. Le modèle réduit de façon significative (de 30% à 90%) le nombre de faux positifs, i.e. le nombre de liens menant vers des pages non pertinentes.

Zheng et al. (2008) décrit une méthode supervisée d'apprentissage d'une ontologie basé sur un réseau de neurones artificiel. Le réseau de neurones est construit automatiquement à partir des données, et permet de comparer les items avec les concepts d'une ontologie statique.

Su et al. (2005) utilise une ontologie de domaine pour décrire les items issus d'un crawling thématique. Un ensemble de mots est associé à chaque concept de l'ontologie. Chaque item est décrit par les concepts de l'ontologie qu'il contient, où chaque label est pondéré par sa fréquence dans l'item. Un algorithme d'apprentissage par renforcement non supervisé permet de faire évoluer l'ontologie lors du crawl. Le calcul du score de chaque item dépend du poids de chaque concept dans l'ontologie par rapport au thème recherché dans le crawl. Le poids des concepts de l'ontologie est fixé au départ, et évolue par apprentissage lors du crawl en fonction de la cooccurrence des concepts, ce qui induit des changements dans l'évaluation de la pertinence des nouveaux items.

Carlson et al. (2010) décrit un crawler non ciblé dont l'objectif est l'apprentissage continu de faits du monde réel. Différents modules permettent la reconnaissance des faits à partir du texte en langage naturel. Une ontologie est peuplée avec les nouveaux faits découverts de façon continue (la tâche d'apprentissage du crawler ne s'arrête pas). Deux types de faits sont détectés, et sont représentés sous la forme d'assertions de classes et de d'assertions de rôles dans l'ontologie. Une assertion de classe permet d'attribuer une catégorie à une entité nommée, par exemple une personne réelle ou une ville. Une assertion de rôle représente une relation sémantique entre deux entités. L'approche est

TABLE 5.1 – Positionnement de l'approche SEMXDM

	Zheng et al. (2008)	Su et al. (2005)	Dong et al. (2014)	SEMXDM
Paradigme d'apprentissage	supervisé	non supervisé	semi-supervisé	non supervisé
Classification	non	oui	oui	oui
Apprentissage de termes	non	oui	oui	oui
Apprentissage de Relations	non	oui	oui	oui
Environnement non contrôlé (web)	non	non	oui	oui

basé sur un apprentissage semi-supervisé des faits, à partir d'un modèle pré-étiqueté de petite taille. L'ontologie est enrichie avec des nouveaux faits chaque jour, et prend en compte un retour utilisateur quotidien pour confirmer ou infirmer les faits découverts, améliorant ainsi continuellement la qualité du modèle.

Dong et al. (2014) propose une méthode de crawling sémantique adaptative (SASF crawler). L'approche combine des méthodes d'apprentissage automatique d'une ontologie, et des méthodes crawling sémantique. L'approche sémantique permet de résoudre les problématiques d'ambiguïté et d'hétérogénéité des données du domaine, tandis que l'apprentissage automatique permet d'adapter le comportement du crawler et d'optimiser ses performances dans un environnement web dynamique. Les items pertinents alimentent un moteur de recherche spécifique au domaine.

5.3/ PROPOSITION : PROCESSUS SEMXDM

Notre étude de la littérature a souligné deux limitations principales des crawlers ciblés :

- La majorité des approches se focalisent uniquement sur le contenu des items. Ces approches ne tirent pas partie du graphe web pour optimiser la recherche des items ou leur classification. Cette problématique recoupe une des limitations des approches de classification étudiée dans le chapitre 2 de ce manuscrit, où la majorité des approches ne considérait pas la classification collective, i.e. l'usage des liens entre les items pour la classification.
- Les approches basées sur les ontologies ont des avantages clairement identifiés, mais sont souvent limitées pour une utilisation dans un contexte web, où les données évoluent constamment.

Le chapitre suivant décrit une nouvelle architecture de crawling ciblé, nommée SEMXDM, basée sur l'architecture de classification hiérarchique multi-label présentée dans le chapitre précédent. Le tableau 5.1 présente le positionnement de l'architecture SEMXDM vis-à-vis des approches similaires décrites précédemment, selon leurs principaux critères :

L'architecture SEMXDM tente de palier les limitations identifiées dans l'état de l'art, par une adaptation continue du modèle de classification sémantique en fonction de flux de données web, et une approche hybride de qualification des données utilisant à la fois le

contenu des items et le graphe web. Pour intégrer ces éléments au crawler, l'architecture est divisée en modules. Chaque module impacte le comportement du crawler et porte sur un processus distinct, i.e. la classification, l'adaptation du modèle de classification dans le temps (module de maintenance), et l'exploitation du graphe web (module de priorité).

6

SEMXDM : WEB CRAWLER SÉMANTIQUE ADAPTATIF

Après avoir présenté l'état de l'art des méthodes de recherche d'information sur le web, et plus spécifiquement des crawlers ciblés, nous avons montré les limites de ceux-ci. Bien que les systèmes existants tirent partie de l'apprentissage automatique et des ontologies pour classer automatiquement l'information, peu d'approches considèrent l'adaptation du modèle de classification en temps réel, et la combinaison de l'exploitation du graphe et du contenu web. En outre, il semble qu'aucune approche ne soit basée sur un apprentissage non supervisé du modèle prédictif, tel que le processus décrit dans les chapitres 3 et 4. D'un point de vue métier, la difficulté d'adaptation des approches existantes à des connaissances et données spécifiques est un frein supplémentaire à la mise en place de ces processus.

Ce chapitre décrit une architecture de recherche d'information sur le web, qui tente de pallier ces limites. Cette architecture s'intéresse spécifiquement à la tâche de croisement d'information sur le web, qui est déterminante dans le domaine de la veille stratégique, dans lequel ces travaux s'inscrivent. L'architecture, nommée SEMXDM¹, est basée sur un crawler ciblé. Pour répondre à l'objectif de croisement d'information, cette architecture repose sur une classification sémantique basée sur le contenu des items, effectuée par l'architecture SHMC présentée dans les chapitres 3 et 4. Tout type de document web contenant du texte structuré ou non structuré est un item (pages web, flux rss, pdfs...), identifié par une URL.

Une première section décrit l'architecture globale de notre approche. La seconde section décrit son implémentation. Dans la troisième section une évaluation de l'approche s'intéresse à la faisabilité de la proposition, et la compare à une approche de l'état de l'art introduite dans le chapitre précédent (approche de type Best-N-First). Les résultats préliminaires de cette évaluation indiquent une performance supérieure à l'approche de l'état de l'art selon des métriques d'évaluation qualitative des crawlers. Les bénéfices et les limitations de notre approche sont analysées, notamment en terme de passage à l'échelle.

1. Semantic Cross-Referencing Data Mining

TABLE 6.1 – Concepts du modèle prédictif

Concepts DL	Description
$Item \sqsubseteq \exists hasTerm.Term$	associe un item à ses attributs (termes)
$Term \sqsubseteq asString.String$	Termes extraits des items
$Label \sqsubseteq Term$	Termes pertinents pour classer les items
$Label \sqsubseteq \forall broader.Label$	Relation de généralisation
$Label \sqsubseteq \forall narrower.Label$	Relation de spécialisation
$broader \equiv narrower^{-}$	les relations Broader et Narrower sont inverse
$Item \sqcap Term \equiv \perp$	Items et Termes sont disjoints
$Label \sqsubseteq \forall hasAlpha.Term$	Termes composant une règle Alpha
$Label \sqsubseteq \forall hasBeta.Term$	Termes composant une règle Beta
$Item \sqsubseteq \exists isClassified.Label$	Représente la classification d'un item

6.1/ ARCHITECTURE SEMXDM

L'architecture SEMXDM architecture comprend deux modules principaux : un crawler ciblé, ou module de recherche, qui permet le parcours du web et la recherche d'items, et un classifieur, ou module de classification, basé sur l'architecture SHMC, qui permet de déterminer la pertinence des items. La classification repose sur un modèle prédictif qui permet de décrire et classer les item. La construction du modèle prédictif est effectuée à l'aide de l'architecture SHMC², présentée dans les chapitres précédents. Le modèle prédictif est décrit par une ontologie d'expressivité \mathcal{ALCI} dont les axiomes en logique de description sont présentés dans le tableau 6.1 (cette ontologie est identique à l'ontologie de l'architecture SHMC). Le modèle prédictif effectue une classification hiérarchique multi-étiquette des items, en fonction des attributs (termes) extraits des items. Le modèle est composé d'une hiérarchie de labels, i.e. les concepts pertinents décrivant les items, et de règles de classification qui permettent d'associer les labels aux items.

Les items du web indexés par le crawler sont intégrés dans l'ontologie en tant qu'individus (ABox), et la classification est effectuée par un raisonneur qui applique de façon exhaustive les règles de classifications du modèle prédictif aux items.

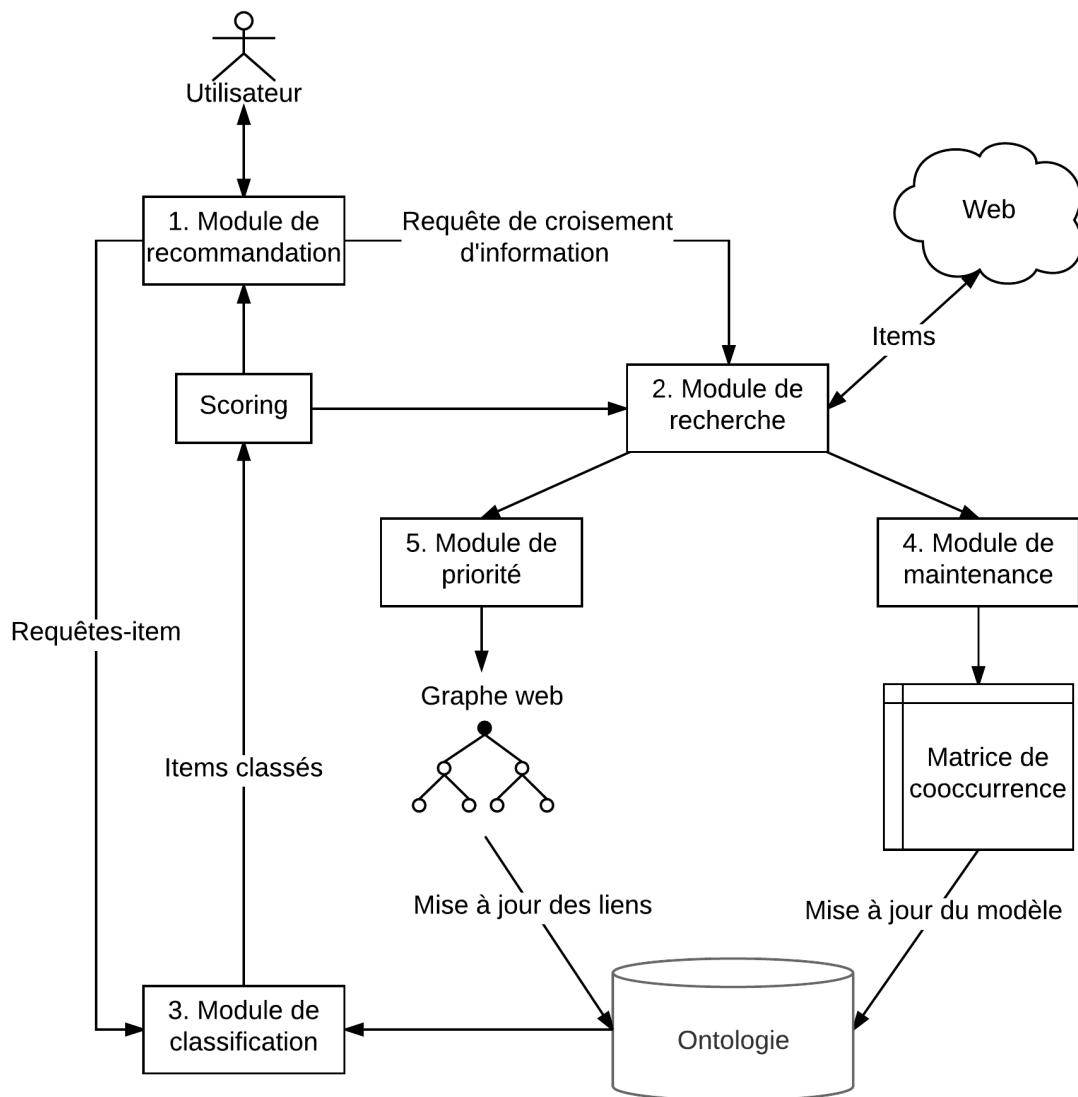
La figure 6.1 présente l'architecture globale et le rôle de chaque module. Les sous-sections suivantes décrivent chaque module séparément.

La combinaison des modules de recherche (2) et de classification (3) est un crawler ciblé basé sur le modèle prédictif issu de l'architecture SHMC, qui effectue une classification sémantique des items. La matrice de cooccurrence et l'ontologie, décrite dans le tableau 6.1 sont des éléments déjà abordés dans le processus SHMC. L'architecture SemXDM comprend trois modules complémentaires, qui définissent la stratégie de recherche du crawler ciblé, et permettent d'adapter cette stratégie au cours du temps.

Le module de recommandation (1) répond aux requêtes de l'utilisateur, notamment les requêtes de croisement d'information sur le web, à partir desquelles une tâche de recherche est lancée. Un module de priorité (5) calcule la pertinence des items du web, en

2. Voir définition 1.2

FIGURE 6.1 – Architecture SemXDM



fonction de leur similarité à la requête de l'utilisateur. Le calcul de pertinence exploite le résultat de la classification des items d'une part, et d'autre part le graphe de liens issu du module de recherche. Enfin, le module de maintenance (4) permet l'adaptation du modèle prédictif, à partir des nouvelles données récoltées. L'objectif de ce module est de répondre à la problématique de classification d'items issus du web, dont les propriétés évoluent au cours du temps.

6.1.1/ MODULE DE RECOMMANDATION

Le module de recommandation est le point d'interaction entre l'utilisateur (expert du domaine) et le système de recherche. Le module répond aux requêtes des utilisateurs, et renvoie les items les plus pertinents en fonction de la requête. Un système de recherche

d'information peut considérer différents types de requête utilisateur³. Tous les types de requête ne sont pas traitées par l'architecture SEMXDM. Nous considérons essentiellement deux types de requêtes, pour effectuer la tâche de croisement d'information : les requêtes de type "mot-clé" (Item queries), et les requêtes "item" ("full-document queries").

La figure 6.2 décrit les deux types de requête considérées :

- Requête par termes (requête mot clé) : ce type de requête prend en entrée un ou plusieurs mots clés définis par l'utilisateur, et recherche dans l'index les items correspondant à la requête. Les mots clés sont normalisés par une chaîne de traitement syntaxique, afin de faire correspondre les mots clés aux termes de l'index sur lesquels les items sont indexés. La recherche est effectuée hors-ligne, i.e. les items retournés sont déjà intégrés dans l'index et la base de connaissances. La requête est alors composée d'un ensemble de termes $\omega_{term_i} = (term_1, term_2, \dots, term_n)$. La réponse à la requête est un ensemble d'items $\omega_{item_j} = (item_1, item_2, \dots, item_n)$ où $\forall j, \exists item_j hasTerm(term_i)$, et $term_i \in \omega_{term_i}$. Un classement des items de ω_{item_j} retournés est effectué par ordre décroissant selon la similarité cosinus entre chaque vecteur de termes et la requête initiale.
- Requête-item : ce type de requête prend en entrée un item déjà présent dans le système de recherche d'information, et classé avec des labels du modèle prédictif. La requête prend alors la forme d'un vecteur de termes correspondant à l'item initial. La requête est envoyée au module de recherche (crawler), qui a pour objectif de chercher des items similaires sur le web. Le calcul de similarité entre les nouveaux items crawlés est similaire au calcul hors ligne. La similarité cosinus entre les vecteurs de termes de chaque item est définie comme score de pertinence. Le calcul du score potentiel des liens sortants de chaque item est décrit dans la section 6.1.5.

6.1.2/ MODULE DE RECHERCHE

Ce module consiste en un crawler web, qui permet une recherche performante d'un grand volume d'items du web. Le module de recherche est le module central de l'architecture, qui connecte l'ensemble des modules. Une recherche d'items est déclenchée lorsqu'une requête utilisateur de croisement d'information est reçue depuis le module de recommandation.

La recherche est divisée en cycles. A chaque cycle, un nombre fixe d'items de la frontière du crawl est fetché. La recherche s'arrête après un nombre pré-déterminé de cycles, ou après un nombre total d'items fetchés. La figure 6.4 décrit le fonctionnement interne du module de recherche.

Les items fetchés sont indexés en fonction de leur contenu (termes), suivant la méthode décrite dans le chapitre 3. Un index inversé des termes est construit à partir des items. Dans le cas du crawler ciblé, l'index est mis à jour avec les nouveaux items crawlés. L'indexation des items est effectuée après chaque cycle du crawl. L'index permet d'accéder aux fréquences de chaque terme, nécessaires notamment pour l'évolution du modèle prédictif effectuée par le module de maintenance.

L'index inversé est composé d'un ensemble de vecteurs d'items, i.e. un vecteur pour chaque terme $term_i \in Term$, de la forme : $v_{term} < item_1, item_2, \dots, item_n >$. Les termes sont extraits de chaque item selon la méthode décrite en section 3.2 Pour chaque item, un

3. Voir section 5.1

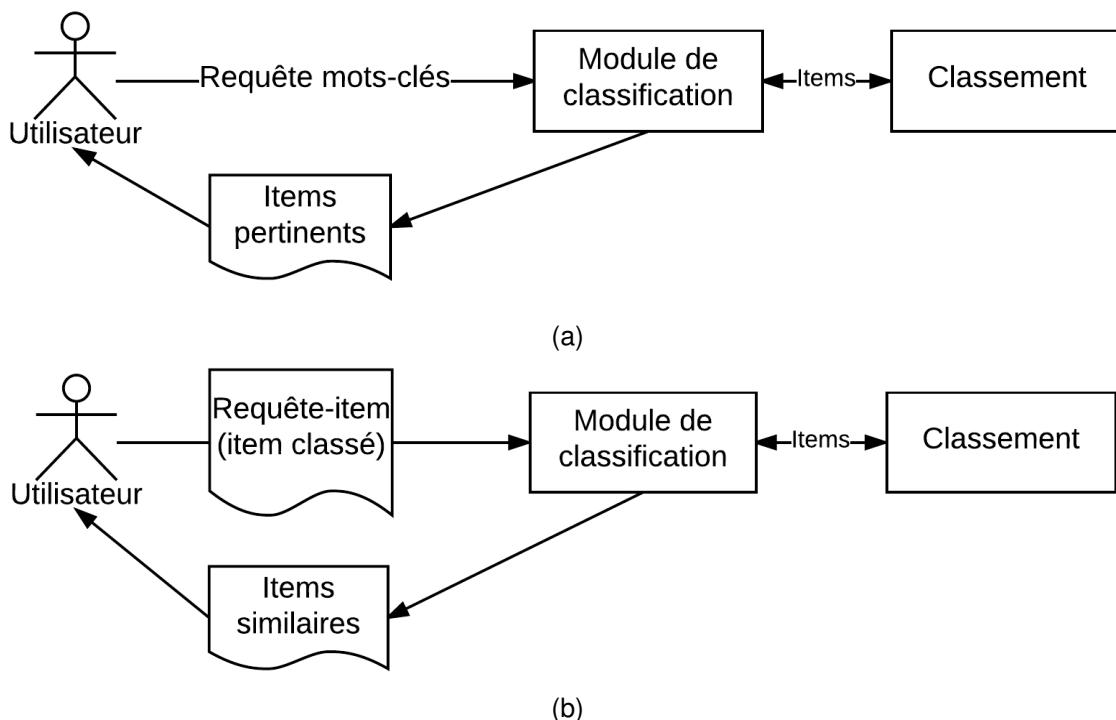


FIGURE 6.2 – Module de recommandation, interaction avec l’utilisateur

processus d’extraction des termes discrimine les termes non pertinents (bruits). Le résultat de l’extraction est un vecteur de termes $v_{item_i} < term_1, term_2, \dots, term_n >$. L’extraction des termes comprend une chaîne de traitement du langage naturel composée de sous-traitements décrits en section 3.2.

6.1.3/ MODULE DE CLASSIFICATION

L’objectif du module de classification est de déterminer la pertinence des items crawlés, en s’appuyant sur la méthode de classification hiérarchique multi-étiquette développée dans les chapitres précédents pour définir la pertinence des items selon les termes les plus pertinents du modèle de classification (labels).

Dans cette approche, une matrice de cooccurrence de termes est créée à partir d’un corpus comprenant un grand nombre d’items. Les termes les plus pertinents sont d’abord identifiés en temps que *Label* dans l’ontologie. Puis, à partir de la matrice de cooccurrence, des règles de classification pour chaque label sont générées, ainsi que des relations hiérarchiques entre les labels.

Le module de classification reprend le principe de la dernière étape du processus SHMC, la réalisation, qui effectue la classification des items dans l’ontologie par un raisonneur (figure 6.3). Le résultat de la classification est un ensemble de relations de classifications entre un item et des labels, i.e. *isClassified(item, label)*.

Dans l’approche SEMXDM, un modèle de classification pré-établi est utilisé à l’initialisation du processus. L’architecture SHMC permet de générer ce modèle et de l’intégrer à une base de connaissances, tel que décrit dans la partie 1 de ce manuscrit.

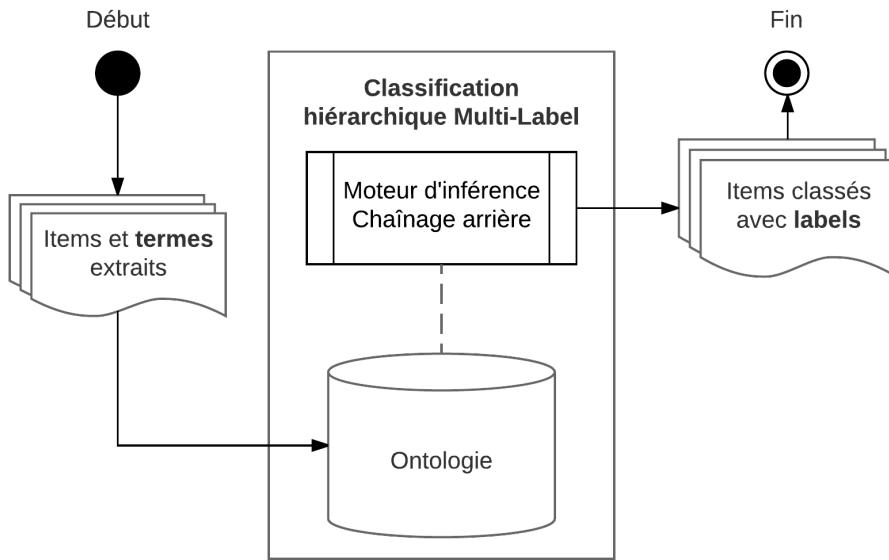


FIGURE 6.3 – Module de classification

La méthode de réalisation effectue la classification par un moteur d'inférence selon le modèle, pour classer les nouveaux items fetchés. Elle est composée de deux sous-étapes : peuplement et classification.

Lors du peuplement, chaque item est d'abord décrit par l'ensemble des termes extraits de celui-ci lors de son indexation. L'ontologie est peuplée avec les nouveaux items en tant qu'individus (Abox), selon les axiomes du modèle (tableau 6.1). Chaque item est décrit par un ensemble de termes pertinents $\omega_{\gamma}^{(item_i)}$ tel que :

$$\omega_{\gamma}^{(item_i)} = \{term_j | \forall term_j \in Term \wedge \gamma < tfidf_{(item_i, term_j, C)}\} \quad (6.1)$$

où γ est le **seuil de pertinence**, $\gamma < tfidf_{(item_i, term_j, C)}$, $term_j \in Term$, $item_i \in Item$. La valeur de $tfidf$ est la fréquence inverse des termes dans l'ensemble des items fetchés, et est calculée de la même façon que lors de la vectorisation (section 3.2).

Un moteur d'inférence est ensuite utilisé pour effectuer la classification hiérarchique multi-étiquette des items, en se basant sur le modèle prédictif i.e. la hiérarchie de labels et les règles de classification. L'étape de classification attribue à chaque item les labels appropriées en fonction du modèle prédictif décrit par la hiérarchie et les règles de classification. Des implémentations de raisonneurs sémantiques basés sur des algorithmes d'inférence (tableau, résolution) tels que Pellet Sirin et al. (2007), FaCT++ Tsarkov et al. (2006), ou Hermit Shearer et al. (2009), permettent d'effectuer une inférence complète des axiomes implicites pour une ontologies ayant une expressivité élevée, telle que OWL2 $SROIQ(\mathcal{D})$. Les travaux de Werner (2015) ont montré que dans un contexte réel ou le nombre de règles est élevé, ces moteurs ne permettent pas de passer l'échelle et de gérer un volume important de données. L'approche de raisonnement basée sur des règles Horn présentée dans la partie précédente est également employée dans cette architecture pour palier cette limitation.

Pour inférer les labels à partir d'un ensemble de termes, le processus de création de règles (Résolution) de l'approche SHMC est utilisé. Les règles sont générées à partir

d'un jeu de données d'entraînement antérieurement au crawl. Ces règles définissent les conditions minimales pour qu'un individu $item_i \in Item$ de l'ontologie soit classé avec un individu $label_j \in Label$. Deux types de règles sont intégrées dans l'ontologie :

Les règles de type Alpha sont basées sur l'appariement entre un terme unique et un label, tel que l'apparition du terme suffise pour déduire une classification, par exemple :

$$\begin{aligned} & Item(?it), Term(?t_1), Label(?t_1), hasTerm(?it, ?t_1) \\ & \rightarrow isClassified(?it, ?t_1) \end{aligned} \quad (6.2)$$

Les règles de type Beta sont basées sur une combinaison de plusieurs termes pertinents pour effectuer la classification, par exemple :

$$\begin{aligned} & Item(?it), Term(?t_1), Term(?t_2), Label(?t_3), \\ & hasTerm(?it, ?t_1), hasTerm(?it, ?t_2) \\ & \rightarrow isClassified(?it, ?t_3) \end{aligned} \quad (6.3)$$

Pour inférer l'intégralité des labels en fonction de la hiérarchie de labels, la règle SWRL suivante est ajoutée à l'ontologie afin de prendre en compte les relations hiérarchiques entre les labels :

$$\begin{aligned} & Item(?item), Label(?labelA), Label(?labelB), broader(?labelA, ?labelB), \\ & isClassified(?item, ?labelA) \rightarrow isClassified(?item, ?labelB) \end{aligned} \quad (6.4)$$

Comme décrit dans le chapitre 4, les règles de classification peuvent être appliquées par chaînage avant ou chaînage arrière, ce qui induit respectivement une classification avant la requête ou au moment de la requête. Dans le cas de l'architecture SEMXDM, la classification au moment de la requête est adaptée car la classification dépend de l'évolution du modèle au cours du temps, et n'est donc pas permanente (voir section 6.1.4). Le moteur d'inférence applique donc les règles par chaînage arrière lors du crawl. L'intégration du module de classification avec le module de crawling est décrite en figure 6.4.

Le résultat de la classification est un vecteur $v_{item_i}^{label}$, où chaque dimension est un label (classe *Label* de l'ontologie noyau). Le vecteur de labels est utilisé par le module priorité pour déterminer le potentiel de pertinence des liens extraits de l'item (section 6.1.5). Après qu'un item soit classé avec succès, sa similarité avec la requête initiale peut être calculée.

Comme décrit dans l'étude de la littérature, les crawlers ciblés standards ne parviennent pas à s'adapter à un environnement non contrôlé tel que le web au cours du temps Dong et al. (2014). D'après Dong et al. (2014), le modèle prédictif sur lequel s'appuie le crawler ciblé doit pouvoir s'adapter aux nouvelles données rencontrées, et aux changements induits dans les caractéristiques des items. Aussi, les crawlers devraient apprendre par expérience comment optimiser les temps de recherche des items pertinents, en se basant sur les chemins (liens hypertexte) connus menant à des items pertinents, Diligenti et al. (2000). Pour répondre à ces problématiques, l'architecture proposée se base sur deux modules supplémentaires décrits dans les sections suivantes. Un premier module de Maintenance a pour objectif l'adaptation du modèle prédictif en fonction de nouvelles données rencontrées. Le second module est le module de priorité, qui a pour objectif de déduire à partir des items déjà fetchés les chemins menant potentiellement vers des items

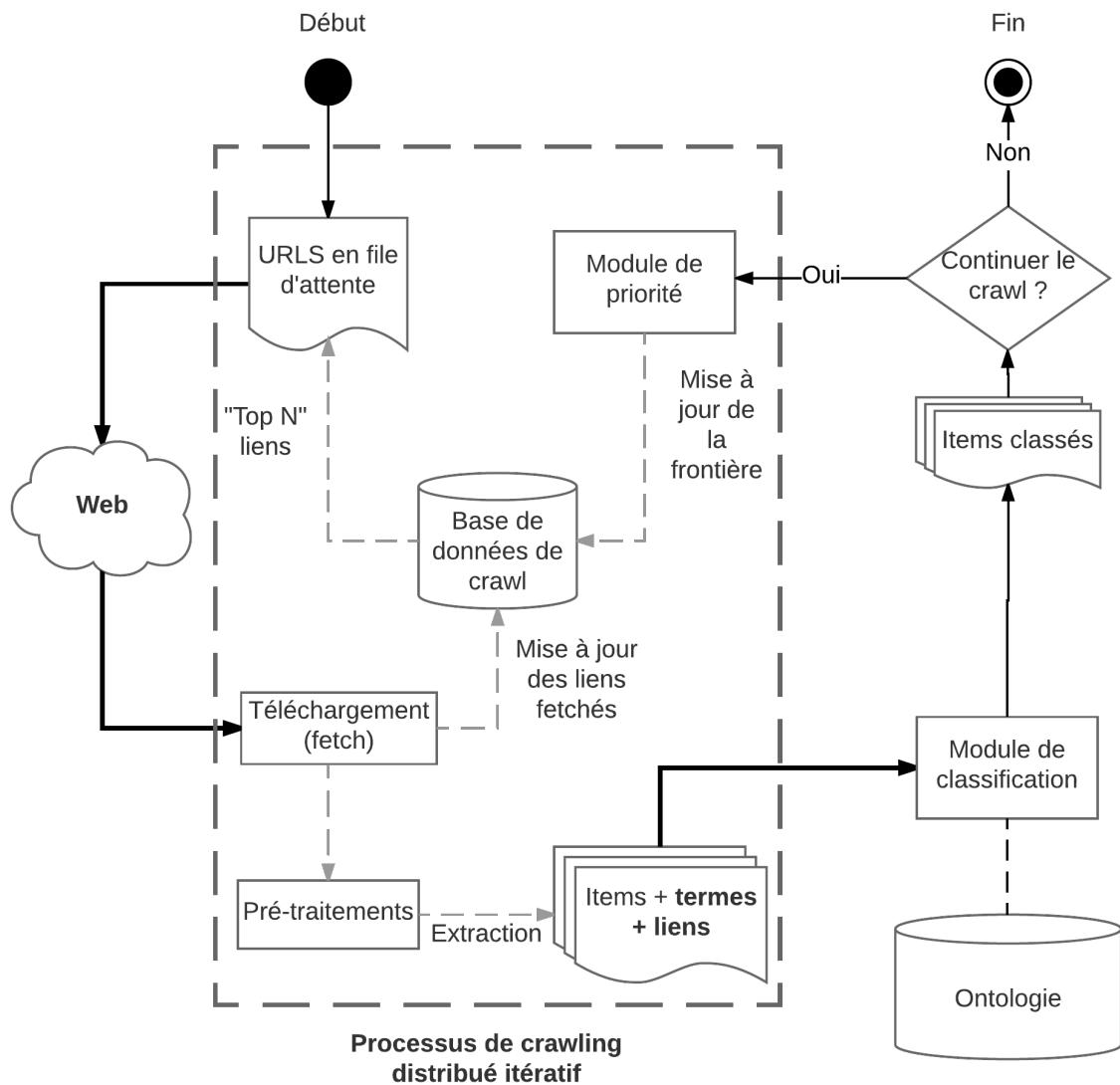


FIGURE 6.4 – Intégration des modules Crawler et Classification

pertinents. Ce dernier modèle calcule dans un premier temps la pertinence des items fetchés par rapport à la requête initiale. Dans un second temps, le module réordonne la frontière du crawl, en adaptant la priorité des items de la frontière en fonction de la pertinence de leurs parents dans le graphe web.

6.1.4/ MODULE D'ÉVOLUTION

Dans un environnement web peu structuré et non contrôlé, le contenu est dynamique et en constante évolution. Les données sont considérées comme non stationnaires, les propriétés des données issues du web évoluent au cours du temps. Ces changements représentent une difficulté pour produire un modèle prédictif représentatif des données, car ils induisent des changements radicaux dans les classes cibles (targets concepts). Cette problématique est nommée dérive conceptuelle (concept drift), Widmer et al. (1996).

Dans ce cas de figure, les performances d'un modèle prédictif sont impactées négative-

ment, car les données d'entraînement utilisées pour le construire ne correspondent plus aux données réelles. La classification des données réelles est alors dégradée.

Nous proposons de palier ces difficultés par l'intégration du module d'évolution dans l'architecture. Ce module est basé sur l'approche d'apprentissage adaptatif de Peixoto et al. (2015a), qui apporte constamment des modifications au modèle prédictif de classification représenté par une ontologie. Le processus permet d'adapter le modèle prédictif en fonction d'un flux non stationnaire de données non structurées. Peixoto et al. (2015a) a montré que les performances pour la tâche de classification de données inconnues sont améliorées lorsque le modèle prédictif est mis à jour par le processus d'apprentissage adaptatif.

Les données émises par le crawler sont utilisées comme entrée du module d'évolution. L'intégration du module est présentée en figure 6.5.

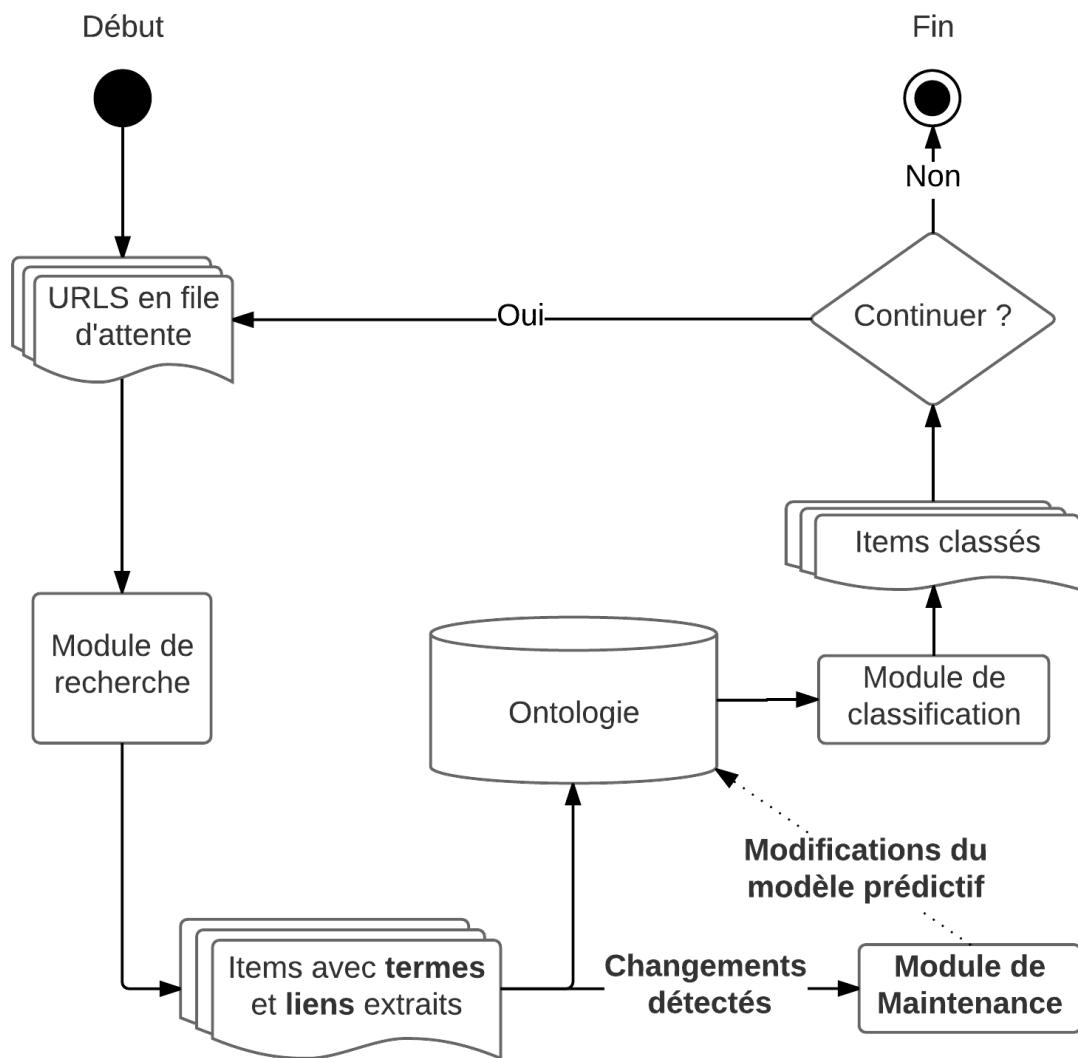


FIGURE 6.5 – Maintenance Module

Le processus d'apprentissage adaptatif se focalise sur l'adaptation du modèle prédictif décrit par l'ontologie, en fonction du flux de données textuelles issu du module de craw-

TABLE 6.2 – Types d'entrées détectées (Input Changes)

Input Change Type	Description
newItem	Un nouvel item est fetché et ajouté à l'index inversé des items
newTerm	Un nouveau terme est extrait d'un item
cooccurrence	Mise à jour de la fréquence d'un terme
appliedModification	Modification Request applied with success

ling. Trois caractéristiques de gestion de flux de données continues sont considérées par le processus :

- Évolution des attributs (feature-evolution), qui correspond à l'apparition de nouveaux termes (features), i.e. des termes jusqu'alors absents de l'index inversé des items. L'évolution des attributs concerne aussi la disparition de termes qui ne sont plus représentatifs des données, lorsque leur fréquence diminue.
- Évolution des concepts (concept-evolution), qui correspond à l'addition ou la suppression des attributs de classes, i.e. es labels dans notre approche.
- Dérive des concepts (concept-drift) : le flux constant de données apporte des changements dans les propriétés statistiques relatives des concepts et des attributs, ce qui induit des modifications du modèle prédictif (règles de classification et hiérarchie de labels).

L'approche d'apprentissage adaptatif de Peixoto et al. (2015a) répercute l'impact des trois caractéristiques ci-dessus dans le modèle prédictif. Le processus est basé sur une adaptation incrémentale du modèle de classification, Hulten et al. (2001), Aggarwal et al. (2006), selon les spécificités de l'ontologie noyau décrites dans le tableau 6.1. L'architecture de module de maintenance n'est présentée que de façon superficielle dans ce manuscrit. Une description exhaustive de l'approche peut être trouvée dans Peixoto et al. (2015a).

La figure 6.6 présente l'architecture interne du module de maintenance. Le processus est divisé en 4 étapes : (1) détection des changements depuis le flux de données, (2) Dérivation des requêtes de modification à partir des changements détectés, (3) Traduction des requêtes de modifications en modifications appliquées au modèle, et (4) application des modifications (évolution du modèle). La dernière étape considère la résolution des inconsistances résultant de l'application des modifications selon 4 sous-étapes : 4.1 application des changements, 4.2 vérification de la consistance, 4.3 résolution des inconsistances et 4.4 application définitive (commit) des modifications.

Les changements de propriétés statistiques, sont nommées Input Changes dans le processus d'apprentissage. Les différents types de changements sont capturés par des capteurs (change sensors) dédiés. Le tableau 6.2 décrit les différents types de changements détectés dans le flux de données fetchées.

Les changements détectés et validés en entrée, ou "Input Changes", génèrent des "Requêtes de Modification" au cours du crawl pour adapter le modèle de classification. Le tableau 6.3 décrit chaque type de requête de modification qui affectent le modèle de classification. Chaque entrée de cooccurrence implique une mise à jour de la fréquence document du terme, ce qui impacte les proportions relatives aux termes cooccurrents.

Les requêtes sont propagées en tant que "Modifications du modèle", i.e. les mises à jour

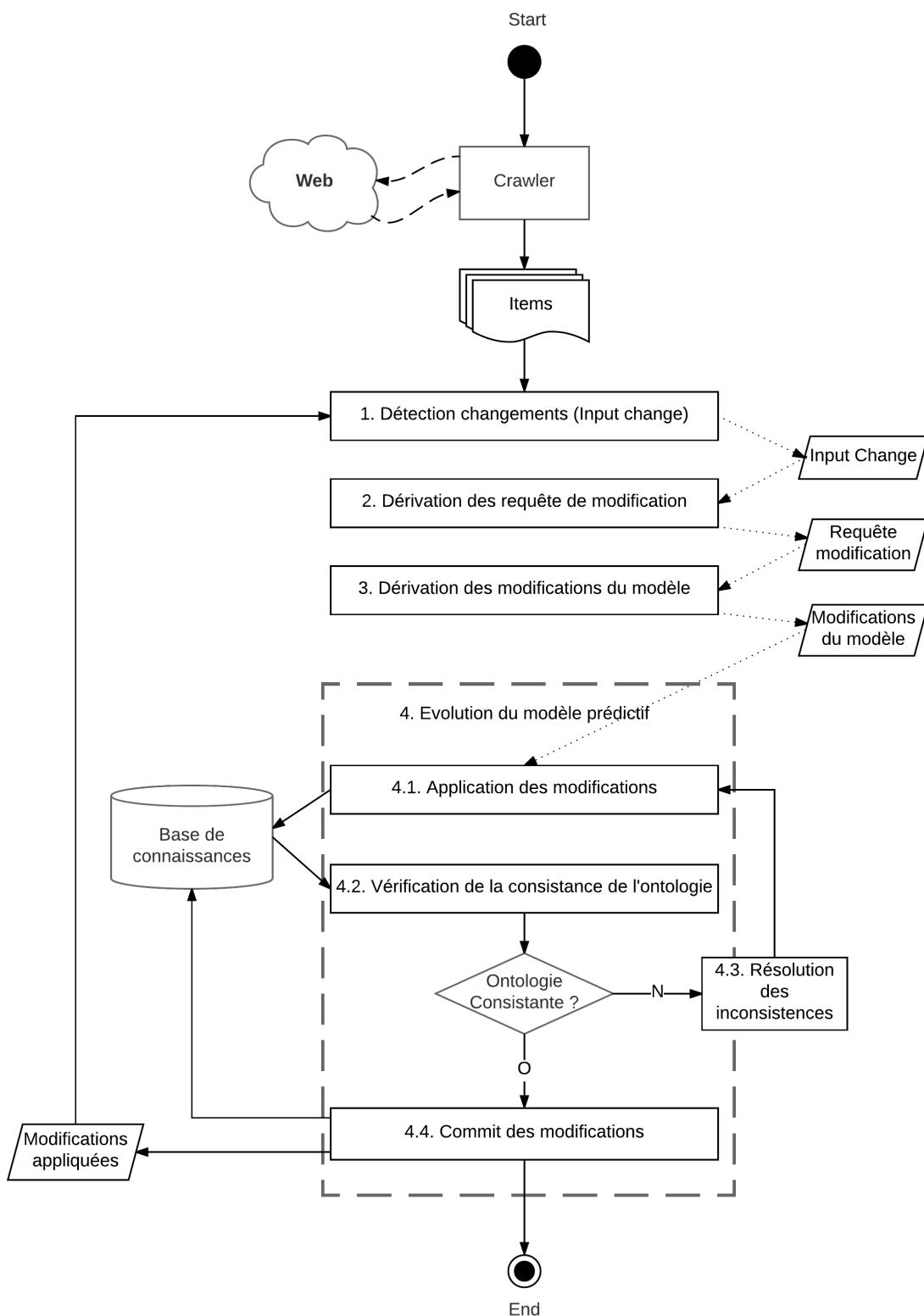


FIGURE 6.6 – Architecture du module de maintenance

de l'ontologie. Les mises à jour du modèle peuvent être effectuées après chaque cycle de crawl ou en parallèle du crawl, suivant le cas d'utilisation. Dans le second cas, le

TABLE 6.3 – Types de requêtes de modification du modèle

Requête de modification	Description
AddTerm	Ajoute un <i>Term</i> au modèle de classification
AddLabel	Ajoute un <i>Label</i> au modèle de classification
DeleteLabel	Supprime un <i>Label</i> du modèle de classification
AddHRelation	Ajoute une relation hiérarchique entre deux labels
DeleteHRelation	Supprime une relation hiérarchique entre deux labels
AddAlphaTerm	Ajoute une relation Alpha entre un label et un terme
DeleteAlphaTerm	Supprime une relation Alpha entre un label et un terme
AddBetaTerm	Ajoute une relation Beta entre un label et un terme
DeleteBetaTerm	Supprime une relation Beta entre un label et un terme

résultat de la classification d'un item est impacté après chaque changement du modèle de classification, issu d'autres items.

Comme décrit dans la section précédente, l'architecture SemXDM effectue la classification des nouveaux items au moment de la requête. L'évolution de modèle est donc prise en compte de façon transparente, sans interruption du processus.

6.1.5/ MODULE DE PRIORITÉ

L'objectif du module de priorité est de combiner les informations issues du graphe web et le contenu des items pour améliorer les décisions du crawler, i.e. pour redéfinir les priorités des items de la frontière de crawl. En opposition au module de classification qui calcule le score des items fetchés, le module de priorité calcule un score potentiel des items de la frontière (items non fetchés).

Pour chaque item $item_i$, le module de priorité peuple l'ontologie avec les liens extraits des items en tant qu'instances (Abox). Le tableau 6.4 décrit les concepts de logique de description ajoutés à l'ontologie noyau (Tbox) pour représenter les liens extraits des items.

TABLE 6.4 – Représentation du graphe web dans l'ontologie

Concepts DL	Description
$Link \sqsubseteq asString.String$	Stocke une URL
$Item \equiv \exists hasLink.Link$	Relation liant un item à son URL
$Item \equiv \exists hasOutlink.Item$	Lien sortant d'un item
$Item \equiv \exists hasInlink.Item$	Lien entrant d'un item
$hasOutlink \equiv hasInlink^{-}$	hasInlink et hasOutlink sont inverses
$Item \sqcap Link \equiv \perp$	Item et Link sont disjoints
$Term \sqcap Link \equiv \perp$	Term et Link sont disjoints

Les propriétés *hasInlink* et *hasOutlink* représentent les liens entrants et sortant pour chaque item intégré dans la base de connaissances.

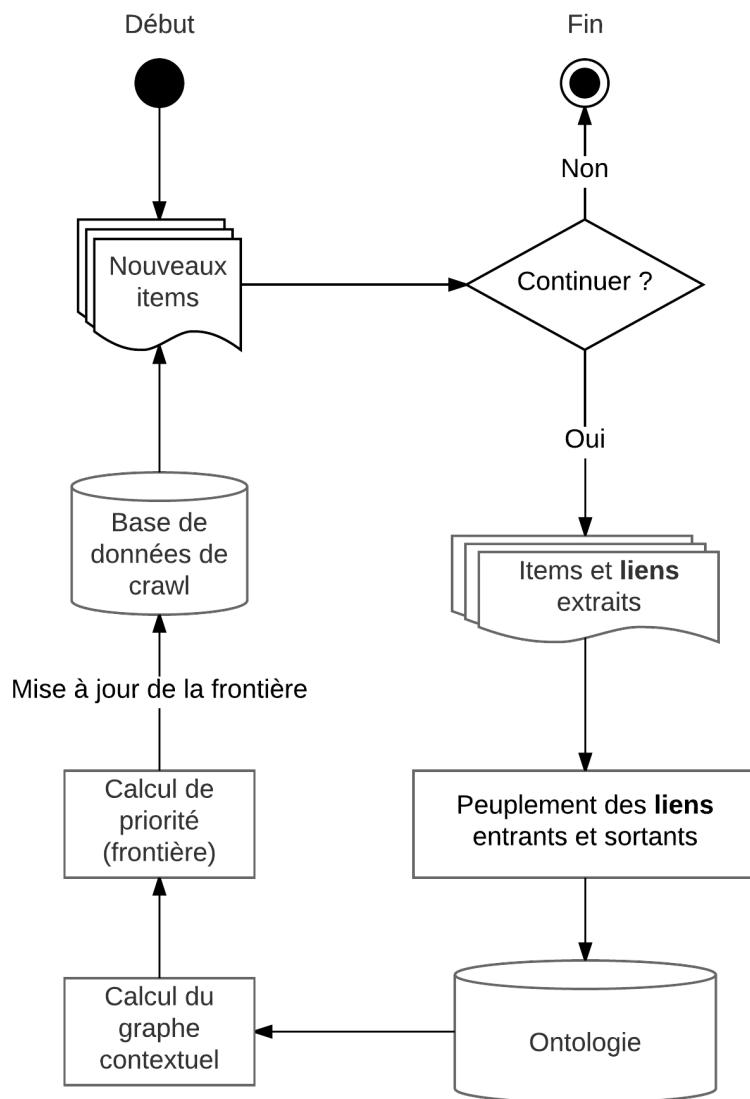


FIGURE 6.7 – Module de priorité

Les liens sont mis à jour dans la base de connaissances après chaque cycle de crawl. Les liens sortants (outlinks) sont mis à jour en fonction des liens extraits des items lors du crawl. Les liens entrants (inlinks) sont la relation opposé des liens sortants et sont mis à jour en conséquence. L'ajout de liens entrants ne fait pas appel à un processus externe tel qu'une requête à un moteur de recherche⁴, par conséquent les liens entrants sont incomplets.

Le résultat du peuplement de l'ontologie est l'ensemble de propriétés $hasInlink \omega_{in}^{item_i} = |\{item_1, item_2, \dots, item_n\}|$ et $hasOutlink \omega_{out}^{item_i} = |\{item_1, item_2, \dots, item_m\}|$, où n est le nombre total de liens entrants, et m le nombre total de liens sortants extraits des items.

Pour identifier les chemins menant potentiellement vers des items pertinents, l'approche graphe contextuel de Diligenti et al. (2000) est utilisée. Dans cette approche, les items

4. Certains moteurs de recherche peuvent être requêtés pour retrouver l'ensemble des liens entrant d'une page

pertinents sont d'abord identifiés. A partir de ces items, les liens entrants de ces items sont extraits de façon récursive. L'ensemble des liens forme un sous-graphe, découpé en couches. Chaque couche correspond à une distance à un item pertinent. La figure 6.8 présente un exemple de graphe contextuel à 3 couches, construit à partir d'un ensemble de pages pertinentes (couche 0).

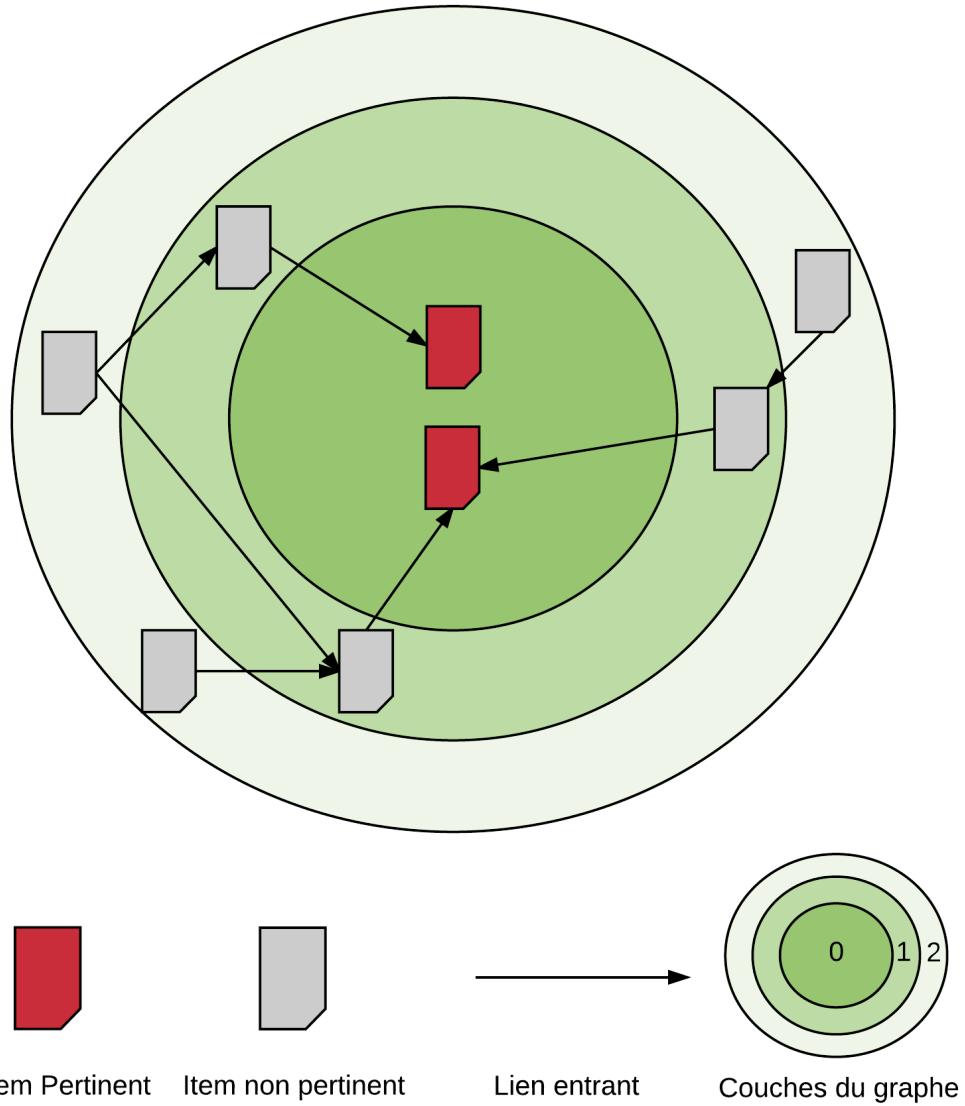


FIGURE 6.8 – Exemple de graphe contextuel à 3 couches

Après chaque cycle de crawl, les couches du graphe contextuel sont recalculées à partir de l'ensemble des items pertinents à jour. Un seuil de pertinence θ défini par l'utilisateur permet de discriminer les pages non pertinentes. L'ensemble des items pertinents $\omega_{rel}^{item_i}$ est défini par :

$$\omega_{rel}^{item_i} = \{item_i | \forall item_i \in Item : Relevance_i > \theta\} \quad (6.5)$$

où $Relevance_i$ est le score de pertinence de l'item (voir section 6.1.3). Lors du premier cycle de crawl, l'ensemble d'items pertinents est composé des seed (pages initiales) fournies par l'utilisateur.

A partir de l'ensemble $\omega_{rel}^{item_i}$, chaque couche L_j du graphe contextuel est définie par l'ensemble des items à une distance de j liens entrants d'un $item_i \in \omega_{rel}^{item_i}$:

$$L_j = \omega^{item_k} = \{item_k | \forall item_k \in Item : \exists path(item_k, item_i)\} \quad (6.6)$$

où $path(item_k, item_i)$ est le plus court chemin dans la base de connaissances entre $item_i$ et $item_k$, composé de j relations *hasInlink* ($|path| = j$). Par exemple, $L_1 = \omega^{item_k}$ où $\forall k, \exists item_k hasInlink(item_i)$.

Chaque couche du graphe contextuel est mise à jour avec l'ensemble des termes contenus dans les items qui la composent : L_j est définie par un couple $(term_k, valeur_k)$ tel que :

- $\forall j, item_j \in L_j$
- $\forall k, term_k \in Term$
- $\forall k, \exists item_j hasTerm(label_k)$
- $\forall j, valeur_k = |item_j|, item_j hasTerm(term_k)$

Un vecteur pondéré v_i^{term} est créé pour chaque couche L_i , où chaque dimension du vecteur est un terme. Chaque vecteur v_i^{term} est mis à jour après chaque phase de crawl avec les nouveaux termes et liens découverts. Un second vecteur $v_{item_i}^{label}$ est généré, où chaque dimension est un *Label* dans l'ontologie. La probabilité qu'un item appartienne à une couche L_i est ensuite approximée par calcul de la distance entre les vecteurs de termes/labels d'une couche du graphe et du nouvel item.

$$\cos(v_{L_i}^{term}, v_{item_j}^{term}) = \frac{v_{L_i}^{term} \cdot v_{item_j}^{term}}{\|v_{L_i}^{term}\| \cdot \|v_{item_j}^{term}\|} \quad (6.7)$$

Le calcul de similarité entre les deux types de vecteurs est identique. Les deux types de vecteurs sont ensuite utilisés dans le calcul du degré d'appartenance de chaque couche du graphe. On définit le degré d'appartenance d_i pour chaque couche du graphe comme la moyenne arithmétique des deux vecteurs :

$$d_i = \frac{\cos(v_{L_i}^{term}, v_{item_j}^{term}) + \cos(v_{L_i}^{label}, v_{item_j}^{label})}{2} \quad (6.8)$$

Le **score de pertinence** final est défini ci-dessous. Une heuristique supplémentaire est employée pour palier les limitations du modèle de classification, qui ne peut permettre de décrire avec précision l'ensemble des items dans un environnement web⁵ :

$$r_{item_j} = \max(\cos(v_{L_i}^{term}, v_{item_j}^{term}), d_i) \quad (6.9)$$

où $r_{item_j} \in [0, 1]$.

5. Cette problématique nommée "incomplete label assignment", est définie par le coût important de création d'un jeu de données complètement étiqueté, i.e. le jeu d'entraînement ne comprend pas tous les labels possibles. Cette problématique est d'autant plus importante dans un contexte Big Data, où le nombre d'items à étiqueter est important, rendant la création d'un modèle parfaitement représentatif des données quasiment impossible Yu et al. (2014).

Cette heuristique permet de conserver une priorité importante pour les items dont seule la distance des vecteurs de termes est élevée.

Le score de pertinence résultant, compris dans l'intervalle $[0, 1]$ est utilisé pour estimer la distance, en nombre de liens, entre un nouvel item et un item pertinent (couche L_0). Les priorités de la frontière de crawl sont mises à jour après chaque crawl en fonction de cette distance. Plus une page est proche de la couche L_0 , plus sa priorité est élevée (une priorité fixe est attribuée à chaque couche du graphe). Les liens qui ne correspondent à aucune couche ont une priorité minimale dans la frontière.

Après chaque cycle de crawl, le graphe contextuel est mis à jour : la première couche du graphe est composée de l'ensemble des items dont la pertinence $\omega_{L_0}^{rel}$ dépasse un seuil θ défini par l'utilisateur, i.e. tous les items dont le score de pertinence avec la première couche du graphe est supérieure au seuil : $r_{item_i} > \theta$. Pour le premier cycle, la couche L_0 du graphe est uniquement composée du vecteur de termes issu de la requête initiale.

Les autres couches du graphe sont modifiées de façon dynamique, en fonction du graphe de liens mis à jour.

6.2/ IMPLÉMENTATION

L'architecture est implémentée en Java, et déployée sur un cluster Hadoop⁶, où l'ensemble des processus distribués sont exécutés. La plateforme de traitement de données Hortonworks⁷, basée sur Ambari, permet de suivre l'exécution du processus. Les nœuds du cluster sont équipés avec un processeur quadri cœur Intel I5 et 8GB de mémoire vive. Le triple-store Stardog est déployé sur un serveur dédié avec les mêmes ressources matérielles.

Les spécificités d'implémentation des différents modules sont décrites ci-dessous.

6.2.1/ MODULE DE RECOMMANDATION

Les items du web récupérés par le système sont indexés dans une base de données Apache Solr⁸, qui permet l'indexation et la recherche des items. Les différentes requêtes utilisateurs (requête mot-clé et requête-item) sont traitées différemment. Les requêtes mot-clé sont envoyées au serveur Apache Solr pour retrouver les items les plus pertinents dans l'index. Solr permet nativement de répondre aux requêtes par mots-clés, et de classer les items en fonction de leur pertinence via un calcul de similarité basé sur le modèle vectoriel. La chaîne de traitement du langage décrite en section 3.2 est utilisée pour normaliser les termes extraits des items ainsi que les mots-clés des requêtes. Les requêtes-item sont gérées par le module de recherche (crawler), qui lance une tâche de recherche d'information sur le web. Les items retrouvées par le crawler sont ensuite indexés et ajoutés à la base existante d'items.

6. <http://hadoop.apache.org/>

7. <https://fr.hortonworks.com/>

8. <http://lucene.apache.org/solr/>

6.2.2/ MODULE DE RECHERCHE

Apache Nutch⁹ est utilisé comme base de développement du crawler. Nutch est un crawler distribué performant, à la base du développement de l'écosystème Hadoop. Des classes dérivées de Nutch permettent d'intégrer l'ensemble des modules de l'architecture, notamment via une extension des plug-in de pré-traitement (parsing), d'indexation (envoi et stockage dans la base Solr), et de scoring par similarité¹⁰. Chaque item est indexé avec son contenu (titre et texte extrait de la page web), son score de similarité en fonction de la requête initiale, et la date de téléchargement.

Le résultat de la requête est ensuite envoyé à l'utilisateur. Les items retrouvés sont présentés à l'utilisateur par ordre décroissant de pertinence.

6.2.3/ MODULE DE CLASSIFICATION

Le module de classification est basé sur l'étape de réalisation de l'architecture SHMC, présentée dans le chapitre 4.

La librairie OWL-API est utilisée pour la conversion des items et de leurs attributs (termes) et leur insertion dans l'ontologie (Abox). Un triple-store performant pour de grands volumes de données nommé Stardog¹¹ est utilisé pour intégrer l'ensemble des éléments composant l'ontologie, i.e. le modèle prédictif et les instances des nouveaux items (Tbox+Abox). Stardog est également utilisé pour effectuer la classification par chaînage arrière dont la méthode a été décrite dans la section 4.2. L'optimisation par sélection des règles est également employée dans ce processus pour réduire le temps de classification des items.

6.2.4/ MODULE DE MAINTENANCE

Le module de maintenance a été développé à partir de l'architecture de maintenance adaptative décrite dans les travaux de thèse de Rafael Peixoto.

Un ou plusieurs flux de données alimentent le processus de maintenance. L'architecture globale est construite dans une optique de déploiement sur un cluster Hadoop. On considérera cependant un seul flux de données provenant du crawler distribué dans l'évaluation du processus.

Le processus de maintenance est développé autour des frameworks Apache HBase¹², Apache Storm et Apache Kafka¹³. L'intégration de Storm et Kafka pour le traitement de flux de données est basé sur des "Topologies" et des "Spout" respectivement issus de Storm et de Kafka. Une topologie est un graphe de traitement du flux : chaque nœud d'une topologie effectue un traitement du flux, et les liens entre ces nœuds indiquent comment le flux de données transite dans le graphe. Les "Spout" dérivent des consommateurs de flux (Consumers) de Kafka. Les Spouts sont des unités de gestion de flux de données sur un cluster de machines. Chaque consommateur est affilié à un groupe

9. <http://nutch.apache.org/>

10. <https://wiki.apache.org/nutch/SimilarityScoringFilter>

11. <http://docs.stardog.com>

12. <https://hbase.apache.org/>

13. <http://storm.apache.org/>, <https://kafka.apache.org/>

(consumer group) qui définit quel type de données seront traités par le consommateur. Les différents flux sont caractérisés par des enregistrements (records), représentés par un tuple clé/valeur de façon similaire au paradigme MapReduce. Le framework est ensuite responsable de l'envoi des flux de données correspondant à chaque consommateur, et des problématiques inhérentes au traitement de grands volumes de données (distribution de la charge de calcul et allocation des ressources matérielles). La répartition de la charge de calcul est gérée par partitionnement du flux d'entrée pour chaque groupe de consommateurs, et par redistribution des flux en fonction du nombre de consommateurs disponibles.

La figure 6.9 présente l'intégration des frameworks Storm et Kafka pour traiter les flux de données dans l'architecture SemXDM. L'implémentation du processus de maintenance ne fait cependant pas partie de l'objet de ces travaux, et est présentée de façon exhaustive dans Peixoto et al. (2016a).

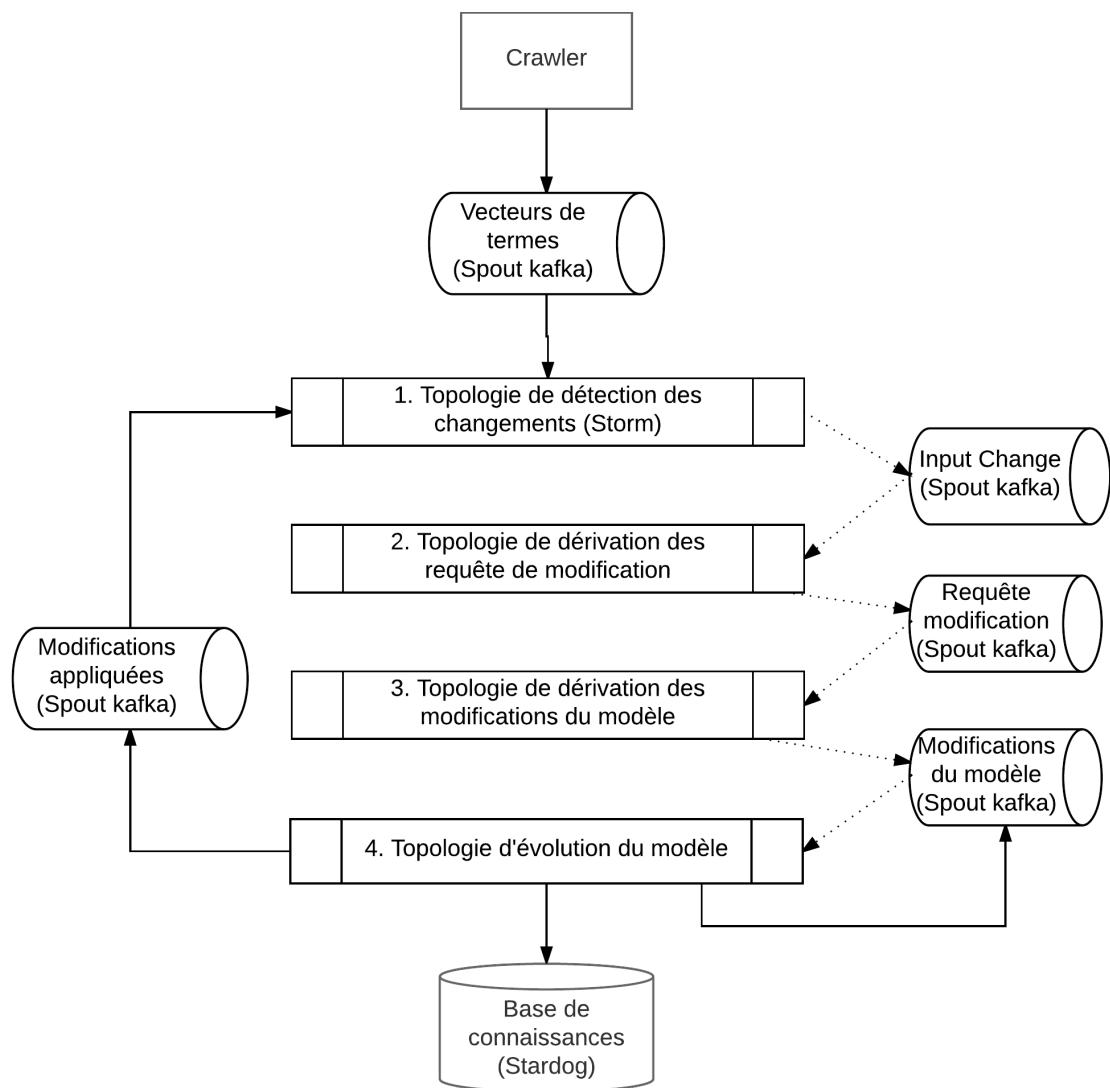


FIGURE 6.9 – Implémentation du module de maintenance

Le module de recherche extrait les vecteurs de termes pondérés correspondant à chaque

item. Ces vecteurs représentent le flux de données en entrée du processus de maintenance. La première topologie (déttection des changements) est responsable du traitement du flux en entrée.

Tous les échanges de données sont loggés¹⁴ dans une base de données HBase. Cette base assure le traitement de chaque type de flux (figures 6.6.1 à 6.6.4), et la persistance des modifications apportées au modèle. Un accès à la base est requis pour logger chaque type de données.

6.2.5/ MODULE DE PRIORITÉ

Le peuplement de l'ontologie avec les liens entrants et sortants est effectué via la librairie OWL-API, après extraction du contenu des items. Le peuplement est effectué simultanément au peuplement des items et des termes associés décrits dans le module de classification.

La construction du graphe contextuel consiste en le peuplement des liens dans le triple store (Stardog), alors représenté par une sous-partie du graphe RDF. Au début de chaque cycle d'un crawl, le graphe contextuel est redéfini, par requête SPARQL¹⁵ au triple store.

6.3/ ÉVALUATION

Cette section présente une évaluation du processus de recherche d'information SemXDM décrit précédemment. Premièrement, les méthodes existantes d'évaluation des robots d'indexations seront décrites. Les résultats seront ensuite présentés, selon deux axes distincts : une évaluation globale de l'approche pour la tâche de croisement d'information, et l'analyse de l'impact individuel de chacun des modules du processus sur la performance de l'approche.

6.3.1/ MÉTHODES D'ÉVALUATION DES ROBOTS D'INDEXATION

- **Taux de récolte :** Le taux de récolte (harvest rate) est une métrique utilisée pour évaluer plus précisément la proportion d'items pertinents retrouvés par le crawler. Sa définition générique est la suivante :

$$\text{Harvest_Rate} = \frac{|\text{items_pertinents}|}{|\text{items_fetchés}|} \quad (6.10)$$

La notion de pertinence est cependant dépendante du cas d'utilisation. La pertinence peut être définie par une autre métrique, par exemple une mesure de similarité avec une requête initiale, ou l'appel à un processus dédié externe au crawler.

14. En informatique, on parle de *log* (diminutif de *logging*) pour désigner un fichier, ou tout autre dispositif, permettant de stocker un historique des évènements attachés à un processus. Ces évènements sont horodatés et ordonnés en fonction du temps. En clair, le *log* est un peu le "journal de bord" d'un système. On parle parfois en français de *fichier journal* pour désigner les logs. Il sera consulté en cas de besoin, par exemple pour essayer d'identifier l'origine d'une panne ou l'auteur d'une intrusion (réussie ou manquée). Source : <https://www.1min30.com/dictionnaire-du-web/log>

15. SPARQL est le langage de requête standard de données RDF <https://www.w3.org/TR/rdf-sparql-query/>

- **Similarité moyenne** : La similarité moyenne

$$S_{\text{imilarité_moyenne}} = \frac{\sum sim(item_i)}{|items_fetchés|} \quad (6.11)$$

- **Précision, Rappel, F1-mesure** :

Ces métriques ont déjà été introduites dans la section 4.3.1 pour la tâche de classification. Dans le cas de l'évaluation d'un crawler, elles permettent d'évaluer la performance de la classification des pages web. Pour un crawler thématique par exemple, la précision correspond à la proportion de pages correctement classées par le crawler comme correspondant au thème. Ces métriques sont des indicateurs de performances objectives, mais sont difficiles à mettre en place dans un environnement non contrôlé pour un grand volume de données. Il est nécessaire de pouvoir confirmer/infirmer le résultat de la classification des pages retrouvées par le crawler. Ces métriques sont donc généralement employées dans un environnement contrôlé, par exemple pour classer les pages d'un seul domaine ou d'un jeu de données fixe, où le nombre de vrais positifs et faux positifs est déterminé à l'avance.

La performance du module de classification ayant déjà été développée dans les chapitres précédents, elle ne sera pas abordée dans les évaluations de ce chapitre.

6.3.2/ ÉVALUATION QUANTITATIVE

L'objectif de cette évaluation et de démontrer la faisabilité de l'approche, et d'observer le comportement de chaque module pour une requête d'exemple (requête-item).

6.3.2.1/ ENVIRONNEMENT DE TEST

Un article économique est utilisé comme requête initiale de croisement d'information. Un premier classifieur est entraîné à partir d'un jeu de données pré-étiqueté composé de 45000 articles économiques, via l'architecture SHMC. Ce classifieur est réutilisé dans l'évaluation qualitative ("classifieur 45k"). Une liste de 1200 URL vers des sites web liés à la presse économique française est utilisé comme frontière initiale du crawl. Cette liste est fournie par l'entreprise Actualis partenaire de ces travaux.

Le module de priorité est composé de 3 couches (couches 0 à 2). La première couche du graphe est composé de l'item-requête initial, tandis que les couches 1 et 2 sont vides à l'initialisation. Le seuil de pertinence θ est fixé à 0.5, i.e. les items dont le score de pertinence r_{item_i} est supérieur à θ sont assignés à la première couche du graphe. Cette valeur est volontairement restrictive afin de conserver une similarité importante avec la requête initiale et limiter l'expansion du graphe.

6.3.2.2/ RÉSULTATS

Les données récoltées à partir de la tâche de crawl sont divisées par cycle (itération) de crawl ou "round". A chaque cycle, les "top N" items dans la frontière sont fetchés, et les statistiques de pertinence sont récoltées pour ce round. Dans cette évaluation, on définit $N = 2000$ items. Le nombre total d'items fetchés est environ égal à 15000 sur 8 cycles

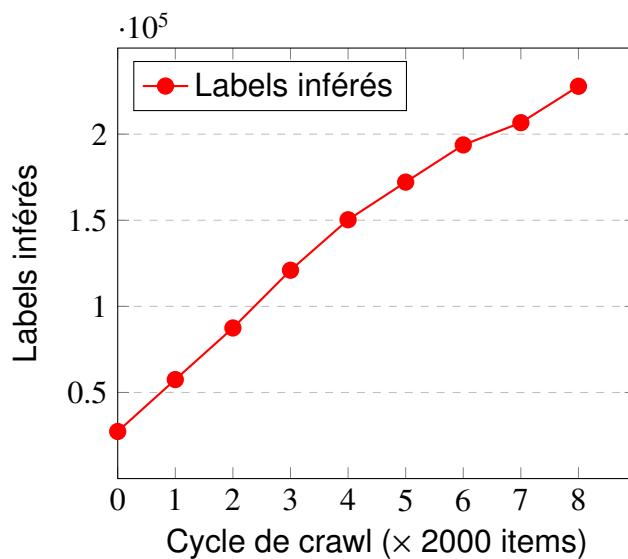


FIGURE 6.10 – Nombre de classifications (inférences) total

(le téléchargement de certains liens échoue naturellement). La figure 6.10 présente le nombre de labels inférés (i.e. les classifications) par le module de classification à chaque cycle.

Le nombre de classifications augmente logiquement en fonction du nombre d'items fetchés. L'augmentation du nombre de classifications est également fonction de l'augmentation du nombre de règles (figure 6.16).

Les figures 6.11, 6.12 et 6.13 présentent le nombre de changements détectés par le module de maintenance. Les types de changements observés correspondent aux types exposés dans le tableau 6.2. Le nombre d'items fetchés étant fixe (2000 items environ, selon les éventuelles erreurs de fetch), on observe le nombre de cooccurrences détectées, le nombre de nouveaux termes et le nombre de modifications appliquées au modèle prédictif (applied modifications). Bien qu'une importante variation du nombre de modifications apportées au modèle soit observée au cycle 3, l'ordre de grandeur reste stable. Nous n'avons pas trouvé de cause directe de cette variation.

La figure 6.11 montre l'évolution du nombre de cooccurrences détectées après chaque cycle de crawl. Le nombre total est relativement élevé (51M de cooccurrences en moyenne pour moins de 2000 items), ce qui impacte grandement la performance de l'approche en temps de calcul. Du fait de l'implémentation de l'approche, chaque cooccurrence induit un accès à HBase ce qui impacte négativement le processus de maintenance.

Bien qu'une variation du nombre de cooccurrence entre les différents cycles soit observée, l'ordre de grandeur reste stable ([49M, 53M]).

La figure 6.12 présente le nombre de nouveaux termes détectés à l'issue de chaque cycle de crawl. Ces termes viennent s'ajouter à l'ensemble de termes qui composent la matrice de cooccurrence : ils correspondent à une nouvelle ligne de la matrice pour laquelle les cooccurrences seront ensuite détectées. On peut voir que le nombre de nouveaux termes décroît à chaque cycle, ce qui est un comportement attendu du crawler ciblé. L'objectif du crawler étant de rechercher sur des items similaires entre eux, le nombre de

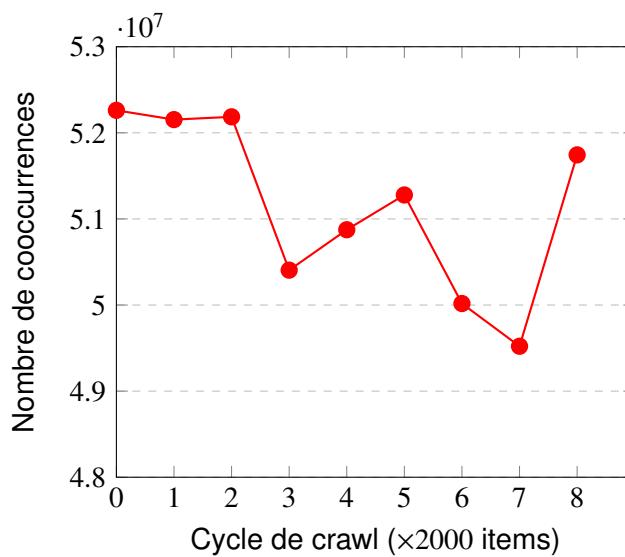


FIGURE 6.11 – Nombre de cooccurrences détectées par cycle (maintenance)

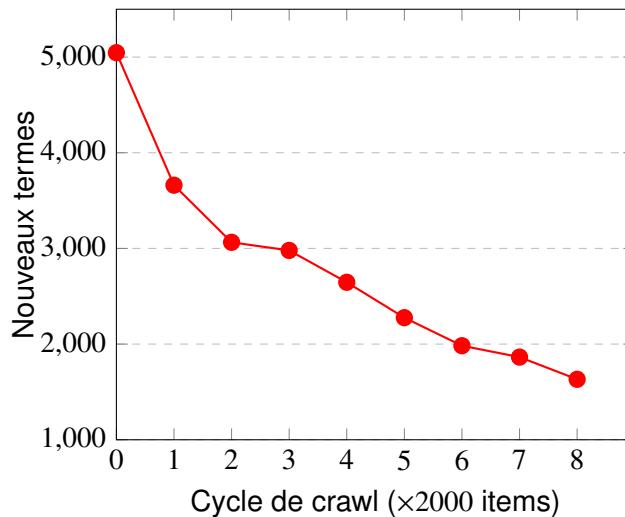


FIGURE 6.12 – Nombre de nouveaux termes détectés par cycle

nouveaux termes décroît en conséquence. Il est important que le crawler respecte cette caractéristique, afin de limiter l'augmentation de la taille de la matrice et par extension le nombre de cooccurrences détectées.

La figure 6.14 montre l'évolution du graphe contextuel à chaque cycle, i.e. le nombre d'items retenus pour chaque couche L_i du graphe au début de chaque cycle.

Il est possible d'observer qu'aucun item n'est considéré comme très pertinent avant le cycle 5. Une valeur plus basse du seuil θ permettrait de faire évoluer le graphe contextuel plus tôt, mais la similarité du graphe avec la requête initiale serait alors impactée. Les caractéristiques de l'évaluation sont également à prendre en compte (nombre de liens fetchés par cycle, nombre de liens initiaux). Ces caractéristiques semblent adaptées à une recherche horizontale plutôt qu'à une recherche en profondeur rapide (verticale), limitant

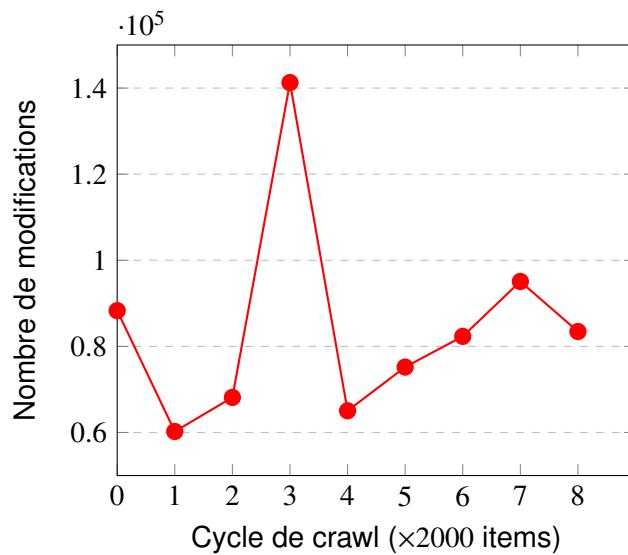


FIGURE 6.13 – Nombre de modifications appliquées au modèle prédictif

la capacité d'exploitation du crawler. L'évaluation qualitative de la section suivante tente de pallier ce problème en augmentant le nombre de cycles et en réduisant le nombre de pages fetchés par cycle. Ces observations sont soulignées dans la figure 6.17 de l'évaluation qualitative (section suivante), qui présente la performance de l'approche pour la même requête initiale.

Enfin, les figures 6.15 et 6.16 montrent l'évolution du modèle prédictif en fonction des changements détectés par le module de maintenance. Les valeurs initiales (cycle 0) correspondent aux propriétés initiales du modèle de classification, généré à partir des données d'entraînement (45k items). La grande majorité des modifications appliquées au modèle prédictif sont issues des cooccurrences détectées, et impactent donc principale-

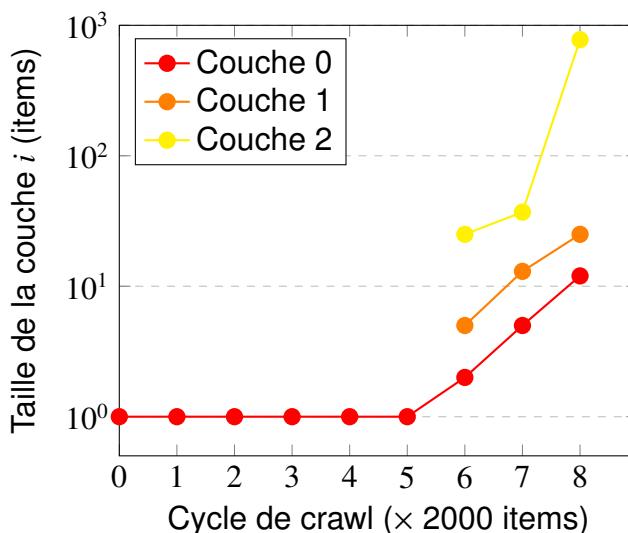


FIGURE 6.14 – Taille du graphe contextuel

ment les règles de classification. Cette observation est en accord avec les observations de Peixoto et al. (2016a). Le nombre moyen de termes et de règles ajoutés au modèle est

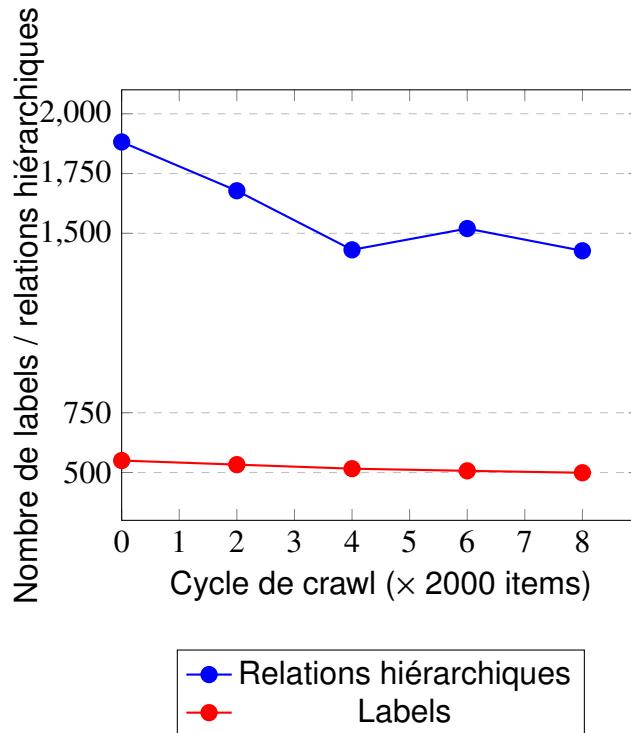


FIGURE 6.15 – Evolution de la hiérarchie de labels

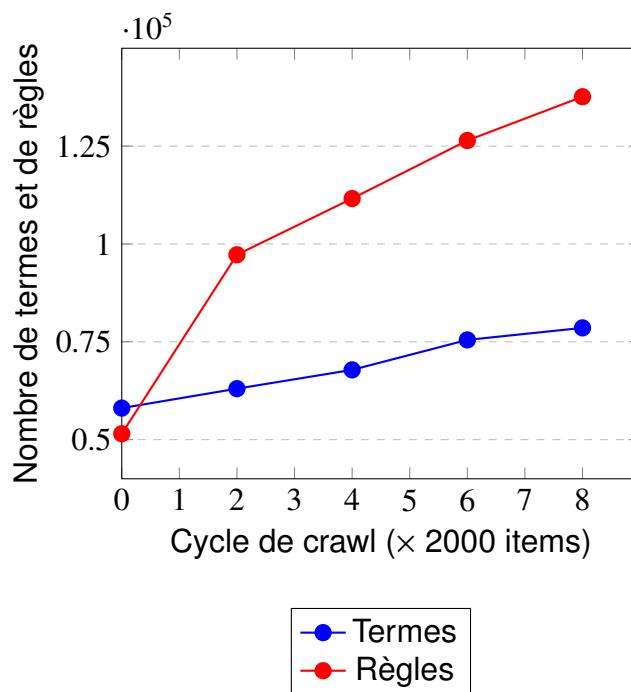


FIGURE 6.16 – Evolution des termes et des règles

cependant plus élevé. Cette différence s'explique par la provenance des items. Les items provenant du web induisent un ensemble de termes plus varié. Aussi, la taille moyenne des items en nombre de termes est plus importante ce qui induit un nombre de cooccurrences élevé.

Une limitation du processus est le coût calculatoire important du processus de maintenance. Le temps de traitement de la maintenance, de la réception de l'item à son impact sur le modèle est élevé comparativement aux autres modules (de plusieurs secondes à plusieurs minutes en fonction de la taille de l'item en nombre de termes). En effet le nombre total de cooccurrences dans un item peut-être très important s'il contient beaucoup de termes, ce qui induit un nombre de modifications propagées importantes. Les limitations techniques de l'environnement de test expliquent en partie cette latence, qui rend le processus de maintenance difficilement applicable en conditions réelles d'utilisation. Tout comme il est proposé dans le chapitre 3 pour la création de la matrice de cooccurrence, il est possible de réduire la fenêtre de détection de cooccurrences pour grandement réduire le coût calculatoire de la maintenance, i.e. en définissant une distance maximale entre deux termes au sein d'un item pour que la cooccurrence soit validée.

6.3.3/ ÉVALUATION QUALITATIVE POUR LE CROISEMENT D'INFORMATION

Cette évaluation s'intéresse à la performance de l'approche proposée pour la tâche de croisement d'information. L'objectif est d'évaluer la capacité du processus pour la recherche d'information pertinente à partir d'une requête-item initiale, et d'observer l'impact des différents modules sur la performance du crawler. Les métriques de taux de récolte et de similarité moyenne introduites dans cette section sont utilisées pour comparer l'approche à un crawler ciblé standard de type Best-N-first. Ce crawler considère la distance lexicale entre la requête initiale et les items fetchés pour attribuer la priorité des nouveaux liens extraits des items. La distance cosinus entre le vecteur de termes de l'item initial et les nouveaux items définit la pertinence des nouveaux items. Le processus d'extraction de termes des items est identique pour notre approche et l'approche standard.

Pour évaluer l'approche et la comparer à l'approche Best-N-First, les métriques de similarité moyenne et de harvest rate présentées précédemment sont utilisées. La similarité des items est définie comme la distance cosinus entre le vecteur de termes d'un item et la première couche L_0 du graphe (la première couche du graphe contient initialement l'item-requête). On définit le harvest rate comme la proportion d'items fetchés dont la distance cosinus dépasse un seuil, fixé à 0.25, i.e. $\cos(\mathbf{v}_{L_0}^{term}, \mathbf{v}_{item_j}^{term}) > 0.25$.

On compare la performance de l'approche SEMXDM à une implémentation de l'approche Best-N-First pour une même requête de croisement d'information. Le harvest rate et la similarité moyenne sont utilisées pour comparer les deux approches. L'objectif pour chaque approche est donc de maximiser ces deux métriques.

La figure 6.17 présente la comparaison entre les deux approches pour une requête de croisement d'information dans les mêmes conditions initiales que l'évaluation précédente.

Les résultats montrent que la performance de l'approche SEMXDM est supérieure à l'approche Best-N-First, selon les deux métriques utilisées. En particulier, le taux de récolte montre que la proportion d'items très pertinents est significativement supérieure après le cycle 5, ce qui est un point notamment essentiel pour la tâche de croisement d'informa-

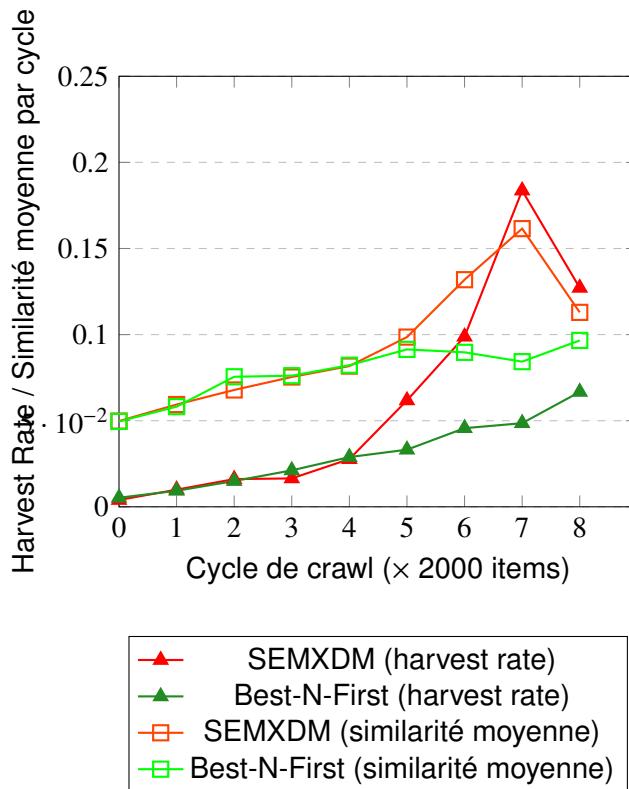


FIGURE 6.17 – Similarité moyenne et taux de récolte par cycle (requête de croisement d'information)

tion.

Ces résultats indiquent que pour une requête-item, l'utilisation du **score de pertinence** comme fonction de priorité à partir des vecteurs de termes et de labels permet de maximiser la performance du crawler. Ces résultats doivent cependant être raffinés par une analyse en conditions réelles pour un grand nombre de requêtes.

D'après nos observations, le crawler tend à exploiter les sources pertinentes seulement après plusieurs cycles. Le nombre d'items fetchés par cycle est relativement élevé (2000) et le nombre de cycles total faible (9), ce qui ne favorise pas une exploitation rapide des sources pertinentes. Les conditions d'expérimentations sont modifiées dans les tests suivants afin de corriger ce biais : un nombre plus important de cycles (40) de taille plus faible (500 items) est alors employé.

6.3.3.1/ IMPACT DE L'ÉVOLUTION DU MODÈLE

Deux modèles de classification sont comparés, construits respectivement à partir de 45 000 et 135 000 items, que l'on nomme respectivement classifieur 45k et classifieur 135k.

Dans le cas où le modèle est peu adapté aux données (classifieur 45k), le calcul du score de pertinence semble impacté par une similarité des labels trop faible en moyenne (figures 6.20 et 6.21), ce qui induit une performance proche de l'approche Best-N-First pour la métrique de similarité moyenne (figures 6.18 et 6.19), i.e. l'heuristique définie

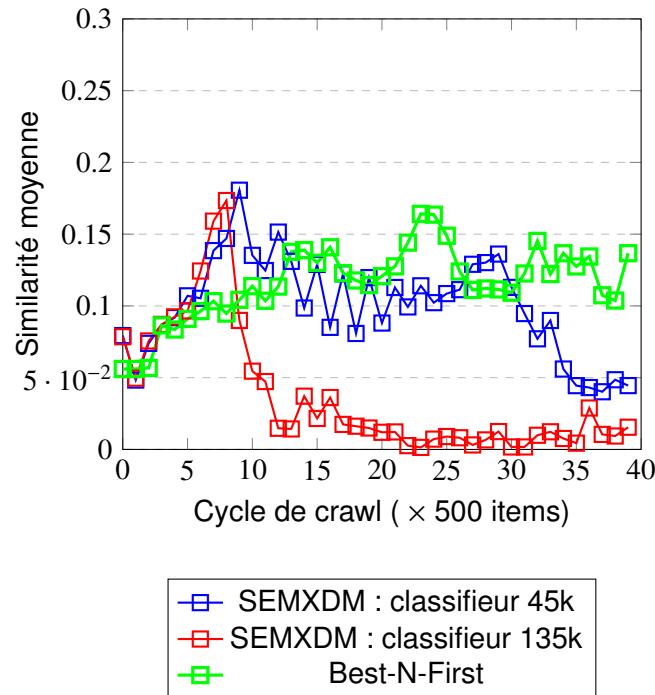


FIGURE 6.18 – Similarité moyenne par cycle

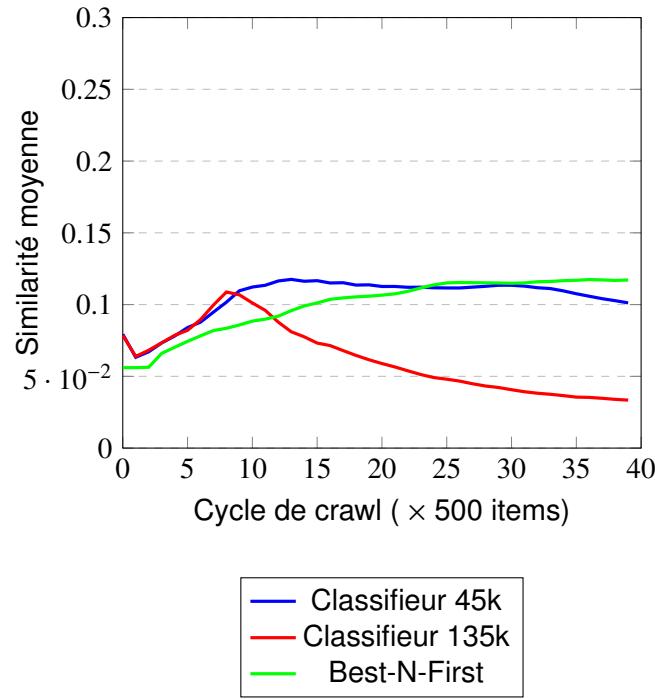
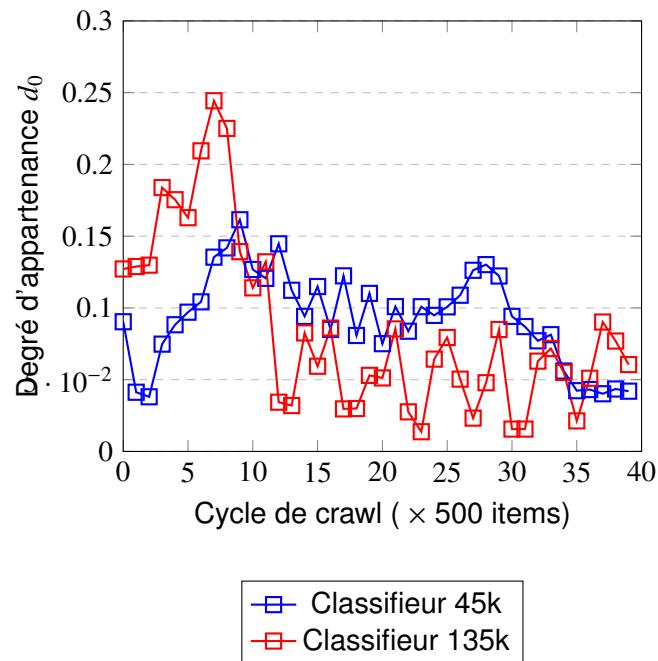
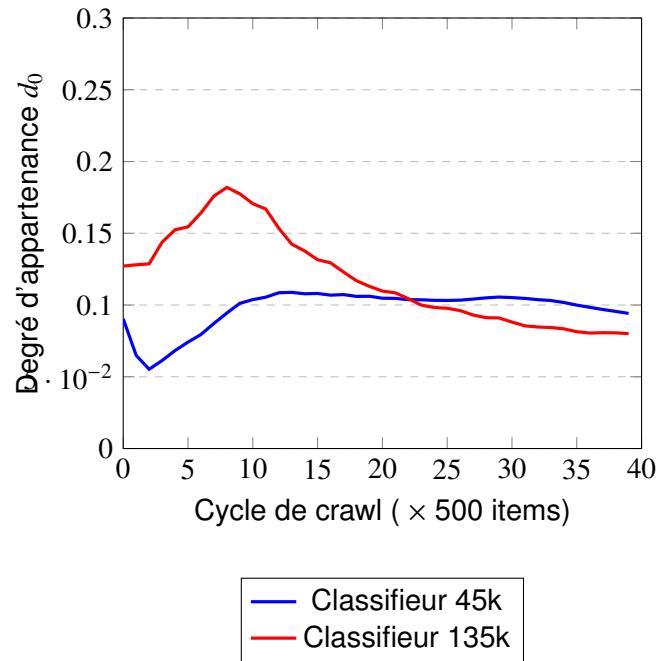


FIGURE 6.19 – Similarité moyenne globale

dans le calcul du **score de pertinence** final considère uniquement la distance cosinus des termes directement extraits des items.

Lorsque le modèle est plus représentatif des données (classifieur 135k), l'attribution de

FIGURE 6.20 – Degré d'appartenance d_0 par cycleFIGURE 6.21 – Degré d'appartenance d_0 global

priorité des liens basée sur le **score de pertinence** est performante dans les premiers cycles (0 à 10) où elle permet de maximiser à la fois la similarité moyenne (figures 6.18 et 6.19) et le **degré d'appartenance** (figures 6.20 et 6.21). Le **score de pertinence** permet d'optimiser la découverte d'items pertinents à ce stade.

En revanche on observe une perte de performance de l'approche dans la suite du crawl (cycles > 10). Dans le cas où le modèle est basé sur un jeu de données plus conséquent (classifieur 135k, courbe rouge), le degré d'appartenance décroît et la similarité moyenne n'est pas maintenue dans le temps (figures 6.18 à 6.21).

Pour palier les limitations des stratégies d'affectation de priorité, il est possible d'alterner ou de combiner les stratégies de crawl. Pour illustrer cette solution, nous proposons de combiner l'approche SEMXDM et l'approche Best-N-First pour une même tâche.

Les figures 6.22 et 6.23 comparent les résultats du crawl avec une stratégie unique (figures 6.20 et 6.21) et une stratégie hybride entre l'approche SEMXDM et l'approche Best-N-First. La figure 6.20 indique une diminution importante de performance de l'approche SEMXDM à partir du dixième cycle. Pour comparer les deux stratégies, la frontière du crawl à ce cycle est partagée pour les deux stratégies. Les résultats sont identiques jusqu'au dixième cycle à partir duquel la stratégie Best-N-First est employée.

La stratégie hybride permet de maximiser le score de pertinence dans les premiers cycles grâce à l'impact positif de la classification, et conserve par la suite un degré de pertinence similaire à l'approche Best-N, là où notre approche tend à perdre en performance dans le temps.

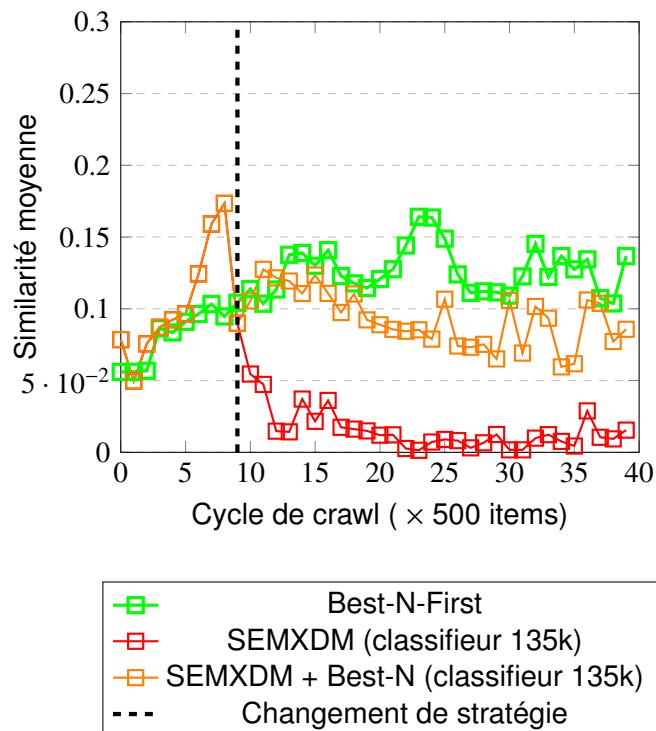


FIGURE 6.22 – Similarité moyenne par cycle

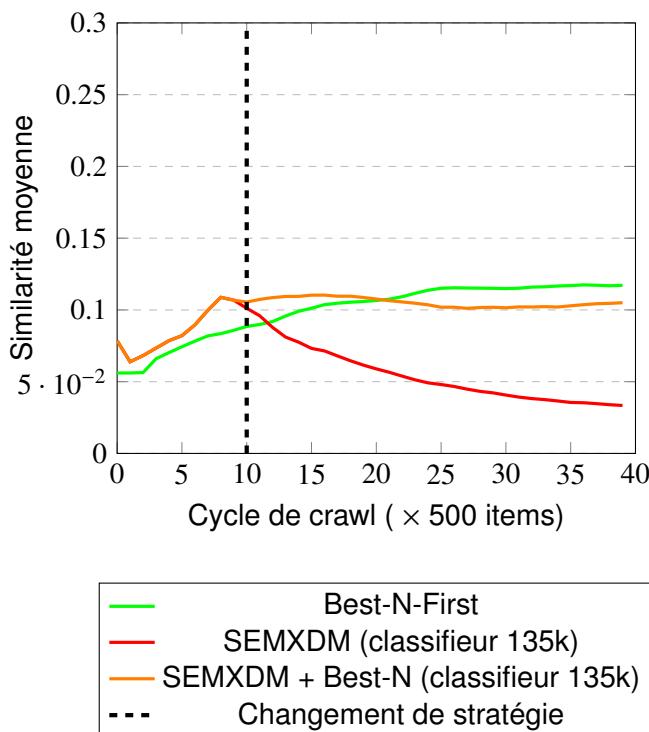


FIGURE 6.23 – Similarité moyenne globale

6.4/ CONCLUSION

Ce chapitre a décrit une architecture de recherche d'information nommée SEMXDM. L'approche est basée sur un crawler ciblé, afin de répondre aux problématiques métier identifiées dans le chapitre 1. L'approche tire partie de l'exploitation d'un modèle de classification sémantique pour décrire les items du web, dont la construction est basée sur l'approche SHMC (chapitres 3 et 4). Afin de répondre aux limitations de l'état de l'art identifiées dans le chapitre 5, l'approche considère premièrement l'adaptation du modèle de classification dans le temps. Deuxièmement, une nouvelle méthode de définition de la priorité dans la frontière du crawl est présentée, basée sur une approche de graphe contextuelle (voir section précédente).

Nous avons montré dans l'évaluation qualitative que l'approche SemXDM permet une amélioration continue de la performance du crawler, comparativement à un crawler ciblé non adaptatif (Best-N-First). L'impact des modules de classification et de priorité dans le score de pertinence pour l'évaluation de la priorité des liens dépasse les résultats obtenus uniquement par l'approche Best-N-First (figure 6.17). Cette amélioration de performance se traduit dans les premiers résultats par la maximisation du taux de récolte et de la similarité moyenne avec une requête initiale (item-requête).

Dans un second temps, l'impact de l'évolution du modèle dans le temps (module de maintenance) a été évalué. Deux modèles de classification sont comparés, construits respectivement à partir de 45k et 135k items. Nous observons une nette amélioration de la performance de l'approche dans les premiers cycles de la recherche pour le second modèle (classifieur 135k). La performance de l'approche décroît cependant dans la suite

de la recherche (courbe rouge). Notre approche semble être performante pour discréderiter rapidement les informations pertinentes au départ de la recherche, mais ne parvient pas à maintenir une similarité moyenne élevée pendant toute la recherche contrairement à l'approche Best-N-First. L'approche Best-N-First semble donc plus appropriée pour une phase d'exploitation de sources pertinentes, tandis que notre approche est plus performante pour la phase d'exploration, où la frontière est composée d'un nombre important de liens peu pertinents¹⁶. Cette hypothèse est supportée par les résultats positifs de notre approche dans la première expérience, où le protocole correspond plus à une phase d'exploration.

Afin de maximiser la performance de l'approche durant toute la recherche, modifier la stratégie du crawler durant le crawl semble être pertinent. Le changement de stratégie permet de maximiser la performance du crawler tout au long de la recherche.

16. Les notions d'exploration et d'exploitation sont introduites dans le chapitre 5

APPLICATION AU DOMAINE DE LA VEILLE STRATÉGIQUE

Ces travaux de thèse s'inscrivent dans un partenariat industriel avec l'entreprise Actualis SARL. La société Actualis est spécialisée dans la production et la distribution de revues de presse économique, principalement en France métropolitaine, dans l'optique de fournir un support personnalisé de veille stratégique à des entreprises de tous secteurs d'activité. Un des objectifs de ces travaux est la mise en place de l'architecture de recherche et de qualification de l'information, développée dans les chapitres précédents, dans le contexte industriel d'Actualis, afin de répondre à des problématiques métier spécifiques.

Ce chapitre introduit d'abord le domaine d'activité de l'entreprise. Les objectifs du partenariat industriel ainsi que le système d'information de l'entreprise, qui fait l'objet d'une refonte dans le cadre de ce partenariat, sont ensuite décrits. Un prototype du nouvel outil de veille est en cours de développement. Cet outil est nommé First Watch. La description de l'architecture et le déploiement de ce prototype clôturent ce chapitre.

7.1/ INTRODUCTION : VEILLE STRATÉGIQUE

La veille stratégique, est *un type de veille informationnelle qui englobe l'ensemble des autres veilles, telles que la veille sociétale, la veille en entreprise, la veille concurrentielle, la veille commerciale, la veille fournisseur, la veille image, la veille juridique ou encore la veille technologique*¹.

Un nombre croissant d'entreprises décident d'entreprendre une activité de veille stratégique, afin d'analyser en temps réel leur environnement industriel (concurrence, opportunités commerciales, évolution du marché, etc.). Cette activité permet de prendre des décisions stratégiques importantes, et indirectement d'améliorer leur compétitivité.

La revue *FirstEco*, produit de l'entreprise Actualis, est un outil de veille qui fournit quotidiennement des synthèses de l'information économique aux clients de l'entreprise. Les limitations de cette revue ont été identifiées dans les travaux de recherche de Werner (2015). Cette revue était faiblement personnalisée, et ne permettait pas de recommander de façon précise les informations les plus pertinentes à chacun des clients. Les travaux de recherche de Werner (2015) ont permis la création d'un nouveau produit à destination des clients d'Actualis, nommé "*FirstEco Pro'fil*". Ce produit est basé sur un système de

1. Source : <http://www.digimind.fr/solutions/par-types/definitions/veille-strategique>

recommandation de nouvelles économiques, qui pallie les défauts de la revue initiale, par une approche ontologique, fondée sur une modélisation métier du domaine (la veille économique), permettant ainsi une qualification précise des items et une recommandation pertinente des items aux clients. *FirstEco Pro'fil* représente aujourd'hui environ 50 pour-cent des nouvelles ventes enregistrées par l'éditeur.

Le système de recommandation de l'entreprise Actualis, développé dans les travaux de Werner (2015), demande l'intervention d'experts du domaine à divers instants de la production de la revue *FirstEco Pro'fil*. Ils interviennent d'une part lors de la classification de l'information, i.e. l'étiquetage des données selon la sémantique du domaine, et d'autre part dans la recherche d'informations pertinentes sur le web, appuyée par divers outils de veille (progiciels). Ces deux aspects de leur travail sont des tâches complexes et fastidieuses compte tenu de la quantité d'informations à traiter chaque jour, et pour lesquelles il n'existe pas d'outil qui soit adapté pour automatiser et faciliter la charge de travail des experts. Un des objectifs des travaux présentés dans ce manuscrit est la création d'un tel outil.

Le système de veille automatisé le plus reconnu dans le monde est Google Actualités (Google News), le portail basé sur le moteur de recherche éponyme. La recherche et le regroupement d'articles effectué par Google Actualités est un cas typique d'activité de veille automatisée. Ce portail recense en temps réel des brèves issues principalement de sites de presse spécialisés, regroupe les informations identiques puis les met à disposition des utilisateurs. Le système se concentre principalement sur l'actualité nationale et internationale, triée dans une édition spécifique pour différentes régions géographiques (plus de 70 régions géographiques). De fait, Google Actualités est une source d'information primordiale pour l'information nationale et internationale dans des domaines variés (politique, économie, science, sport, ...). En revanche le portail n'est pas assez spécifique pour traiter de l'information très localisée (régions, départements et circonscriptions en France) et très spécialisée (les catégories sont générales par définition). Google Actualités est le fruit de plusieurs années de travail et naturellement, construire un système similaire est une tâche complexe.

Pour approcher du résultat observé sur Google Actualités, la conception d'un outil de veille automatisé doit répondre à plusieurs questions sous-jacentes :

- Comment identifier les sources d'information pertinentes sur le web ?
- Comment distinguer les pages contenant un article informatif des autres pages web ? Sur un site d'information, toutes les pages web ne contiennent pas un article traitant d'une information spécifique. Une distinction doit être faite entre les pages contenant un article, les pages d'index regroupant différentes sections du site, ou les pages décrivant brièvement les articles, i.e. par une liste de titres et de chapôs²...
- Comment extraire le texte de l'article automatiquement ? La structure hétérogène des pages web ne permet pas l'exécution d'un processus unique d'extraction du texte pertinent d'une page. L'utilisation d'algorithmes appropriés, éventuellement accompagnés d'heuristiques est nécessaire. Cette problématique a été abordée dans la section 5.1.1.
- Quels termes sont pertinents au sein de l'article ? L'indexation des articles en fonction de leur contenu doit discriminer les termes non pertinents, et extraire uniquement les termes permettant de définir le sens de l'information. Il s'agit d'une

2. Texte court coiffant un article, permettant d'amener le lecteur à entrer dans l'article.

problématique complexe, abordée dans la section 3.2.

- Comment regrouper les articles similaires, et quelle méthode de comparaison des articles utiliser ? Le regroupement d'articles similaires est essentiel pour que les informations présentées par le système de veille soient exhaustives. Cette propriété correspond à des notions de recherche d'information et de classification/clustering déjà abordées dans les chapitres 1 et 5. Le croisement d'informations similaires, tel qu'il est effectué par Google Actualités repose sur ces notions.

Le nouvel outil doit en outre s'articuler autour du système d'information existant, introduit en première partie de ce document, et être intégré aux processus de production de l'entreprise Actualis (voir annexe C). Le nouvel outil de veille impacte le processus d'agrégation/recherche d'information et le processus d'analyse/classification, tel que décrit dans la figure 7.1.

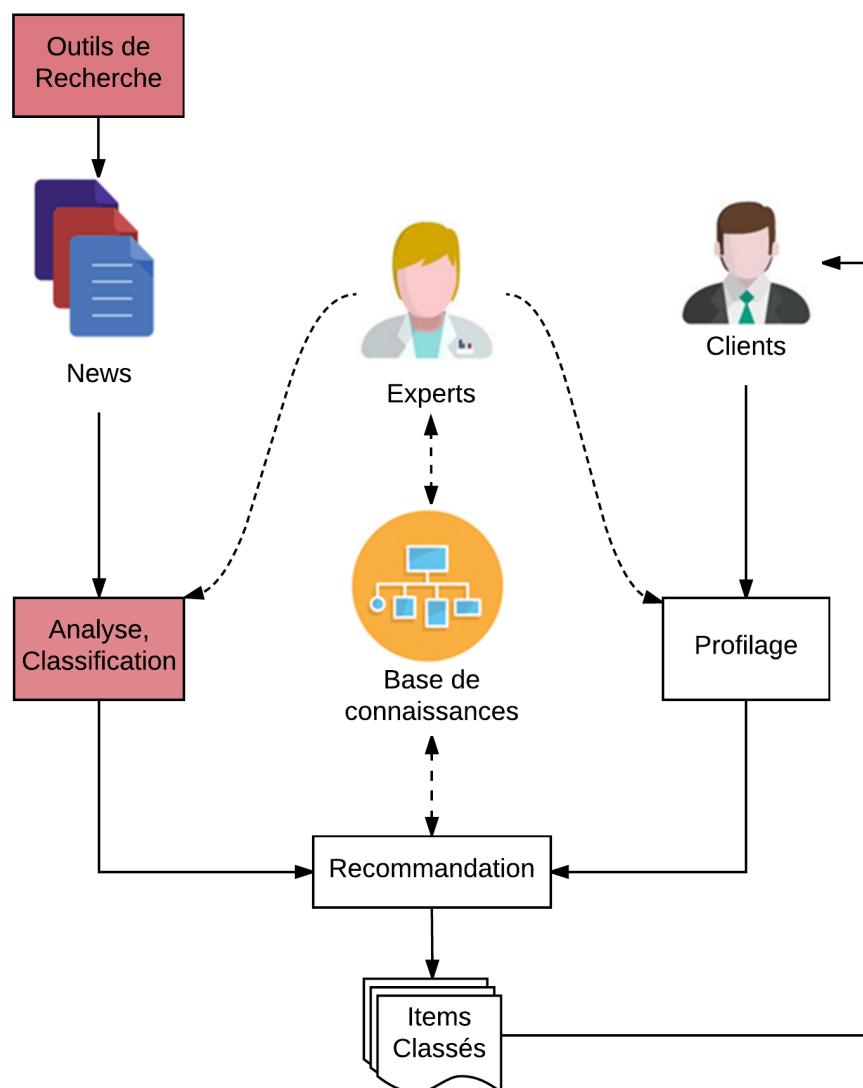


FIGURE 7.1 – Sections du processus de production métier impacté par le nouvel outil

Les sections suivantes décrivent brièvement le système de recommandation existant, la base de connaissances qui est centrale dans ce système, et les objectifs de mise en place du nouvel outil.

7.2/ SYSTÈME DE RECOMMANDATION EXISTANT

Les nouvelles économiques, ou articles, contiennent des informations variées, dont la pertinence diffère selon le contexte, i.e. le format de distribution, l'objectif d'utilisation, et le consommateur de l'information. Les différentes propriétés descriptives caractérisant les articles sont nommées facettes de description. Les facettes de description sont une vue de l'information, dans un contexte donné. De la même manière, le consommateur de l'information peut être décrit par un profil utilisateur lui-même définit par un ensemble de facettes de description. Les travaux de Werner (2015) se sont concentrés sur la description formelle et la mise en place technique de facettes de description dans le cadre d'un système de recommandation, et plus particulièrement sur la spécificité des facettes de description économiques. Chaque facette a été définie par un *vocabulaire d'indexation*, représentatif des connaissances métier détenues par les experts du domaine. Les différents vocabulaires sont intégrés dans une base de connaissances commune, qui permet une qualification précise de l'information et une description précise des profils utilisateurs.

Une fois un profil utilisateur établit selon un ensemble de facettes descriptives, un processus de recommandation permet alors de sélectionner les articles les plus pertinents en fonction des besoins des utilisateurs.

7.2.1/ BASE DE CONNAISSANCES

Les facettes de description sont les propriétés caractérisant un item, i.e. un profil ou un article. Ces propriétés portent sur toute information caractéristique de cet item. Par exemple, les facettes telles que le nom de l'auteur, le nombre de caractères ou de mots, la région géographique concernée, permettent une description des articles. Toutefois, en fonction de l'objectif d'utilisation, elles ne sont pas toutes pertinentes Werner (2015). L'objectif des experts est de catégoriser les profils et les articles en fonction de facettes descriptives économiques, qui sont plus susceptibles d'être vues comme pertinentes par un client, comparativement aux informations administratives (date de publication, auteur, sources, type d'article).

Dans le cadre de la recommandation d'articles aux clients, la base de connaissances peut alors être divisée en trois parties : les *vocabulaires d'indexation* correspondants aux facettes, une base des articles indexés via ces vocabulaires, et une base des profils également indexés via ces vocabulaires. La base de profils indexés contient les descriptions des profils des utilisateurs tels qu'ils ont été créés par les experts en charge des relations clients lors d'un processus d'indexation manuelle. La base d'articles indexés contient les descriptions des articles rédigés par les documentalistes. Le processus d'indexation des profils et des articles ne sont pas détaillés dans ce manuscrit. On s'intéressera par la suite uniquement à la catégorisation des articles, qui est l'objet de la mise en place du modèle prédictif.

Les principales facettes de description utilisées dans la catégorisation des items sont divisées en trois catégories :

- Le **secteur économique**. Ces facettes définissent les secteurs économiques traités par l'article, par exemple la métallurgie, l'informatique, ou les énergies renouvelables.
- Le **thème**. Ces facettes définissent les événements économiques traités par l'article, par exemple un investissement, un changement de locaux, etc.

- La **localisation**. Ces facettes définissent définit les villes, régions, départements et pays traités dans l'article.

Pour chaque type de facette, un vocabulaire contrôlé est défini dans la base de connaissances. Les vocabulaires d'indexation contiennent les termes nécessaires à l'indexation des articles et des profils. Les termes sont organisés sous la forme de vocabulaires contrôlés et structurés correspondant aux facettes descriptives des items.

7.2.2/ PROCESSUS DE RECOMMANDATION

À partir de l'information contenue dans les nouvelles économiques, celles-ci sont catégorisées selon ces facettes.

La figure 7.2 présente un exemple de nouvelle économique indexée manuellement selon différentes facettes de description dans la revue FirstEco Pro'Fil. Les facettes de localisation sont représentées en rouge, les facettes de thème en vert et les facettes de secteur économique en bleu.

FIGURE 7.2 – Exemple d'article Indexé

Le processus de recommandation consiste à proposer aux clients les informations pertinentes qui correspondent le plus à leur profil. Les profils utilisateurs et les articles correspondent à un vecteur étendu de facettes provenant des différents vocabulaires d'indexation. Un algorithme spécifique de calcul de distance vectorielle et un ensemble d'heuristiques permettent de calculer un score de pertinence d'un article pour un profil Werner (2015). Le score de pertinence définit l'ordre d'apparition des nouvelles proposées au client. La figure 7.3 présente un exemple de recommandation d'articles par rapport à un profil utilisateur.

7.2.3/ OBJECTIFS DE L'APPLICATION

Dans le cadre de leur travail de veille quotidien, les documentalistes qui rédigent la revue FirstEco utilisent un logiciel de veille, WebSite Watcher³, pour rechercher de nouvelles informations et les faire suivre aux clients. Cette étape de veille est à la base du processus

3. <http://www.website-watcher.fr/>

The screenshot shows a user interface for managing and filtering articles. At the top, there are links for 'MES ARTICLES', 'MES FILTRES', 'MA SELECTION', 'MES ENVOIS', 'RECHERCHE 360°', 'FRANCE ET MONDE', 'APPELS D'OFFRES', and 'ANNONCES LEGALES'. Below this is a search bar with the placeholder 'Vos articles filtrés du 19 juin 2017 au 30 juin 2017'. On the left, there are three sections: 'Localités' (listing ON (01) Ain (1), ON Franche-Comté (0), ON Limousin (0)), 'Thèmes' (listing ON Création d'entreprise (0), ON Difficulté de l'entreprise (1), ON Développement financier de l'entreprise (0)), and 'Secteurs d'activité' (listing ON Banque, finance et assurance (0), ON Energie (1), ON Recherche & Développement (0), ON Transport et logistique (0)). On the right, a specific article is highlighted: 'Incendie à la centrale du Bugey' (published on June 19, 2017). The article summary includes tags like (01) AIN, CENTRALE NUCLÉAIRE, SINISTRE, and SINISTRE. It describes an incident at the Bugey nuclear power plant where a fire occurred in reactor number 5, leading to the activation of an emergency center in Montrouge. Below the article are links for 'Site Nucléaire du Bugey', 'Sources', 'Publié le 21 juin 2017', and options for 'Lecture confortable', 'Ajouter à ma sélection', and sharing.

FIGURE 7.3 – Exemple de recommandation. Le profil utilisateur est décrit à gauche par un ensemble de facettes.

de création et de recommandation des produits FirstEco (revue, newsletter, FirstProfil⁴). La dimension temporelle du travail de veille est primordiale au sein de l'entreprise : le temps de veille est limité par définition, et cet exercice est renouvelé chaque jour. Par conséquent, les documentalistes se doivent d'être efficents dans la gestion du temps de veille. Le projet de recherche dans lequel s'inscrivent ces travaux est défini dans ce cadre par "*le développement d'un outil de veille automatisée, capable de classer et diffuser l'information à l'interne ou à l'externe*". Ce projet a pour motivation l'amélioration des conditions de travail des documentalistes (gain de temps), et la mise à disposition de nouvelles fonctionnalités utiles à la veille documentaire. Pour atteindre ces objectifs, le nouvel outil doit corriger les défauts du logiciel de veille actuel, i.e. WebSite Watcher, tout en ajoutant de nouvelles fonctionnalités pour la recherche, l'analyse et la structuration de l'information. Le projet est plus précisément composé de deux outils complémentaires, qui correspondent aux deux axes de recherche développés dans les parties 2 et 3 de ce manuscrit :

- Un outil de **collecte** qui permet de récolter de grands volumes d'informations depuis le web. L'objectif de cet outil est de permettre la recherche précise d'une information, et potentiellement d'augmenter le nombre de sources d'informations de la base de sources de veille, par la découverte de nouvelles sources d'information.
- Un outil d'**analyse**, permettant de qualifier précisément l'information (classification) selon les vocabulaires contrôlés d'Actualis, i.e. l'ontologie de domaine définie par les 3 facettes (thèmes, secteurs économiques et localisation) décrites dans la section précédente.

Différentes fonctionnalités communes à ces deux outils ont été définies dans la conception du nouvel outil. Certaines fonctionnalités sont déjà présentes dans WebSite Watcher, mais sont limitées. En outre, l'outil doit être modulable afin d'ajouter des fonctionnalités supplémentaires dans le futur.

4. <http://www.firsteco.fr/>

L'outil d'**analyse** effectue la classification de l'information selon les entités de la base de connaissance. Il s'agit d'une fonctionnalité prioritaire de l'application. L'objectif n'est cependant pas de remplacer l'indexation manuelle des articles. La précision du résultat étant critique pour les clients, le résultat de la classification doit toujours être supervisé par les documentalistes.

L'outil de **collecte** permet de récolter de grandes quantités d'information économiques de façon automatique. Dans le cas de la refonte du processus de veille, il doit répondre aux besoins suivants :

- Mise à jour de la base de sources d'informations de façon périodique (quotidien). La mise à jour des sources doit permettre d'afficher les sources contenant une nouvelle information. Un problème sous-jacent est la gestion du bruit lors de la mise à jour (pages légèrement modifiées mais ne contenant pas de nouvelle information).
- Extraction automatique du texte pertinent dans les pages articles : dans WebSite Watcher, cette fonctionnalité est déléguée aux filtres. Ces filtres sont relativement efficaces, mais demandent un temps de paramétrage important, et sont difficiles à répliquer pour un grand nombre de sources. Le nouvel outil doit permettre d'automatiser le processus d'extraction, ou à défaut de proposer un paramétrage peu chronophage pour le documentaliste. L'extraction automatique du texte pertinent permet par la suite d'indexer précisément chaque article.
- Croisement d'informations : à partir d'une information donnée, rechercher la même information dans d'autres sources, notamment des sources inconnues sur le web, i.e. des sources ne faisant pas partie de la base de sources veillées chaque jour.
- Découverte de sources : rechercher de nouvelles sources d'informations sur le web. Cette fonctionnalité est liée au croisement d'informations. L'objectif est de proposer de nouvelles sources d'informations potentielles aux documentalistes lorsque des informations pertinentes sont trouvées dans des sources inconnues.

Les parties II et III décrivent respectivement un système de classification et un système de recherche d'informations, basés sur une approche générique et un apprentissage non supervisé. Il est important de souligner l'écart qui existe entre le cas d'utilisation réel au sein de l'entreprise Actualis, et les architectures décrites précédemment dans ce manuscrit. Le nouveau processus de recherche et de qualification automatique de l'information doit pouvoir s'intégrer aux processus métiers de l'entreprise en étant le moins intrusif possible. De fait, les objectifs de l'application dans ce contexte industriel sont :

- Un gain de temps lors de la qualification des articles par les documentalistes, qui conduit à une performance accrue de la phase de rédaction de la revue FirstEco Pro'fil
- Apporter une plus-value en terme de qualification de l'information, i.e. une qualification plus exhaustive comparativement aux annotations manuelles seules
- Une simplification de la phase de recherche passive d'informations, qui répond à la problématique de croisement et de découverte d'informations évoquée en introduction de ce manuscrit
- La mise à disposition d'une architecture modulable et extensible, qui doit permettre de proposer des services plus performants à l'avenir, à destination des documentalistes

7.3/ ARCHITECTURE

Cette section décrit comment l'architecture de recherche et de qualification automatique de l'information a été mise en place au sein de l'entreprise Actualis SARL. La mise en place des deux architectures décrites précédemment dans ce manuscrit (SHMC et SEMXDM) est définie dans les deux sous-sections suivantes. La dernière sous-section décrit le déploiement de l'architecture complète.

7.3.1/ APPRENTISSAGE SUPERVISÉ D'UN MODÈLE DE CLASSIFICATION DE NOUVELLES ÉCONOMIQUES

Deux cas d'utilisation de l'architecture SHMC au sein de l'entreprise ont été définis :

- L'apprentissage d'un modèle de classification de nouvelles économiques de façon supervisée, à partir de nouvelles économiques sémantiquement enrichies manuellement par les documentalistes. Les données initiales employées pour cette approche sont issues de la revue FirstEco Pro'fil.
- L'extension des différents vocabulaires d'indexation, par un apprentissage non supervisé de nouveaux concepts et de nouvelles relations hiérarchiques entre ces concepts. Les vocabulaires d'indexation doivent en effet être mis à jour occasionnellement, afin de refléter l'évolution des différentes facettes de description au cours du temps. Ces modifications impactent le processus de classification.

Le premier cas d'utilisation est la fonctionnalité principale attendue. Le second cas est en cours d'étude au sein de l'entreprise, mais n'est pas prioritaire et n'est pas actuellement intégré au prototype de l'outil de veille.

L'approche et le système existant sont basés sur les technologies du web sémantique, ce qui permet de rapprocher plus simplement le modèle d'apprentissage établit dans les chapitres précédents et le modèle d'indexation de l'entreprise, grâce à l'unification des formats d'un point de vue de l'implémentation.

Comme décrit dans la section précédente, les vocabulaires contrôlés définissent un ensemble de concepts de haut niveau, structurés entre eux par des relations de subsomption. Dans la facette des thèmes, les relations *hasSubTopic* et *isSubTopicOf* représentent les relations de subsomption entre les thèmes, tandis que les relations *hasSubSector* et *isSubSectorOf* représentent les relations de subsomption dans la facette des secteurs. Les différents vocabulaires contrôlés sont donc considérés comme une hiérarchie initiale de classes pour l'apprentissage du modèle. Dans le prototype, on ne considère donc pas la mise en place de la phase de hiérarchisation du processus SHMC. Les vocabulaires contrôlés étant structurés de façon hiérarchique, l'apprentissage automatique des labels et de la hiérarchie ne sont pas nécessaires.

Les données d'apprentissage sont des articles économiques pré-étiquetés provenant de la revue FirstEco Pro'fil. Ces articles sont une synthèse d'articles de presse rédigés par les documentalistes (voir annexe C). Afin de permettre une classification automatique des articles originaux utilisés par les documentalistes pour rédiger la synthèse, il semble plus pertinent d'utiliser directement les articles extraits des sources d'information comme données d'apprentissage. La construction d'un jeu de données pré-étiqueté de ce type est également un des objectifs de la mise en place du nouvel outil de veille. En effet, la synthèse rédigée par les documentalistes est généralement de taille réduite, et contient

donc moins de vocabulaire contextuel comparativement à l'article original.

L'application du processus d'apprentissage est similaire à la description des chapitres 3 et 4. L'architecture technologique est présentée dans la figure 7.6 dans la section suivante. Lors de la phase d'indexation, les données d'apprentissage sont importées dans un serveur Solr.

La figure 7.4 présente un exemple d'article de synthèse en entrée du processus. L'article est un document XML au format Solr, défini par un ensemble de champs (content, secteur, uri, département, id). Le champ *uri* contient l'étiquetage manuellement effectué par les documentalistes. Les étiquettes correspondent à des concepts des vocabulaires d'indexation des différentes facettes de l'ontologie. Par exemple, l'*uri* <http://www.firsteco.fr/2013/12/ontologie#sector-1623> correspond à un secteur d'activité. Le contenu de l'article permet la création d'un vecteur de termes pour chaque article.

```
<doc>
  <field name="content">N°1 de l'escalade indoor en France, le lyonnais Climb Up investit
  <field name="typeArticle">ArticleFlash</field>
  <field name="title"/>
  <field name="secteur">SECTOR_REVÊTEMENT_MURAL_SECTOR_SPORT </field>
  <field name="uri">http://www.firsteco.fr/2013/12/ontologie#tailleEntreprise-TPE</field>
  <field name="uri">http://www.firsteco.fr/2013/12/ontologie#sector-1623</field>
  <field name="uri">http://www.firsteco.fr/2013/12/ontologie#city-33063</field>
  <field name="uri">http://www.firsteco.fr/2013/12/ontologie#topic-41</field>
  <field name="uri">http://www.firsteco.fr/2013/12/ontologie#sector-121</field>
  <field name="departement">33</field>
  <field name="departement">69</field>
  <field name="id">10004</field>
</doc>
```

FIGURE 7.4 – Exemple d'article étiqueté

Le processus d'apprentissage est ensuite effectué tel qu'il est présenté dans les chapitres 3 et 4.

Dans le cas d'utilisation de l'entreprise, un besoin de contrôle sur la qualité et la quantité des étiquetages proposés par le modèle prédictif a été identifié. Par conséquent, plutôt que générer un unique modèle prédictif correspondant à des seuils prédéfinis, plusieurs modèles sont générés à partir des mêmes données. Les différents modèles sont générés en faisant varier les seuils déterminants de création du modèle, notamment les seuils de création des règles de classification. L'algorithme de création des règles définit deux seuils α et β pour la création des règles (voir section 4.1.1).

La hiérarchie de labels étant fixe (vocabulaires d'indexation basés sur les facettes), les différents modèles prédictifs sont distingués uniquement par les règles de classification. L'intégration des différents modèles est détaillée dans la section déploiement ci-après.

7.3.2/ APPLICATION DU PROCESSUS DE RECHERCHE POUR LA VEILLE STRATÉGIQUE

Nous avons identifié 3 phases distinctes du processus de collecte d'informations dans l'entreprise, qui doivent être intégrées dans le nouvel outil (figure 7.5) :

- La phase catalogue consiste en la recherche de nouveaux liens menant vers des articles à partir des pages de type catalogue, i.e. les pages qui réfèrent les

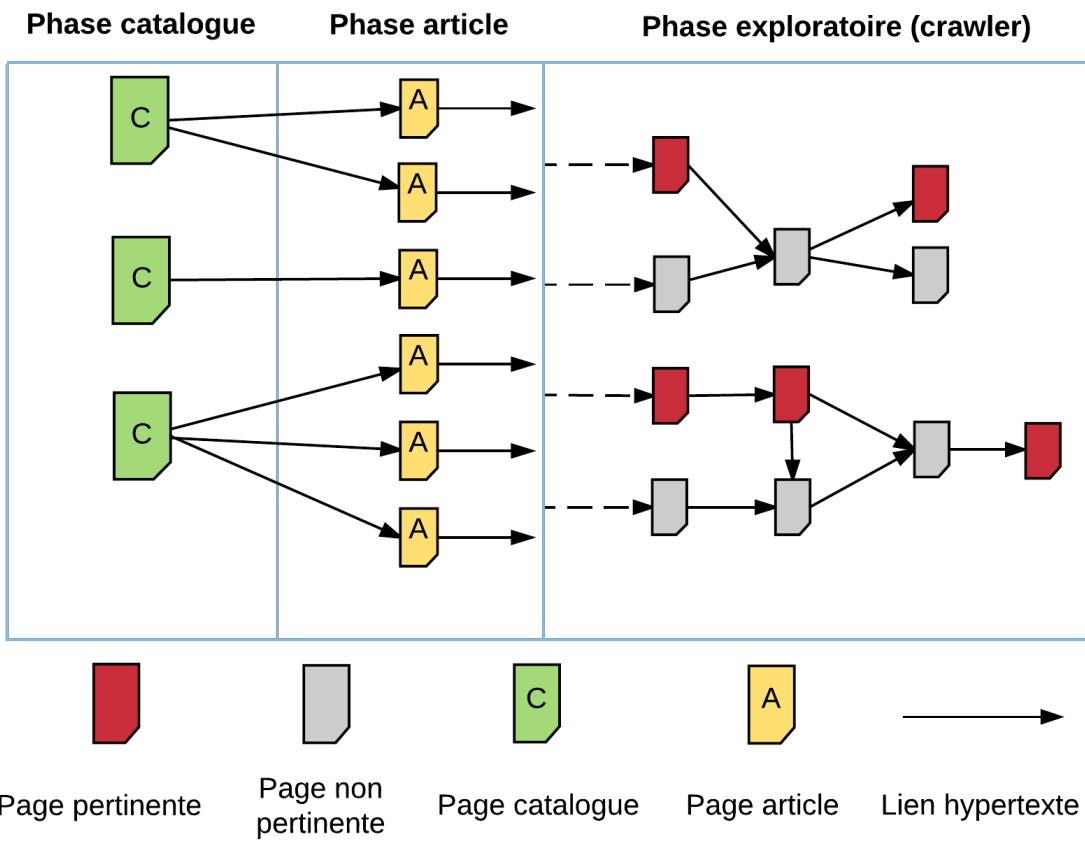


FIGURE 7.5 – Phases de la collecte d’informations

nouveaux articles, et qui les introduisent par titre et éventuellement un chapô. Les liens qui pointent vers de nouveaux articles sont suivis dans la phase d’article. L’ensemble des liens visités est mis à jour dans une base de données pour valider les liens à suivre. Les pages catalogues sont fixes et ne nécessitent pas de processus d’extraction d’informations (seuls les liens sortants sont extraits de ces pages). Lors de la phase catalogue, une analyse manuelle peut souvent s’arrêter au chapô pour retirer l’essentiel de l’information, i.e. il n’est pas toujours nécessaire d’accéder aux pages articles individuelles. En revanche, un processus d’indexation peut tirer bénéfice d’un article complet contenant plus de contexte, pour la construction d’un modèle prédictif. Il est donc nécessaire d’identifier les liens menant vers des pages articles.

- La phase article consiste à extraire l’information des pages articles, pour évaluer leur pertinence et stocker le résultat de l’extraction dans une base de données (voir architecture technique ci-après). Le résultat de l’extraction d’information est un vecteur de termes, basé sur les portions pertinentes de texte extraites de l’article. Les problématiques associées à l’extraction d’information à partir de contenu web, abordées dans le chapitre 5, sont ici à prendre en compte, notamment la discrimination des portions non pertinentes de la page. La chaîne de traitement syntaxique développée dans les deux architectures est également employée lors de la phase article.
- La phase exploratoire concerne la mise en application de l’architecture de recherche d’informations, développée dans le chapitre 6. Cette phase répond aux objectifs de croisement et de découverte d’information. Les requêtes de recherche

sont émises par les documentalistes. L'architecture SEMXDM permet dans ce cadre le croisement d'informations par un crawler sans supervision. Les résultats de la tâche de croisement (documents potentiellement pertinents) sont triés et stockés, afin d'être revus ultérieurement par le documentaliste. La recherche peut être passive et réitérée dans le temps de façon automatique.

Chaque phase est mise en place dans le nouvel outil. La phase catalogue et la phase article font partie du processus actuel de production des documentalistes (voir annexe C). Le nouvel outil doit cependant améliorer les limites des outils utilisés actuellement dans ces deux phases décrites dans la section précédente.

Une des problématiques liées à l'exploitation des pages catalogue et des pages articles est l'absence de norme dans la présentation de l'information à partir des différentes sources d'information. Les deux types de page sont en effet très hétérogènes, et des heuristiques doivent souvent être employées pour traiter certains cas particuliers. La structure des pages est affectée par l'outil de visualisation de la page, qui est classiquement un ordinateur ou un appareil mobile. Cependant, dans le cas d'un outil de veille automatisé, il est commun d'extraire simplement la structure HTML de la page. Cette approche n'est aujourd'hui pas suffisante pour permettre l'extraction du contenu de la plupart des sites web. En effet, la structuration des sites est de plus en plus dépendante d'exécution de code dynamique (essentiellement basé sur JavaScript), qui n'est prise en compte que lors d'une navigation réelle sur le site. Un robot d'indexation classique, par exemple, n'exécute pas de code dynamique lors du fetching (téléchargement) d'une page. Une solution d'automatisation de cette tâche est alors de baser le fetching des pages sur des technologies telles que le framework SeleniumHQ⁵, qui émule le comportement d'un navigateur web classique, et permet de récupérer une version plus représentative du contenu attendu des pages.

Lors de la phase article, l'ancien outil permettait le paramétrage de l'extraction automatique de texte (filtres). Cette fonctionnalité est cependant perfectible et coûteuse en temps pour les documentalistes (le nombre important de sources ne leur permet pas de s'attarder sur le paramétrage de toutes les sources).

La sous-section suivante décrit l'architecture technique de l'application.

7.3.3/ DÉPLOIEMENT

Le déploiement de l'application comprend un client mis à disposition des documentalistes, et un système d'information qui regroupe l'ensemble des technologies requises pour la mise en place par l'architecture de recherche d'information, ainsi que la construction et l'application du modèle prédictif.

La figure 7.6 présente l'architecture technique de l'application. Pour distribuer la quantité importante de pages à mettre à jour de façon régulière, l'étape de fetching⁶ des pages est déléguée à un ensemble de machines. Ces machines sont également responsables du fetching dans les deux autres phases (article et exploratoire). Le déploiement de ces machines et la répartition de la charge consiste en la mise en place d'un hub de conteneurs légers *Docker*, qui intègrent le framework SeleniumHQ.

Une plateforme distribuée, dite de stockage et d'analyse, comprend les outils néces-

5. <http://www.seleniumhq.org/>

6. Téléchargement automatisé

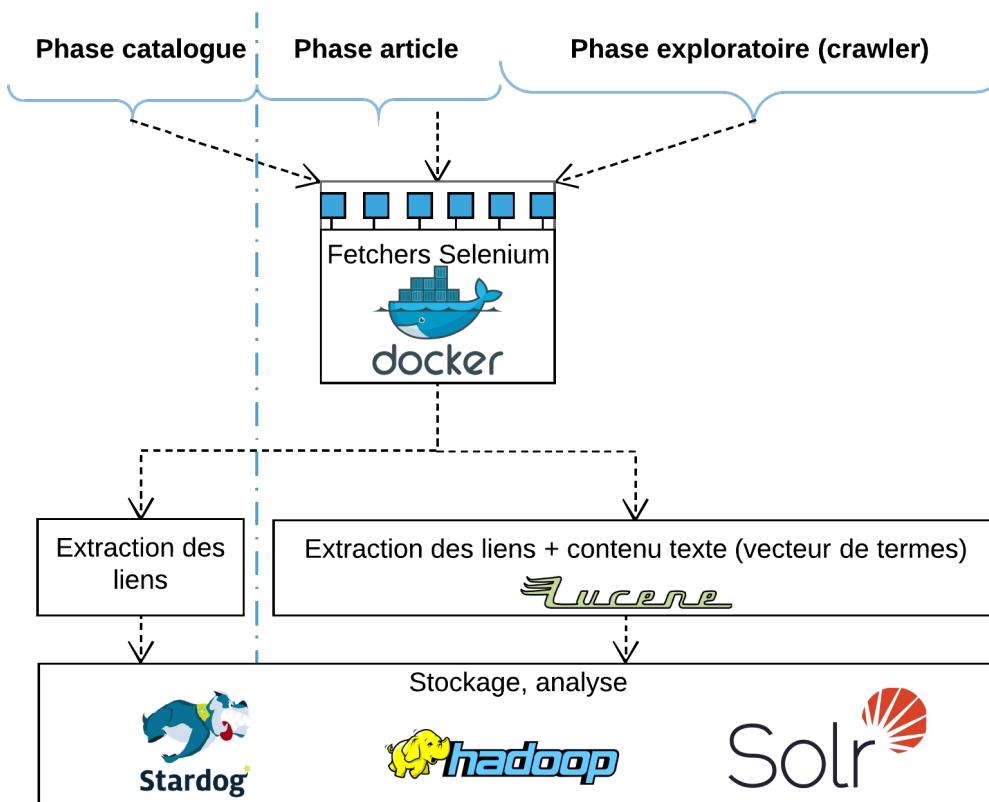


FIGURE 7.6 – Architecture du nouvel outil "FirstWatch"

saires à la mise en place des deux architectures SHMC et SEMXDM. L'environnement de traitement massif des données Hadoop permet la construction du modèle prédictif et l'application de la phase exploratoire, i.e. de l'architecture distribuée de recherche d'information.

La construction du modèle prédictif fait l'objet d'une supervision manuelle. Une fois étiquetées, les données récoltées durant le cycle de vie de l'application sont importées dans un serveur Solr, conformément au format défini dans la section précédente. L'ingestion de nouvelles données d'apprentissage durant le cycle de vie de l'application permettra de faire évoluer le modèle prédictif de façon périodique, pour refléter l'évolution des vocabulaires d'indexation et des articles économiques.

Les différents modèles prédictifs correspondant à différents seuils sont intégrés dans le triple store Stardog. Pour distinguer les règles issues des différents modèles, l'intégration des différentes règles de classification dans le triple store est définie par l'ajout d'une annotation⁷ correspondant aux seuils utilisés lors de la création des règles SWRL.

L'annotation permet de distinguer les règles en fonction de la valeur de seuil employée lors de leur création. De cette façon il est possible de sélectionner une valeur de seuil au moment de la classification de façon dynamique. Cette propriété est importante pour permettre de refléter le degré de confiance accordé à chaque règle. Les règles basées sur une valeur de seuil restrictive (élevée) sont plus sûres, mais moins nombreuses. Il est ainsi possible pour les documentalistes d'influer directement sur la qualité (précision)

7. <https://www.w3.org/TR/owl-ref>

et la quantité (rappel) d'étiquetages réalisés lors de l'application des règles (i.e. lors de l'inférence).

Afin d'optimiser la performance du triple store lors de l'inférence, le processus de sélection des règles est étendu pour appliquer une sélection en fonction des valeurs de seuils désirées. Une requête SPARQL permet la sélection et l'activation/la désactivation des règles strictement nécessaires à l'inférence :

Listing 7.1 – Sélection de règle par seuil

```

1 select distinct ?rule ?seuil
2 where {
3     ?rule <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
4     <http://www.w3.org/2003/11/swrl#Imp> .
5     ?rule Ontologie:seuil ?seuil .
6         FILTER (?seuil > 40)
7 }
```

Listing 7.2 – Activation de règles par seuil

```

1 DELETE { ?rule <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
2 <http://www.w3.org/2003/11/swrl#ImpDeactivated> }
3 INSERT { ?rule <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
4 <http://www.w3.org/2003/11/swrl#Imp> }
5 WHERE
6 { { ?rule <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
7 <http://www.w3.org/2003/11/swrl#ImpDeactivated> .
8     ?rule ontologie:seuil ?seuil
9     FILTER(?seuil > 70)} }
```

7.3.4/ APERÇU DU PROTOTYPE

Le nouvel outil sera graduellement pris en main par une partie de l'équipe de rédaction dans l'entreprise à partir de septembre 2017, afin d'évaluer les apports vis-à-vis de l'ancien outil. Les figures 7.7 à 7.12 montrent le prototype de l'application côté client, i.e. l'interface de l'outil de veille mise à disposition des documentalistes.

La figure 7.8 montre les options de paramétrage d'une source d'information. L'interface permet de configurer les métadonnées associées à une source, ainsi que l'extraction de contenu à partir de cette source pour la phase catalogue et la phase article. Le client permet de prévisualiser le résultat de l'extraction de la phase article pour chaque source.

Les signets sont des catégories permettant le regroupement de sources d'information, définies par les documentalistes de façon libre. Les signets peuvent par exemple permettre de regrouper les articles en fonction d'étiquetages réalisés par le modèle prédictif, une requête plein-texte, un ensemble de sources prédéfinies, ou un autre type de métadonnées (figures 7.9 à 7.11).

L'interface de gestion des sources permet la mise à jour des sources d'information spécifiques à un/une documentaliste. Chaque source correspond à une région géographique, et à un type de source (flux RSS, blog, site de presse, réseau social...). Les sources peuvent être triées singulièrement en fonction de types définis par le documentaliste.

L'interface de requête de recherche d'information est en cours de développement.

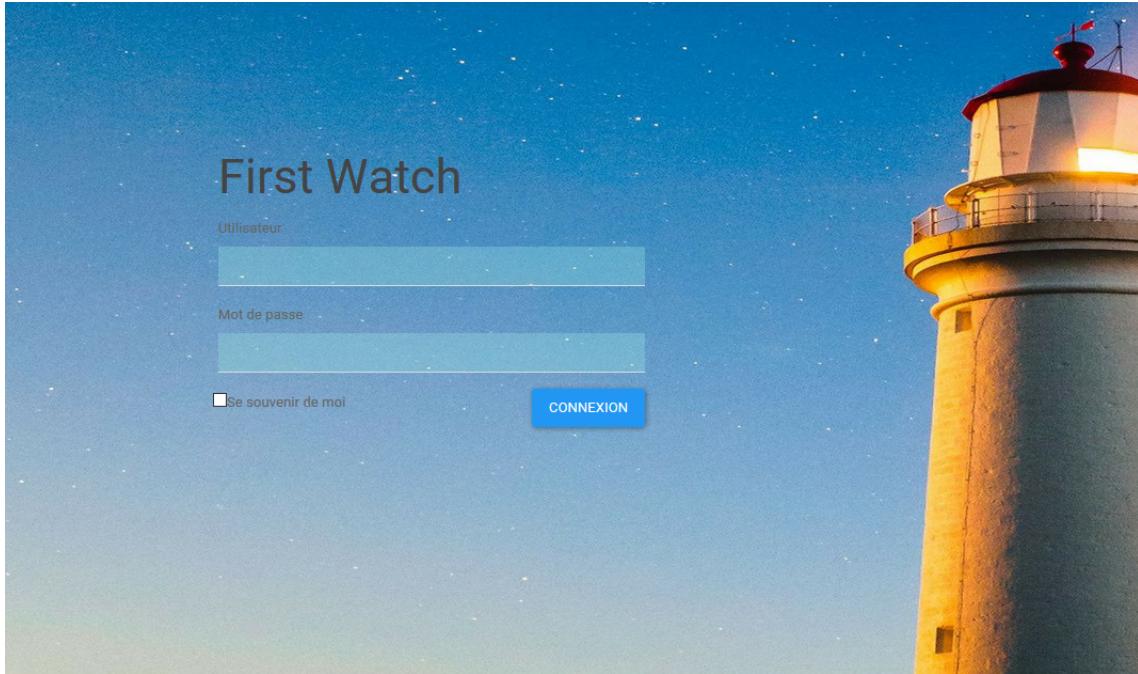


FIGURE 7.7 – Page d’authentification de l’application

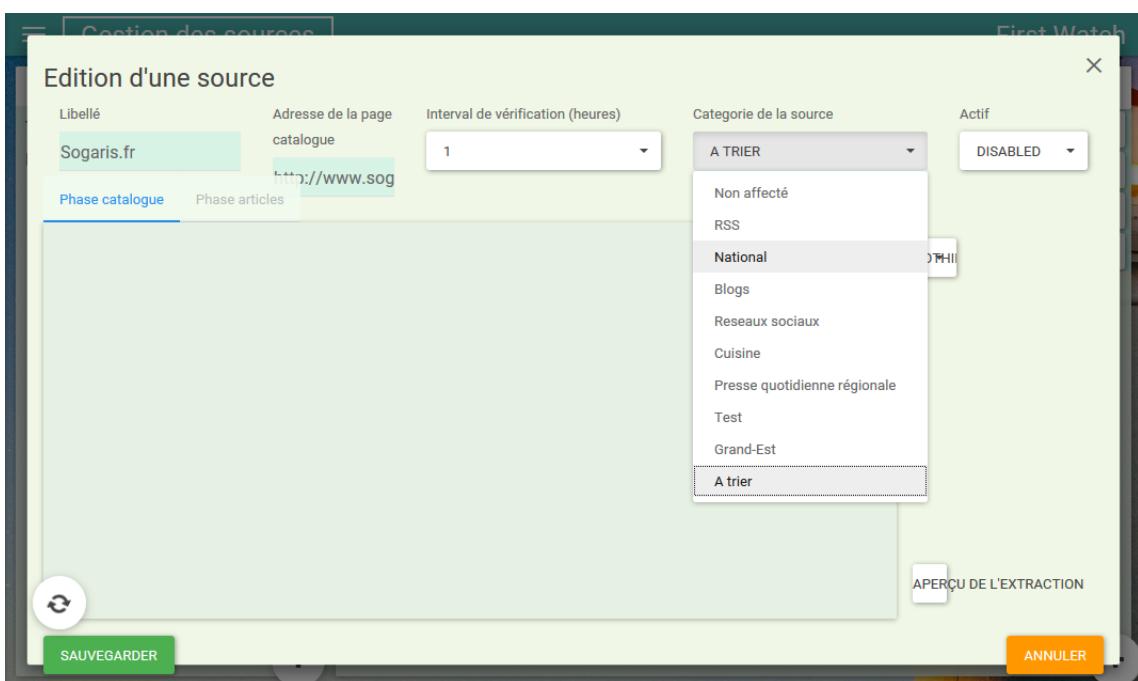


FIGURE 7.8 – Paramétrage d’une source d’information

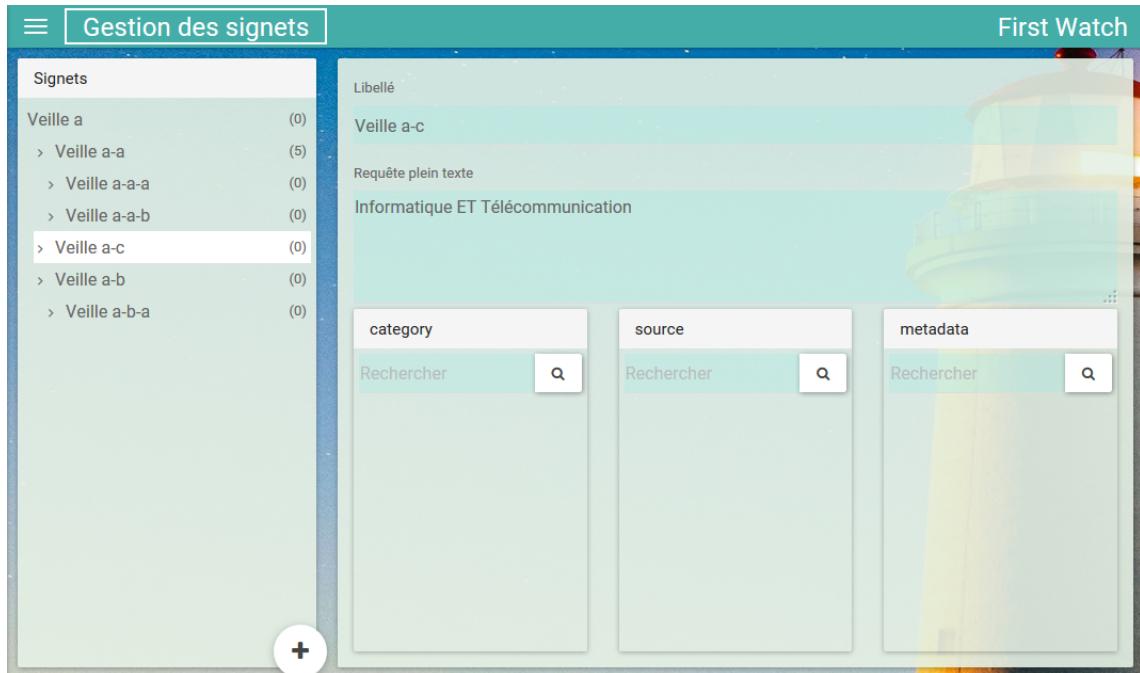


FIGURE 7.9 – Édition de la liste des signets

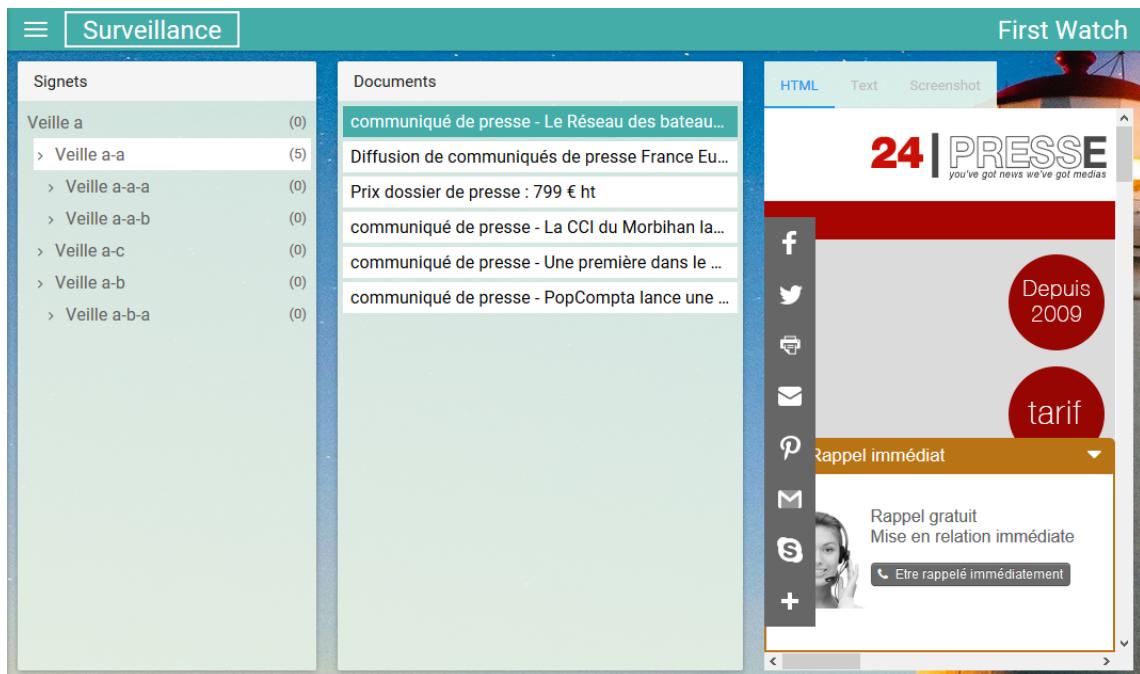


FIGURE 7.10 – Signets : affichage de la page originale

The screenshot shows the 'Surveillance' tab of the First Watch application. On the left, there's a sidebar titled 'Signets' containing a tree view of monitoring categories. The 'Veille a' category has several sub-categories: 'Veille a-a' (5 items), 'Veille a-a-a' (0 items), 'Veille a-a-b' (0 items), 'Veille a-c' (0 items), 'Veille a-b' (0 items), and 'Veille a-b-a' (0 items). The main panel is titled 'Documents' and lists several press releases. One item is highlighted: 'communiqué de presse - Le Réseau des bateau...'. Below it are other items like 'Diffusion de communiqués de presse France Eu...', 'Prix dossier de presse : 799 € ht', etc. To the right, a preview window titled 'First Watch' shows the content of the selected document: a quote about boat permits and training requirements.

FIGURE 7.11 – Signets : affichage du texte extrait

The screenshot shows the 'Gestion des sources' (Source Management) tab of the First Watch application. On the left, there's a sidebar titled 'Catégories' listing various news sources. The 'Toutes les sources' category has 4 items. Under 'RSS', there are 'National' (806 items), 'Blogs' (10 items), 'Réseaux sociaux' (0 items), 'Cuisine' (9 items), 'Presse quotidienne régionale' (0 items), 'Test' (2 items), 'Grand-Est' (1 item), and 'A trier' (4 items). The main panel is titled 'Sources' and displays a table with four rows. The first three rows have URLs starting with 'http://www.21centralepartners.' and the fourth row has 'http://www.sogaris.fr'. Each row includes columns for the source name, URL, status (1, disabled, 2), and several action icons (edit, delete, etc.).

FIGURE 7.12 – Gestionnaire de sources d'information

IV

DISCUSSION ET CONCLUSION

8

CONCLUSION

8.1/ DISCUSSION, CONTRIBUTIONS

Ce manuscrit a décrit une nouvelle approche pour la recherche et la classification automatique de données web. Cette approche est axée sur deux architectures qui tentent de répondre à différents verrous scientifiques (section 1.3.5), à l'intersection de quatre domaines : le traitement massif des données, l'apprentissage machine, la recherche d'informations, et le web sémantique. Une étude approfondie de ces domaines a montré la pertinence de l'approche, et une architecture fonctionnelle a été développée afin de démontrer sa faisabilité. Ces travaux reposent en outre sur des publications nationales et internationales Hassan et al. (2015), Peixoto et al. (2015b) Peixoto et al. (2015c) Peixoto et al. (2016b) Peixoto et al. (2016c) Hassan et al. (2017a) Hassan et al. (2017b). L'approche a par la suite été mise en place dans un contexte industriel, afin de répondre à des problématiques métier précises.

La première partie de ce document décrit une architecture de classification hiérarchique multi-étiquette nommée SHMC. Cette architecture répond à deux verrous scientifiques décrits en introduction : d'une part l'apprentissage non-supervisé d'un modèle prédictif décrit par une ontologie à partir de larges volumes de données, et d'autre part l'exploitation du raisonnement logique pour la classification de textes en langage naturel dans un contexte Big Data. Pour répondre à ces verrous, l'approche s'appuie notamment sur les technologies web sémantique et du big data. Les expérimentations spécifiques à cette architecture présentée dans cette partie sont focalisées sur trois points :

- La démonstration de la faisabilité de l'approche (i.e. preuve de concept)
- La performance de cette approche comparativement à l'état de l'art en matière de classification hiérarchique multi-étiquette en contexte supervisé
- La performance de l'approche pour l'apprentissage non supervisé d'un modèle prédictif à partir de larges volumes de données

Les résultats de ces expérimentations ont montré que l'approche a été implémentée avec succès et répond à des problématiques identifiées dans l'introduction de ce manuscrit. En particulier, l'architecture SHMC permet de palier les problématiques de passage à l'échelle exposées dans les travaux antérieurs de Werner et al. (2014), qui montrait les limites d'une approche de classification utilisant un modèle prédictif décrit uniquement dans via la TBox de l'ontologie.

La seconde partie a décrit une architecture de recherche d'information sur le web basée un robot d'indexation ciblé, nommée SEMXDM. Cette partie s'intéresse aux deux derniers verrous scientifiques identifiés en introduction, i.e. l'intégration d'un modèle prédictif

pour améliorer la découverte de sources d'information, et l'adaptation de ce modèle prédictif dans un contexte web. Le modèle prédictif est utilisé pour classer automatiquement de nouvelles informations extraites du web. Ces mêmes informations sont utilisées pour enrichir et mettre à jour en temps réel le modèle prédictif, afin d'améliorer de façon incrémentale la recherche de nouvelles informations. Les contributions de cette architecture ont été soulignées. Premièrement, la capacité d'adaptation en temps réel de l'architecture à de nouvelles données non structurées est établie, bien que des limitations techniques aient été identifiées. L'expérience acquise par le robot d'indexation lors de la phase de recherche est prise en compte dans l'attribution de la priorité des liens de la frontière, ce qui permet d'optimiser la temps de recherche d'information de l'approche. Comparativement à une approche standard, notre approche permet une optimisation du temps de recherche qui peut être critique dans un cadre industriel, de par la nature éphémère de l'information. Une étude plus approfondie de l'approche en contexte industriel doit cependant être menée.

Les évaluations ont montré une performance positive des deux architectures, et soulignent l'intérêt de l'approche. Les limites des deux architectures ont été exposées dans les parties II et III. Des pistes de réflexion vis-à-vis de ces limitations ont été identifiées et seront introduites dans les perspectives de ces travaux (section suivante).

Enfin, la dernière partie de ce document s'intéresse à l'application de l'approche et son rapprochement avec un domaine métier, la veille stratégique, et à sa mise en place dans un cadre industriel concret au sein de l'entreprise Actualis SARL. L'objectif principal de l'application du processus dans ce contexte est de déterminer automatiquement la valeur d'informations non structurées, par une classification précise de l'information selon une base de connaissances spécifique au domaine. Cette partie s'est notamment intéressée à l'intégration d'une base de connaissances métier utilisée par des experts du domaine dans la construction du modèle prédictif généré par l'architecture SHMC. Un nouvel outil de veille basé sur les deux architectures, First Watch, est en cours de développement au sein de l'entreprise.

La section suivante reprend les limites fondamentales et techniques de l'approche afin d'en souligner les points critiques, et suggérer à la fois des solutions pratiques nécessaires son amélioration, mais aussi un regard extérieur sur l'évolution du contexte scientifique dans lequel ces travaux s'inscrivent.

8.2/ LIMITATIONS ET TRAVAUX FUTURS

Les problématiques traitées dans ces travaux concernent des domaines qui subissent un changement profond cette dernière décennie. Le développement croissant des approches basées sur l'apprentissage profond (deep learning) a bouleversé le domaine de l'apprentissage automatique, grâce aux évolutions technologiques, à la prolifération de jeux de données de taille conséquente, et à l'évolution des architectures d'apprentissage profond. Ces approches ont été appliquées avec succès pour des tâches variées, et en particulier pour la classification. Bien que la majorité des applications portent sur des données graphiques (images, vidéos), notamment par des approches basées sur les réseaux de convolution, l'application de telles méthodes à d'autres types de données telles que le texte en langage naturel à également fait l'objet de travaux récents encourageants, sans nécessairement reposer sur une base symbolique ou le domaine du traitement au-

tomatique du langage Zhang et al. (2015).

La nature hiérarchique de l'apprentissage de caractéristiques (features) de haut niveau semble en effet permettre de percevoir la nature composite des données, contrairement à d'autres approches en apprentissage automatique. L'apprentissage profond semble ainsi accepté comme la meilleure solution existante à ce jour pour la création de systèmes intelligents basés pour la perception du monde basé sur l'exemple, i.e via l'apprentissage d'un modèle prédictif.

Il apparaît clair que les approches symboliques, notamment celles issues du web sémantique, conservent des propriétés indispensables à l'évolution des systèmes dits intelligents et de l'intelligence artificielle, notamment pour la prolifération de nouvelles données structurées sémantiquement, et à l'interconnexion de ces données. Ces propriétés sont utilisées tous les jours à bon escient, notamment dans le cadre du Linked Data. Dans sa dimension descriptive, il semble que le web sémantique représente un socle technique indispensable à la création de systèmes intelligents qui surpassent le cadre d'une tâche spécifique.

Pour que les systèmes intelligents évoluent au delà du cadre de la perception, les notions de raisonnement, d'anticipation, de planification et de mémoire associative Sukhbaatar et al. (2015) semblent essentielles. Il est possible que les approches symboliques contribuent à l'évolution de l'intelligence artificielle en se focalisant sur ces dimensions. La disponibilité de connaissances établies sur le monde, et la possibilité d'inférer sur ces connaissances sont des atouts indéniables qui pourraient être mis à profit dans la création de systèmes intelligents hybrides ou ensemblistes.

Dans le cadre de la classification, ces propriétés pourraient alors être associées à l'apprentissage profond, afin de compléter ces approches. Le passage à l'échelle du raisonnement logique et l'introduction de mécanismes d'inférence floues, qui font l'objet de nombreux travaux récents dans le web sémantique, sont des pistes intéressantes de développement pour des systèmes intelligents qui reposent sur le web sémantique. Ces évolutions sont encourageantes pour la mise en place de tels systèmes.

Les deux architectures présentées dans ce manuscrit sont expérimentales, et volontairement simplistes d'un point de vue fondamental et conceptuel. Ces travaux ont tenté de démontrer la pertinence de la complémentarité des approches symboliques et de l'apprentissage automatique, dans une période d'innovation majeure dans le domaine de l'intelligence artificielle, où l'écosystème des technologies évolue très rapidement. L'intégration de connaissances métier dans le processus d'apprentissage d'un modèle prédictif est en outre un atout majeur pour la création de systèmes intelligents appliqués à un contexte industriel.

Dans le cadre de ces travaux, les évolutions technologiques impliquent des possibilités d'amélioration de performance des approches décrites dans ce manuscrit. De nouvelles technologies se sont développées autour des environnements de calcul distribués, sur lesquels se basent les architectures développées dans les chapitres 3, 4 et 6. Ces technologies portent à la fois sur le traitement par lot, dont l'architecture SHMC fait l'objet, et le traitement par flux dont l'architecture SEMXDM fait l'objet.

Dans le cas de la première architecture, le développement du calcul distribué pour le raisonnement logique, et la disponibilité de frameworks plus performants de traitement par lot désormais matures, tels que Apache Spark, permettraient une amélioration des performances de l'application en terme de temps de calcul. En ce qui concerne le rai-

sonnement logique, le potentiel de l'emploi de règles de Horn pour l'application de règles d'association d'expressivité faible a été validé par nos expérimentations. Il serait intéressant d'étudier les extensions possibles du modèle prédictif, en terme d'expressivité, selon l'évolution des technologies de raisonnement distribué.

Les résultats encourageants de l'apprentissage d'un modèle prédictif en contexte non supervisé pour la tâche de classification, présentés dans les chapitres 3 et 4, posent la question de l'exploitabilité d'un tel modèle dans un processus de production concret. L'interconnexion d'un tel modèle à des données structurées telles que les données issues du Linked Data semble essentielle à ce niveau. Dans le contexte industriel étudié, la pertinence d'une approche non supervisée pour l'extension des vocabulaires contrôlés de l'entreprise a été soulignée.

La modularité des différents éléments de l'architecture SHMC permet une extension des algorithmes employées au sein de chaque phase du processus d'apprentissage, notamment pour l'apprentissage de règles de classification. L'intégration d'algorithmes d'apprentissage de règles d'association de l'état de l'art (Apriori, Eclat, FP-growth...), ou d'autres algorithmes de classification est étudiée pour l'évolution du prototype. Les chapitres 4 et 7 ont en outre suggéré la pertinence de l'association de différents modèles prédictifs au sein d'une même base de connaissance. Des méthodes de décision ensemblistes pourraient alors être appliquées pour améliorer la performance globale de l'application pour la tâche de classification, tout en conservant les avantages issus de l'approche symbolique, en particulier une compréhension facilitée du processus de classification et le lien établit avec les connaissances métier.

Dans la seconde architecture, le gain de performance en terme de recherche d'information indiqué dans le chapitre 6 devrait être le sujet d'expérimentations plus poussées qui n'ont pas été menées faute de temps. Aussi, ce gain de performance, bien que pertinent, demande la mise en place d'une architecture matérielle et logicielle lourde et difficile à maintenir dans un environnement de production. Ces faiblesses peuvent être amoindries par une mise à jour technologique de l'application, ainsi que du paradigme de maintenance incrémentale du modèle prédictif. Des frameworks de traitement par flux plus récents permettent une mise en place simplifiée de ce type de processus.

Les contraintes d'application de méthodes d'extraction d'information dans un contexte web sont un frein important à la mise en place d'un système de veille complètement automatisé. Pour pallier les limitations de l'application dans ce domaine, il semble nécessaire que des travaux soient menés au niveau de l'apprentissage automatique pour l'extraction d'informations à partir de contenus non structurés et semi-structurés. L'analyse de la redondance et de la pertinence de l'information sur le web, relativement à un nom de domaine, est probablement une piste de développement intéressante des approches d'apprentissage automatique. Des approches purement graphiques, i.e. qui ne sont pas basées sur la structuration syntaxique d'un document web sont éventuellement à envisager afin de répliquer un comportement d'extraction d'information qui correspond au sens commun.

BIBLIOGRAPHIE

- Aggarwal, C. C., Al-Garawi, F., et Yu, P. S. (2001). **Intelligent crawling on the world wide web with arbitrary predicates**. In *Proceedings of the 10th international conference on World Wide Web*, pages 96–105. ACM.
- Aggarwal, C. C., Han, J., Wang, J., et Yu, P. S. (2006). **A framework for on-demand classification of evolving data streams**. *IEEE Transactions on Knowledge and Data Engineering*, 18(5) :577–589.
- Agichtein, E., et Gravano, L. (2000). **Snowball : Extracting relations from large plain-text collections**. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM.
- Aparicio, R., et Acuna, E. (2013). **Using Ontologies To Improve Document Classification With Transductive Support Vector Machines**. *International Journal*, 3(3) :1–15.
- Asuncion, A., Welling, M., Smyth, P., et Teh, Y. W. (2009). **On smoothing and inference for topic models**. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press.
- Baader, F. (2003). **The description logic handbook : Theory, implementation and applications**. Cambridge university press.
- Baker, C. F., Fillmore, C. J., et Lowe, J. B. (1998). **The berkeley framenet project**. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.
- Bao, S., Li, R., Yu, Y., et Cao, Y. (2008). **Competitor mining with the web**. *IEEE Transactions on Knowledge and Data Engineering*, 20(10) :1297–1310.
- Batsakis, S., Petrakis, E. G., et Milios, E. (2009). **Improving the performance of focused web crawlers**. *Data & Knowledge Engineering*, 68(10) :1001–1013.
- Ben-David, D., Domany, T., et Tarem, A. (2010). **Enterprise data classification using semantic web technologies**. In *The Semantic Web-ISWC 2010*, ISWC’10, pages 66–81, Berlin, Heidelberg. Springer-Verlag.
- Bergmark, D., Lagoze, C., et Sbitiyakov, A. (2002). **Focused crawls, tunneling, and digital libraries**. In *International Conference on Theory and Practice of Digital Libraries*, pages 91–106. Springer.
- Bi, W., et Kwok, J. (2011). **Multi-label classification on tree-and DAG-structured hierarchies**. *Yeast*, pages 1–8.
- Blei, D. M., Ng, A. Y., et Jordan, M. I. (2003). **Latent dirichlet allocation**. *Journal of machine Learning research*, 3(Jan) :993–1022.

- Bobadilla, J., Ortega, F., Hernando, A., et Gutiérrez, A. (2013). **Recommender systems survey**. *Knowledge-based systems*, 46 :109–132.
- Brank, J., Grobelnik, M., et Mladenic, D. (2005). **A survey of ontology evaluation techniques**. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, pages 166–170.
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., et Mitchell, T. M. (2010). **Toward an architecture for never-ending language learning**. In *AAAI*, volume 5, page 3.
- Cerri, R., Barros, R. C., et De Carvalho, A. C. (2014). **Hierarchical multi-label classification using local neural networks**. *Journal of Computer and System Sciences*, 80(1) :39–56.
- Chaabani, M., Mezghiche, M., et Strecker, M. (2009). **Formalisation de la logique de description alc dans l'assistant de preuve coq**. *Proc. 3es Journées francophones sur les ontologies*, pages 139–147.
- Chakrabarti, S., Punera, K., et Subramanyam, M. (2002). **Accelerated focused crawling through online relevance feedback**. In *Proceedings of the 11th international conference on World Wide Web*, pages 148–159. ACM.
- Chakrabarti, S., Van den Berg, M., et Dom, B. (1999). **Focused crawling : a new approach to topic-specific web resource discovery**. *Computer Networks*, 31(11) :1623–1640.
- Chang, C.-H., Kayed, M., Girgis, M. R., et Shaalan, K. F. (2006). **A survey of web information extraction systems**. *IEEE transactions on knowledge and data engineering*, 18(10) :1411–1428.
- Chau, M., Zeng, D., Chen, H., Huang, M., et Hendriawan, D. (2003). **Design and evaluation of a multi-agent collaborative web mining system**. *Decision Support Systems*, 35(1) :167–183.
- Chen, M., Mao, S., et Liu, Y. (2014). **Big data : A survey**. *Mobile Networks and Applications*, 19(2) :171–209.
- Chen, Y. (2007). **A novel hybrid focused crawling algorithm to build domain-specific collections**. PhD thesis, Virginia Polytechnic Institute and State University.
- Cho, J., Garcia-Molina, H., et Page, L. (1998). **Efficient crawling through url ordering**.
- Cilibrasi, R. L., et Vitanyi, P. M. (2007). **The google similarity distance**. *IEEE Transactions on knowledge and data engineering*, 19(3).
- Costa, R., Lima, C., Sarraipa, J., et Jardim-Gonçalves, R. (2013). **Semantic enrichment of building and construction knowledge sources using a domain ontology for classification**. In *AIP Conference Proceedings*, volume 1558, pages 1381–1384.
- Danlos, L., Nakamura, T., et Pradet, Q. (2015). **Traduction de verbnet vers le français**. In *Congrès ACFAS*, page 1.

- De Bra, P., Houben, G.-J., Kornatzky, Y., et Post, R. (1994). **Information retrieval in distributed hypertexts**. In *Intelligent Multimedia Information Retrieval Systems and Management-Volume 1*, pages 481–491. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.
- De Knijff, J., Frasincar, F., et Hogenboom, F. (2013). **Domain taxonomy learning from text : The subsumption method versus hierarchical clustering**. *Data & Knowledge Engineering*, 83 :54–69.
- Dean, J., et Ghemawat, S. (2008). **MapReduce : Simplified Data Processing on Large Clusters**. *Communications of the ACM*, 51(1) :1–13.
- Dellschaft, K., et Staab, S. (2006). **On how to perform a gold standard based evaluation of ontology learning**. In *International Semantic Web Conference*, pages 228–241. Springer.
- Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., Gori, M., et others (2000). **Focused crawling using context graphs**. In *VLDB*, pages 527–534.
- Dong, H., et Hussain, F. K. (2014). **Self-adaptive semantic focused crawler for mining services information discovery**. *IEEE Transactions On Industrial Informatics*, 10(2) :1616–1626.
- Dubremetz, M. (2013). **Vers une identification automatique du chiasme de mots**. *Actes de la 15e Rencontres des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL'2013)*, pages 150–163.
- Dunning, T. (1993). **Accurate Methods for the Statistics of Surprise and Coincidence**. *Computational Linguistics*, 19 :61–74.
- Ehrig, M., et Maedche, A. (2003). **Ontology-focused crawling of web documents**. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 1174–1178. ACM.
- Elberrichi, Z., Amel, B., et Malika, T. (2012). **Medical Documents Classification Based on the Domain Ontology MeSH**. *arXiv preprint arXiv :1207.0446*.
- Endrédy, I., et Novák, A. (2013). **More effective boilerplate removal-the goldminer algorithm**. *Polibits*, (48) :79–83.
- Fang, J., Guo, L., et Niu, Y. (2010). **Documents classification by using ontology reasoning and similarity measure**. In *Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010*, volume 4, pages 1535–1539.
- Farias, T. M., Roxin, A., et Nicolle, C. (2015). **FOWLA, A federated architecture for ontologies**. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9202, pages 97–111.
- Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., Schler, Y., et Zamir, O. (1998). **Text Mining at the Term Level**. In *Second European Symposium, PKDD '98*, number September, pages 65–73. Springer-Verlag.
- Gagnon, M. (2007). **Logique descriptive et owl**. *Cours dispensé à l'école polytechnique de Montréal, Canada*.

- Galinina, A., et Borisov, A. (2013). **Knowledge modelling for ontology-based multiattribute classification system.** *Applied Information and Communication* . . . , pages 103–109.
- Garrido, A. L., Gómez, O., Ibarri, S., et Mena, E. (2012). **An experience developing a semantic annotation system in a media group.** In *International Conference on Application of Natural Language to Information Systems*, pages 333–338. Springer.
- Griffiths, T. L., et Steyvers, M. (2004). **Finding scientific topics.** *Proceedings of the National academy of Sciences*, 101(suppl 1) :5228–5235.
- Guérif, S. (2006). **Réduction de dimension en apprentissage numérique non supervisé.** PhD thesis, Paris 13.
- Hadouche, F., et Lapalme, G. (2010). **Une version électronique du lvf comparée avec d'autres ressources lexicales.** *Langages*, 179(3) :193–220.
- Hassan, T., Cruz, C., et Bertaix, A. (2017a). **Ontology-based approach for unsupervised and adaptive focused crawling.** In *Proceedings of The International Workshop on Semantic Big Data*, page 2. ACM.
- Hassan, T., Cruz, C., et Bertaix, A. (2017b). **Predictive and evolutive cross-referencing for web textual sources.** In *SAI Computing Conference (SAI)*, 2017. IEEE.
- Hassan, T., Peixoto, R., Cruz, C., Bertaix, A., et Silva, N. (2015). **Semantic HMC for big data analysis.** In *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, pages 26–28.
- Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R. J., et Wang, M. (2009). **A framework for semantic link discovery over relational data.** In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1027–1036. ACM.
- Hersovici, M., Jacovi, M., Maarek, Y. S., Pelleg, D., Shtalhaim, M., et Ur, S. (1998). **The shark-search algorithm. an application : tailored web site mapping.** *Computer Networks and ISDN Systems*, 30(1) :317–326.
- Hitzler, P., et Janowicz, K. (2013). **Linked Data, Big Data, and the 4th Paradigm.** *Semantic Web*, 4(3) :333–335.
- Hliaoutakis, A., Varelas, G., Voutsakis, E., Petrakis, E. G., et Milius, E. (2006). **Information retrieval by semantic similarity.** *International journal on semantic Web and information systems (IJSWIS)*, 2(3) :55–73.
- Hulten, G., Spencer, L., et Domingos, P. (2001). **Mining time-changing data streams.** In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM.
- Jiang, J. J., et Conrath, D. W. (1997). **Semantic similarity based on corpus statistics and lexical taxonomy.** *arXiv preprint cmp-lg/9709008*.

- Johnson, I., Abécassis, J., Charnomordic, B., Destercke, S., et Thomopoulos, R. (2010). **Making ontology-based knowledge and decision trees interact : An approach to enrich knowledge and increase expert confidence in data-driven models.** In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6291 LNAI, pages 304–316. Springer.
- Kassim, J. M., et Rahmany, M. (2009). **Introduction to semantic search engine.** In *Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on*, volume 2, pages 380–386. IEEE.
- Kboubi, F., Habacha, A., et BenAhmed, M. (2010). **L'exploitation des relations d'association de termes pour l'enrichissement de l'indexation de documents textuels.** In *Proceedings of 10th International Conference of Journees d'Analyse statistique des Donnees Textuelles*, pages 9–11.
- Kleinberg, J. M. (1999). **Authoritative sources in a hyperlinked environment.** *Journal of the ACM (JACM)*, 46(5) :604–632.
- Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., et Tomkins, A. S. (1999). **The web as a graph : measurements, models, and methods.** In *International Computing and Combinatorics Conference*, pages 1–17. Springer.
- Kocev, D., Vens, C., Struyf, J., et Džeroski, S. (2007). **Ensembles of multi-objective decision trees.** In *European Conference on Machine Learning*, pages 624–631. Springer.
- Kohlschütter, C., Fankhauser, P., et Nejdl, W. (2010). **Boilerplate detection using shallow text features.** In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM.
- Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., et Saarela, A. (2000). **Self organization of a massive document collection.** *IEEE transactions on neural networks*, 11(3) :574–585.
- Kong, X., Shi, X., et Yu, P. S. (2011). **Multi-label collective classification.** In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 618–629. SIAM.
- Kraft, R., et Stata, R. (2003). **Finding buying guides with a web carnivore.** In *Web Congress, 2003. Proceedings. First Latin American*, pages 84–92. IEEE.
- Lambe, P. (2014). **Organising knowledge : taxonomies, knowledge and organisational effectiveness.** Elsevier.
- Li, J., Furuse, K., et Yamaguchi, K. (2005). **Focused crawling by exploiting anchor text using decision tree.** In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1190–1191. ACM.
- Lin, J. (2013). **Monoidify! monoids as a design principle for efficient mapreduce algorithms.** *arXiv preprint arXiv :1304.7544*.
- Lin, J., et Dyer, C. (2010). **Data-intensive text processing with mapreduce.** *Synthesis Lectures on Human Language Technologies*, 3(1) :1–177.

- Liu, B. (2007). **Web data mining : exploring hyperlinks, contents, and usage data.** Springer Science & Business Media.
- Lops, P., De Gemmis, M., et Semeraro, G. (2011). **Content-based recommender systems : State of the art and trends.** In *Recommender systems handbook*, pages 73–105. Springer.
- Lu, H., Zhan, D., Zhou, L., et He, D. (2016). **An improved focused crawler : Using web page classification and link priority evaluation.** *Mathematical Problems in Engineering*, 2016.
- Madjarov, G., Kocev, D., Gjorgjevikj, D., et Džeroski, S. (2012). **An extensive experimental comparison of methods for multi-label learning.** *Pattern Recognition*, 45(9) :3084–3104.
- Maedche, A., et Staab, S. (2001). **Ontology learning for the semantic web.** *IEEE Intelligent systems*, 16(2) :72–79.
- Malherbe, M. (1983). **Les langages de l'humanité(une encyclopédie des 3000 langues parlées dans le monde).** Bouquins.
- Manning, C. D., Raghavan, P., Schütze, H., et others (2008). **Introduction to information retrieval**, volume 1. Cambridge university press Cambridge.
- McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., et Barton, D. (2012). **Big data. The management revolution.** *Harvard Bus Rev*, 90(10) :61–67.
- McCown, F., et Nelson, M. L. (2007). **Agreeing to disagree : search engines and their public interfaces.** In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 309–318. ACM.
- Meijer, K., Frasincar, F., et Hogenboom, F. (2014). **A semantic approach for extracting domain taxonomies from text.** *Decision Support Systems*, 62 :78–93.
- Menczer, F. (2004). **Lexical and semantic clustering by web links.** *Journal of the American Society for Information Science and Technology*, 55(14) :1261–1269.
- Menczer, F., Pant, G., et Srinivasan, P. (2004). **Topical web crawlers : Evaluating adaptive algorithms.** *ACM Transactions on Internet Technology (TOIT)*, 4(4) :378–419.
- Miller, G. A. (1995). **Wordnet : a lexical database for english.** *Communications of the ACM*, 38(11) :39–41.
- Moller, R., et Haarslev, V. (2009). **Tableau-Based Reasoning.** In *Handbook on Ontologies*, page 654.
- Mouton, C., Richert, B., et de Chalendar, G. (2009). **Traduction de framenet par dictionnaires bilingues avec evaluation sur la paire anglais-francais.**
- Nazarenko, A., et Zargayouna, H. (2009). **Evaluating term extraction.** In *International Conference Recent Advances in Natural Language Processing (RANLP'09)*, pages 299–304.
- Page, L., Brin, S., Motwani, R., et Winograd, T. (1999). **The pagerank citation ranking : Bringing order to the web.** Technical Report, Stanford InfoLab.

- Pant, G., et Srinivasan, P. (2005). **Learning to crawl : Comparing classification schemes.** *ACM Transactions on Information Systems (TOIS)*, 23(4) :430–462.
- Pant, G., et Srinivasan, P. (2006). **Link contexts in classifier-guided topical crawlers.** *IEEE Transactions on Knowledge and Data Engineering*, 18(1) :107–122.
- Pant, G., Tsiotouliklis, K., Johnson, J., et Giles, C. L. (2004). **Panorama : extending digital libraries with topical crawlers.** In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 142–150. ACM.
- Papanikolaou, Y., Rubin, T. N., et Tsoumakas, G. (2015). **Improving gibbs sampling predictions on unseen data for latent dirichlet allocation.** *arXiv preprint arXiv :1505.02065*.
- Peixoto, R., Cruz, C., et Silva, N. (2015a). **Semantic hmc : Ontology-described hierarchy maintenance in big data context.** In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 492–501. Springer.
- Peixoto, R., Cruz, C., et Silva, N. (2016a). **Adaptive learning process for the evolution of ontology-described classification model in big data context.** In *SAI Computing Conference (SAI), 2016*, pages 532–540. IEEE.
- Peixoto, R., Hassan, T., Cruz, C., Bertaux, A., et Silva, N. (2015b). **Semantic hmc : a predictive model using multi-label classification for big data.** In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 2, pages 173–179. IEEE.
- Peixoto, R., Hassan, T., Cruz, C., Bertaux, A., et Silva, N. (2015c). **Semantic HMC : A Predictive Model using Multi-Label Classification For Big Data.** In *The 9th IEEE International Conference on Big Data Science and Engineering (IEEE BigDataSE-15)*.
- Peixoto, R., Hassan, T., Cruz, C., Bertaux, A., et Silva, N. (2016b). **An unsupervised classification process for large datasets using web reasoning.** In ACM, editor, *SBD'16 : Semantic Big Data Proceedings*, San Francisco (CA), USA.
- Peixoto, R., Hassan, T., Cruz, C., Bertaux, A., et Silva, N. (2016c). **Hierarchical multi-label classification using web reasoning for large datasets.** *Open Journal of Semantic Web (OJSW)*, 3(1) :1–15.
- Perret, L. (2005). **A question answering system for French.** Springer.
- Pomíkálek, J. (2011). **Removing boilerplate and duplicate content from web corpora.** *Disertacní práce, Masarykova univerzita, Fakulta informatiky*.
- Porter, M. F. (2001). **Snowball : A language for stemming algorithms.**
- Porzel, R., et Malaka, R. (2004). **A task-based approach for ontology evaluation.** In *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*. Citeseer.
- Pradet, Q., de Chalendar, G., et Desormeaux Baguenier, J. (2014). **Wonef, an improved, expanded and evaluated automatic french translation of wordnet.** In *Proceedings of the Seventh Global Wordnet Conference (GWC2014)*, pages 32–39.
- Raad, J., et Cruz, C. (2015). **A survey on ontology evaluation methods.** In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*.

- Read, J., Pfahringer, B., Holmes, G., et Frank, E. (2011). **Classifier chains for multi-label classification**. *Machine learning*, 85(3) :333–359.
- Ricci, F., Rokach, L., et Shapira, B. (2011). **Introduction to recommender systems handbook**. Springer.
- Ristoski, P., et Paulheim, H. (2016). **Semantic web in data mining and knowledge discovery : A comprehensive survey**. *Web semantics : science, services and agents on the World Wide Web*, 36 :1–22.
- Sagot, B., et Fišer, D. (2008). **Construction d'un wordnet libre du français à partir de ressources multilingues**. In *TALN 2008-Traitement Automatique des Langues Naturelles*.
- Salton, G., et Buckley, C. (1988). **Term-weighting approaches in automatic text retrieval**. *Information Processing & Management*, 24(5) :513–523.
- Salton, G., Wong, A., et Yang, C.-S. (1975). **A vector space model for automatic indexing**. *Communications of the ACM*, 18(11) :613–620.
- Sanderson, M., et Croft, B. (1999). **Deriving concept hierarchies from text**. *22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213.
- Santos, A. P., et Rodrigues, F. (2009). **Multi-label hierarchical text classification using the acm taxonomy**. In *14th Portuguese Conference on Artificial Intelligence (EPIA)*, pages 553–564.
- Savoy, J. (1999). **A stemming procedure and stopword list for general french corpora**. *JASIS*, 50(10) :944–952.
- Savoy, J. (2006). **Light stemming approaches for the french, portuguese, german and hungarian languages**. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1031–1035. ACM.
- Schmidt-Schauß, M., et Smolka, G. (1991). **Attributive concept descriptions with complements**. *Artificial intelligence*, 48(1) :1–26.
- Schuler, K. K. (2005). **Verbnet : A broad-coverage, comprehensive verb lexicon**.
- Shearer, R., et Horrocks, I. (2009). **Hypertableau Reasoning for Description Logics**. *Journal of Artificial Intelligence Research*, 36(June 2008) :165–228.
- Sheth, A. (2014). **Transforming big data into smart data : Deriving value via harnessing volume, variety, and velocity using semantic techniques and technologies**. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 2–2. IEEE.
- Silla Jr, C. N., et Freitas, A. A. (2011). **A survey of hierarchical classification across different application domains**. *Data Mining and Knowledge Discovery*, 22(1-2) :31–72.
- Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., et Katz, Y. (2007). **Pellet : A practical OWL-DL reasoner**.

- Soualah-Alila, F. (2015). **CAMLearn : Une Architecture de Système de Recommandation Sémantique Sensible au Contexte. Application au Domaine du M-Learning.** PhD thesis, Université de Bourgogne.
- Studer, R., Benjamins, V. R., et Fensel, D. (1998a). **Knowledge engineering : principles and methods.** *Data & knowledge engineering*, 25(1) :161–197.
- Studer, R., Benjamins, V. R., et Fensel, D. (1998b). **Knowledge engineering : principles and methods.** *Data & knowledge engineering*, 25(1-2) :161–197.
- Su, C., Gao, Y., Yang, J., et Luo, B. (2005). **An efficient adaptive focused crawler based on ontology learning.** In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 6–pp. IEEE.
- Sucar, L. E., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H., et Larrañaga, P. (2014). **Multi-label classification with bayesian network-based chain classifiers.** *Pattern Recognition Letters*, 41 :14 – 22. Supervised and Unsupervised Classification Techniques and their Applications.
- Sukhbaatar, S., Weston, J., Fergus, R., et others (2015). **End-to-end memory networks.** In *Advances in neural information processing systems*, pages 2440–2448.
- Syed, A. R., Gillela, K., et Venugopal, C. (2013). **The Future Revolution on Big Data.** *International Journal of Advanced Research in Computer and Communication Engineering*, 2(6) :2446–2451.
- Tarski, A. (1941). **On the calculus of relations.** *The Journal of Symbolic Logic*, 6(03) :73–89.
- The Apache Software Foundation (2013). **Mahout**.
- Tollari, S. (2006). **Indexation et recherche d'images par fusion d'informations textuelles et visuelles.** PhD thesis, Toulon.
- Tolle, K. M., et Chen, H. (2000). **Comparing noun phrasing techniques for use with medical digital library tools.** *Journal of the American Society for Information Science*, 51(4) :352–370.
- Tsarkov, D., et Horrocks, I. (2006). **FaCT++ Description Logic Reasoner : System Description.** In Furbach, U., et Shankar, N., editors, *Proceedings of the Third International Joint Conference (IJCAR)*, volume 4130, pages 292–297. Springer Berlin / Heidelberg.
- Tsoumakas, G., Katakis, I., et Overview, A. (2007). **Multi-Label Classification : An Overview.** *International Journal of Data Warehousing and Mining*, 3(September) :1–13.
- Tsoumakas, G., Katakis, I., et Vlahavas, I. (2008). **Effective and efficient multilabel classification in domains with large number of labels.** In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, pages 30–44.
- Tsoumakas, G., Katakis, I., et Vlahavas, I. (2009). **Mining multi-label data.** In *Data mining and knowledge discovery handbook*, pages 667–685. Springer.
- Urbani, J. (2013). **Three Laws Learned from Web-scale Reasoning.** In *2013 AAAI Fall Symposium Series*, pages 76–79.

- Urbani, J., Kotoulas, S., Maassen, J., Van Harmelen, F., et Bal, H. (2012). **WebPIE : A Web-scale Parallel Inference Engine using MapReduce**. *Journal of Web Semantics*, 10 :59–75.
- Urbani, J., Van Harmelen, F., Schlobach, S., et Bal, H. (2011). **QueryPIE : Backward reasoning for OWL horst over very large knowledge bases**. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7031 LNCS of ISWC'11, pages 730–745, Berlin, Heidelberg. Springer-Verlag.
- van den Bosch, A., Bogers, T., et de Kunder, M. (2015). **A longitudinal analysis of search engine index size**. In *Proceedings of ISSI 2015 Istanbul : 15th International Society of Scientometrics and Informetrics Conference, Istanbul, Turkey, 29 June to 3 July, 2015*.
- Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E. G., et Milios, E. E. (2005). **Semantic similarity methods in wordnet and their application to information retrieval on the web**. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 10–16. ACM.
- Vivaldi, J., et Rodriguez, H. (2007). **Evaluation of terms and term extraction systems : A practical approach**. *Terminology*, 13(2) :225–248.
- Vogrincic, S., et Bosnic, Z. (2011). **Ontology-based multi-label classification of economic articles**. *Comput. Sci. Inf. Syst.*, 8(1) :101–119.
- Volz, J., Bizer, C., Gaedke, M., et Kobilarov, G. (2009). **Silk-a link discovery framework for the web of data**. *LDOW*, 538.
- Werner, D. (2015). **Indexation et recommandation d'informations : vers une qualification precise des items par une approche ontologique, fondee sur une modelisation metier du domaine : application à la recommandation d'articles économiques**. PhD thesis. These de doctorat dirigée par Cruz, Christophe et Bertaux, Aurelie Informatique Dijon 2015.
- Werner, D., Silva, N., Cruz, C., et Bertaux, A. (2014). **Using DL-reasoner for hierarchical multilabel classification applied to economical e-news**. In *Proceedings of 2014 Science and Information Conference, SAI 2014*, pages 313–320.
- Widmer, G., et Kubat, M. (1996). **Learning in the presence of concept drift and hidden contexts**. *Machine learning*, 23(1) :69–101.
- Wong, W., Liu, W., et Bennamoun, M. (2012). **Ontology learning from text : A look back and into the future**. *ACM Computing Surveys (CSUR)*, 44(4) :20.
- Xu, Q., et Zuo, W. (2007). **First-order focused crawling**. In *Proceedings of the 16th international conference on World Wide Web*, pages 1159–1160. ACM.
- Yu, H.-F., Jain, P., Kar, P., et Dhillon, I. (2014). **Large-scale multi-label learning with missing labels**. In *International Conference on Machine Learning*, pages 593–601.
- Zhang, M.-L., et Zhou, Z.-H. (2007). **MI-knn : A lazy learning approach to multi-label learning**. *Pattern recognition*, 40(7) :2038–2048.

- Zhang, X., Zhao, J., et LeCun, Y. (2015). **Character-level convolutional networks for text classification**. In *Advances in neural information processing systems*, pages 649–657.
- Zheng, H.-T., Kang, B.-Y., et Kim, H.-G. (2008). **An ontology-based approach to learnable focused crawling**. *Information Sciences*, 178(23) :4512–4522.

TABLE DES FIGURES

1.1 Système de recommandation mis en place dans les travaux de Werner (2015)	5
2.1 Types de structure pour la classification	18
2.2 Classification d'un item selon une hiérarchie et un DAG	19
3.1 Le processus HMC Sémantique	26
3.2 Le processus SHMC : indexation, vectorisation et hiérarchisation	28
3.3 Exemple de chaîne de traitement	30
3.4 Exemple d'index inversé	31
3.5 Diagrammes d'Euler de la méthode de subsomption	37
3.6 Nombre de termes issus de l'apprentissage	40
3.7 Nombre de labels issus de l'apprentissage	40
3.8 Nombre de relations hiérarchiques issus de l'apprentissage	41
3.9 Distance sémantique globale en fonction du seuil $st1$	47
3.10 Nombre de relations hiérarchiques en fonction du seuil $st1$	48
4.1 Le processus SHMC : résolution et réalisation	49
4.2 Ensembles Alpha et Beta	51
4.3 Classification avant la requête et chaînage avant	56
4.4 Classification au moment de la requête et chaînage arrière	56
4.5 Nombre de règles en fonction du jeu de données	59
4.6 Nombre de classifications (relations <i>isClassified</i>) en fonction du jeu de données	60
4.7 Variation des seuils de hiérarchisation. $Alpha = 90, Beta = 10$ (valeurs par défaut)	61
4.8 Variation des seuils de résolution. $st1 = 80, st2 = 70$ (valeurs par défaut)	62
4.9 Portion de la hiérarchie de labels automatiquement générée à partir du jeu de données Delicious	65
5.1 Contenu pertinent en vert. Contenu peu pertinent en rouge.	72
5.2 Description d'un crawler générique	73

5.3 Exemple de distribution de pertinence des pages	76
6.1 Architecture SemXDM	87
6.2 Module de recommandation, interaction avec l'utilisateur	89
6.3 Module de classification	90
6.4 Intégration des modules Crawler et Classification	92
6.5 Maintenance Module	93
6.6 Architecture du module de maintenance	95
6.7 Module de priorité	97
6.8 Exemple de graphe contextuel à 3 couches	98
6.9 Implémentation du module de maintenance	102
6.10 Nombre de classifications (inférences) total	105
6.11 Nombre de cooccurrences détectées par cycle (maintenance)	106
6.12 Nombre de nouveaux termes détectés par cycle	106
6.13 Nombre de modifications appliquées au modèle prédictif	107
6.14 Taille du graphe contextuel	107
6.15 Evolution de la hiérarchie de labels	108
6.16 Evolution des termes et des règles	108
6.17 Similarité moyenne et taux de récolte par cycle (requête de croisement d'information)	110
6.18 Similarité moyenne par cycle	111
6.19 Similarité moyenne globale	111
6.20 Degré d'appartenance d_0 par cycle	112
6.21 Degré d'appartenance d_0 global	112
6.22 Similarité moyenne par cycle	113
6.23 Similarité moyenne globale	114
7.1 Sections du processus de production métier impacté par le nouvel outil . .	119
7.2 Exemple d'article Indexé	121
7.3 Exemple de recommandation. Le profil utilisateur est décrit à gauche par un ensemble de facettes.	122
7.4 Exemple d'article étiqueté	125
7.5 Phases de la collecte d'informations	126
7.6 Architecture du nouvel outil "FirstWatch"	128
7.7 Page d'authentification de l'application	130
7.8 Paramétrage d'une source d'information	130

7.9 Édition de la liste des signets	131
7.10 Signets : affichage de la page originale	131
7.11 Signets : affichage du texte extrait	132
7.12 Gestionnaire de sources d'information	132
A.1 Système de représentation de la connaissance	161
B.1 Fonctionnement Hadoop MapReduce	174
B.2 Principe d'une tâche MapReduce. <i>Source : http://www.glennclockwood.com/data-intensive/hadoop/overview.html</i>	174

LISTE DES TABLES

3.1 Concepts du modèle prédictif	27
3.2 Matrice de cooccurrence de taille 3×3	34
3.3 Jeux de données wikipedia	38
3.4 Spécifications matérielles	38
3.5 Paramétrage du processus	39
3.6 Temps d'exécution	42
3.7 Précision lexicale basée sur Wordnet et DBpedia	45
4.1 Exemples de règle Alpha	52
4.2 Exemples de règle Beta	52
4.3 Paramètres par défaut	59
4.4 Description du modèle initial (pré-résolution)	59
4.5 Valeurs des paramètres pour l'évaluation	64
4.6 Résultats de l'évaluation (Delicious/Bibtex)	65
4.7 Comparaison aux approches de la littérature (Delicious/Bibtex)	65
5.1 Positionnement de l'approche SEMXDM	82
6.1 Concepts du modèle prédictif	86
6.2 Types d'entrées détectées (Input Changes)	94
6.3 Types de requêtes de modification du modèle	96
6.4 Représentation du graphe web dans l'ontologie	96
A.1 Grammaire du langage de description \mathcal{AL}	163
A.2 La sémantique du langage de description de concepts selon \mathcal{AL}	164
A.3 Constructeurs des logiques de description	164
A.4 Syntaxe et sémantique des constructeurs	165
A.5 Familles de logique de description	165
A.6 Constructeurs OWL	166
A.7 Axiomes OWL 2	167

A.8	Complexité du raisonnement en fonction de la logique de description et donc de son expressivité	170
A.9	Caractéristiques des raisonneurs	171
B.1	Exemple Word Count, mapper	175
B.2	Exemple Word Count, reducer	175

LISTE DES DÉFINITIONS

1	Définition : Fouille de données	6
2	Définition : Logique de description	6
3	Définition : Ontologie	6
4	Définition : Moteur d'inférence	7
5	Définition : Traitement automatique du langage	7
6	Définition : Web crawler	7
7	Définition : Classification hiérarchique multi-étiquette sémantique	7
8	Définition : Raisonneur sémantique	50
9	Définition : Triple Store	50
10	Définition : Expressivité, décidabilité	160
11	Définition : Base de connaissance	161
12	Définition : Satisfaisabilité, Cohérence, Consistance	165

A

LOGIQUES DE DESCRIPTION ET RAISONNEMENT LOGIQUE

Cette annexe présente des notions relatives aux logiques de description et au raisonnement logique, nécessaires à la compréhension du manuscrit. Certaines notions essentielles sont introduites dans le manuscrit, notamment les définitions des concepts de logique de description, moteurs d'inférence, raisonneurs, et triples stores dans les sections 1.2 et 4.1.1.

Les Logiques de description (\mathcal{LD}) ou logiques descriptives sont une famille de langages basée sur le langage \mathcal{AL} (Attributive Language) introduits par Schmidt-Schauß et al. (1991). Elles définissent un formalisme pour représenter la connaissance terminologique d'un domaine d'application de manière structurée.

Les logiques de description ont une dimension descriptive et une dimension logique. La première concerne la représentation des connaissances, la seconde concerne le raisonnement logique à partir des connaissances. Leur combinaison permet notamment la déduction de connaissances inédites.

Cette annexe décrit précisément ces deux dimensions, et les met en rapport avec les technologies du domaine du Web Sémantique. Elle est fortement inspirée des travaux de Soualah-Alila (2015) et Werner (2015).

A.1/ LES LOGIQUES DE DESCRIPTION ET LE WEB SÉMANTIQUE

Les logiques de description (\mathcal{LD}) sont nées de la logique du premier ordre (au sens du formalisme mathématique), i.e. le calcul des prédictats du premier ordre, et plus spécifiquement de fragments décidables de la logique du premier ordre. En outre, certaines notions sur lesquelles se basent les logiques de description sont liées aux logiques modales propositionnelles et aux logiques dynamiques.

Définition 10 : Expressivité, décidabilité

La notion de décidabilité est liée au calcul des prédictats du premier ordre. Le calcul des prédictats du premier ordre est indécidable, i.e. il n'existe pas d'algorithme de décision complet pour cette logique. Les logiques de description sont nées de fragments décidables de la logique du premier ordre. Cette notion de décidabilité se retrouve en logique. Une affirmation logique est dite décidable si on peut la démontrer ou démontrer sa négation dans le cadre d'une théorie donnée^a. Une proposition est donc indécidable dans une théorie s'il est impossible de la déduire ou de déduire sa négation, à partir des axiomes de cette théorie. Les familles de logique de description sont construites autour d'une combinaison des constructeurs logiques (voir tableaux A.3 à A.5). L'expressivité d'un langage dépend des constructeurs logiques qui le définissent. Un langage plus expressif permet de représenter des langages plus complexes. L'ajout de constructeurs logiques à une logique augmente son expressivité, au détriment de sa décidabilité A.8.

a. <http://binaire.blog.lemonde.fr/2015/06/17/decidable-indecidable/>

Les logiques de description ont été introduits en tant que langage de modélisation de données dans les années 80, dans le but de rendre la représentation de connaissances plus naturelle et plus complète qu'avec la logique du premier ordre.

Les logiques de description ont une sémantique formelle conforme au cadre des logiques de Tarski (1941). Les connaissances d'un domaine y sont représentées par des entités qui correspondent à une syntaxe à laquelle est associée une sémantique :

(1) Les éléments du monde réel y sont représentés à l'aide de concepts, rôles et individus. Les concepts ainsi que les rôles sont organisés de façon hiérarchique à l'aide de relations de subsomption. L'utilisation de ce type de logique permet la réalisation de tâches de raisonnement.

(2) La sémantique des logiques de description est clairement définie et associée à la description des concepts ainsi que des rôles. Cette sémantique affecte la façon de manipuler ces éléments. Cette sémantique est définie comme suit : Soit $\mathfrak{C} = \{C_1, C_2, \dots\}$ un ensemble fini de concepts atomiques, $\mathfrak{R} = \{R_1, R_2, \dots\}$ un ensemble fini de rôles atomiques et $\mathfrak{I} = \{a_1, a_2, \dots\}$ un ensemble fini d'individus. Si $\mathfrak{C}, \mathfrak{R}, \mathfrak{I}$ sont disjoints deux à deux, $S = \langle \mathfrak{C}, \mathfrak{R}, \mathfrak{I} \rangle$ est une signature. Une fois qu'une signature S est fixée, un modèle d'interprétation \mathcal{I} pour S est défini comme un couple $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$, où $\Delta^{\mathcal{I}}$ est un ensemble non vide, le domaine d'interprétation, et $.^{\mathcal{I}}$ est la fonction d'interprétation qui associe :

- un élément $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ à chaque individu $a_i \in \mathfrak{I}$,
- un sous-ensemble $C_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ à chaque concept atomique $C_i \in \mathfrak{C}$
- une relation $R_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ à chaque rôle atomique $R_i \in \mathfrak{R}$.

A.1.1/ LES BASES DE CONNAISSANCES

Définition 11 : Base de connaissance

Une base de connaissance regroupe des connaissances spécifiques à un domaine spécialisé, sous une forme exploitable par un ordinateur. Une base de connaissance est composée d'une partie terminologique, qualifiée de *TBox* (i.e. Terminology Box), et d'une partie assertionnelle, qualifiée de *ABox* (i.e. Assertions Box). La *ABox* décrit les individus et leurs relations (quel individu appartient à quel concept nommé, quel individu est lié à quel autre à travers quel rôle).

La base de connaissances joue un rôle d'entrepôt de données dans un système de représentation de connaissances, à la fois pour la structure conceptuelle du modèle (*TBox*) et pour les individus (*ABox*). La figure A.1 décrit l'agencement des différents éléments d'un système de représentation de la connaissance basé sur une logique de description, autour d'une base de connaissances¹.

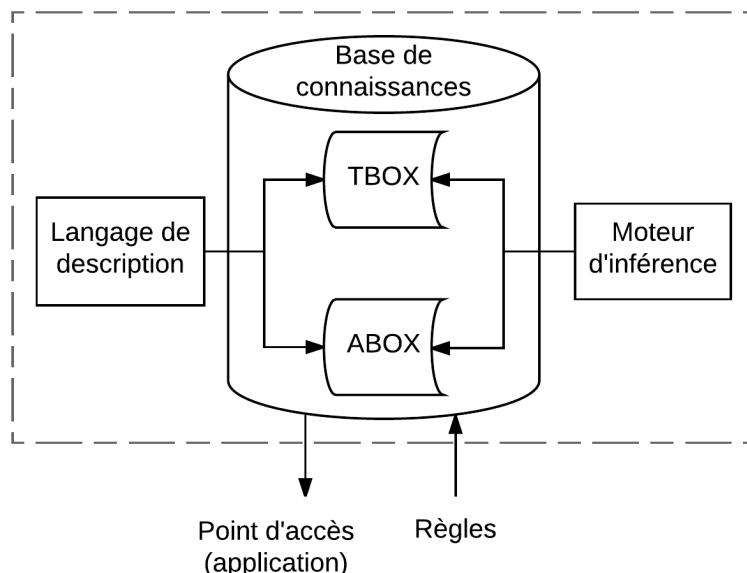


FIGURE A.1 – Système de représentation de la connaissance

Une base de connaissances typique basée sur une logique de description est définie formellement de la manière suivante :

Soit un langage de description \mathcal{L} et une signature \mathcal{S} , une base de connaissances Σ dans \mathcal{L} est une paire $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ telle que :

- la *TBox* \mathcal{T} est un ensemble fini, qui peut être vide, d'expressions appelées GCI ("General Concept Inclusion") de la forme $C \sqsubseteq D$ où C et D sont des concepts sans restriction. La *TBox* contient les définitions des concepts. Un nouveau concept peut être défini en fonction des concepts déjà présents dans la *TBox*. Seuls des axiomes d'équivalence $C \equiv D$ et d'inclusion $C \sqsubseteq D$ sont utilisés pour la constituer. De façon générale, $C \equiv D$ si et seulement si $C \sqsubseteq D$ et $D \sqsubseteq C$. Une interprétation \mathcal{I} d'une *TBox* satisfait $C \sqsubseteq D$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, et elle satisfait $C \equiv D$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \wedge D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$.

1. http://www.i3s.unice.fr/~nlt/cours/master/dl/dl_pmlt.pdf

- la *ABox* \mathcal{A} est un ensemble fini, qui peut être vide, d'expressions de la forme $C(a)$ ou $R(a, b)$, où C est un concept sans restriction, R est un rôle qui n'est pas nécessairement atomique, et a, b appartiennent à \mathfrak{I} . Les formules de \mathcal{A} sont appelées des assertions.

Une interprétation \mathcal{I} d'une *ABox* associe à chaque constante a un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$. \mathcal{I} est un modèle de la *ABox* si pour toute assertion $C(a) \Rightarrow a^{\mathcal{I}} \in C^{\mathcal{I}}$ et $R(a, b) \Rightarrow (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Un modèle est une interprétation qui satisfait tous les axiomes d'une base de connaissances. Les logiques de description adoptent l'hypothèse de nom unique, qui implique que les individus soient distincts et possèdent toujours des noms différents. De plus, les bases de connaissances reposant sur les logiques de description adoptent la sémantique du monde ouvert. L'hypothèse du monde ouvert implique que les informations peuvent être incomplètes. Ainsi, ce qui ne peut pas être prouvé à partir des informations disponibles n'est pas nécessairement faux.

L'hypothèse du monde clos implique que l'information dans la base de connaissances est nécessairement complète et donc que ce qui ne peut pas être prouvé à partir des informations disponibles est faux.

Par exemple, l'unicité de nommage implique dans l'hypothèse du monde ouvert que si deux individus ont le même nom, ils sont confondus et forment donc le même individu. Dans l'hypothèse du monde clos, la base de connaissances ne permettrait pas la création de deux individus du même nom, qui ne respectent pas la restriction de nommage unique.

A.1.2/ LES FAMILLES DE LOGIQUES DE DESCRIPTION

Les \mathcal{LD} forment une famille de langages de représentation de connaissances. Dans le formalisme des \mathcal{LD} , un concept représente un ensemble d'individus et un rôle représente une relation binaire entre individus. Un concept correspond à une entité générique d'un domaine d'application et un individu à une entité particulière, instance d'un concept.

Il existe plusieurs familles de langages de description des concepts, des rôles et des individus. La logique de description \mathcal{AL} (Attribute Language) est la logique minimale, ou logique attributive, définie par (Schmidt-Schauß et al., 1991). Elle est dite minimale car une logique d'expressivité inférieure n'aurait pas d'intérêt pratique Baader (2003). Les descriptions possibles dans le langage \mathcal{AL} sont présentées par le tableau A.1. On suppose que A est un concept atomique et que C et D sont des concepts atomiques ou complexes, i.e. définis à partir de concepts atomiques. Le langage de base \mathcal{AL} est défini à partir des éléments syntaxiques suivants :

- Le concept $\text{Top } \top$ dénote le concept le plus général et le concept $\text{Bottom } \perp$ le concept le plus spécifique. Intuitivement, l'extension de Top inclut tous les individus possibles tandis que celle de Bottom est vide.
- Le constructeur \neg correspond à la négation et ne porte que sur les concepts primaires (atomiques). Le constructeur $\text{and } \sqcap$ permet de définir une conjonction d'expressions conceptuelles.
- La quantification universelle $\forall R.C$ précise le co-domaine du rôle R et la quantification existentielle non typée $\exists R.\top$ introduit le rôle R et affirme l'existence d'au moins un couple d'individus en relation par l'intermédiaire de R . On obtient alors le langage $\mathcal{AL} = \{\top, \perp, A, \neg, C \sqcap D, \forall R.C, \exists R.\top\}$ qui peut être par la suite enrichi par plusieurs autres constructeurs.

TABLE A.1 – Grammaire du langage de description \mathcal{AL}

Syntaxe	Définition
\top	Concept universel Top
\perp	Concept vide Bottom
A	Concept atomique
$\neg A$	Complément de concept atomique
$C \sqcap D$	Conjonction de concepts
$\forall R.C$	Quantificateur universel
$\exists R.\top$	Quantificateur existentiel non typé

La sémantique du langage \mathcal{AL} fait appel à la théorie des ensembles. Essentiellement, à chaque concept est associé un ensemble d'individus dénotés par ce concept. Une interprétation suppose donc l'existence d'un ensemble non vide qui représente des entités du monde décrit. Une fonction d'interprétation (décrise précédemment) associe à un concept C un sous-ensemble C^I de Δ^I et à un rôle R un sous-ensemble R^I de $\Delta^I \times \Delta^I$ Gagnon (2007). La sémantique de \mathcal{AL} est alors définie dans le tableau A.2.

La logique \mathcal{AL} n'est généralement pas suffisamment expressive pour représenter certaines connaissances. Il existe plusieurs autres constructeurs que l'on peut ajouter à ce langage pour le rendre plus expressif. La logique \mathcal{ALC} est l'extension de la logique \mathcal{AL} par la négation complète C (i.e. négation de concepts non primitifs) Chaabani et al. (2009). \mathcal{AL} constitue la logique descriptive de base, \mathcal{ALC} la logique descriptive minimale telle que : $\mathcal{ALC} = \mathcal{AL} \cup \{\neg C\}$. Les différentes logiques de description qui existent aujourd'hui sont des déclinaisons de la logique de base \mathcal{ALC} avec les différents constructeurs du tableau A.3. Le tableau A.4 décrit la syntaxe et la sémantique des différents constructeurs des logiques de description.

Les langages ainsi obtenus sont plus expressifs, tels qu'avec l'utilisation des constructeurs suivants :

- Le constructeur \mathcal{U} permet de définir la disjonction de concepts, notée $C \sqcup D$. L'extension correspondante d' \mathcal{ALC} est \mathcal{ALCU} .
- Le constructeur \mathcal{R} permet de définir la conjonction de rôles, notée $R_1 \sqcap R_2$, les rôles R_1 et R_2 étant primitifs. L'extension correspondante d' \mathcal{ALC} est \mathcal{ALCR} .
- La quantification existentielle typée \mathcal{E} , notée $\exists R.C$ (la quantification $\exists R.C$ introduit un rôle R de co-domaine C et impose l'existence d'au moins un couple d'individus (x,y) en relation par l'intermédiaire de R , où C est le type de y). L'extension correspondante d' \mathcal{ALC} est \mathcal{ALCE} .
- Le constructeur \mathcal{N} , noté $\geq nR$ (i.e. au moins nR) et $\leq nR$ (i.e. au plus nR), fixent la cardinalité minimale et maximale du rôle auquel ils sont associés. L'extension correspondante d' \mathcal{ALC} est \mathcal{ALCN} .

Certaines logiques sont équivalentes, notamment \mathcal{ALC} et \mathcal{ALUE} . Ces deux dernières logiques augmentées par \mathcal{R}^+ sont notées \mathcal{S} . Les principales familles de logiques sont illustrées dans le tableau A.5.

TABLE A.2 – La sémantique du langage de description de concepts selon \mathcal{AL}

Syntaxe	Sémantique
A	A^T
\top	Δ^T
\perp	\emptyset
$\neg A$	$\Delta^T \setminus A^T$
$C \sqcap D$	$C^T \cap D^T$
$\forall R.C$	$\{x \in \Delta^T \mid \forall y \in \Delta^T, (x, y) \in R^T \Rightarrow y \in C^T\}$
$\exists R.\top$	$\{x \in \Delta^T \mid \exists y \in \Delta^T, (x, y) \in R^T\}$

TABLE A.3 – Constructeurs des logiques de description

Symbol	Définition
\mathcal{F}	Rôles fonctionnels
\mathcal{E}	Quantifications existentielles typées
\mathcal{U}	Disjonction de rôles
\mathcal{C}	Négation complète
\mathcal{S}	Abréviation de la logique \mathcal{ALC} avec la transitivité des rôles \mathcal{R}^+
\mathcal{H}	Hiérarchie de rôles
\mathcal{O}	Classes nominales ou énumération par individus
\mathcal{I}	Rôles inverses
\mathcal{N}	Restrictions de cardinalité
\mathcal{Q}	Restrictions de cardinalité qualifiée
(\mathcal{D})	Support des types primitifs

A.1.3/ LE LANGAGE ONTOLOGIQUE OWL

Le langage OWL, proposé par le consortium W3C, est le langage de représentation des connaissances permettant la définition d'ontologies. Il est basé sur les travaux du domaine des logiques de description précédemment introduites. Il hérite des logiques de description la possibilité de mise en place de processus de vérification de cohérence et de consistance ainsi que de déduction de connaissances à partir des connaissances existantes. Ces propriétés font partie des mécanismes d'inférence, effectués dans le cadre du web sémantique par un raisonneur conforme au standard OWL (les différents mécanismes d'inférence sont détaillés dans la section suivante).

TABLE A.4 – Syntaxe et sémantique des constructeurs

Constructeurs	Syntaxe	Sémantique
\mathcal{F}	$\leq 1R$	$\{x \in \Delta^{\mathcal{I}} y, (x, y) \in R^{\mathcal{I}} \leq 1\}$
\mathcal{E}	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \exists y \in \Delta^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
\mathcal{U}	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
\mathcal{C}	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
\mathcal{H}	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
\mathcal{O}	\mathcal{I}	$\mathcal{I}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ avec $ \mathcal{I}^{\mathcal{I}} = 1$
\mathcal{I}	R^{-1}	$\{(x, y) \in R^{\mathcal{I}} (y, x) \in R^{\mathcal{I}}\}$
\mathcal{N}	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} y, (x, y) \in R^{\mathcal{I}} \geq n\}$
\mathcal{N}	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} y, (x, y) \in R^{\mathcal{I}} \leq n\}$
\mathcal{N}	$= nR$	$\{x \in \Delta^{\mathcal{I}} y, (x, y) \in R^{\mathcal{I}} = n\}$
\mathcal{Q}	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} y \in C^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}} \geq n\}$
\mathcal{Q}	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} y \in C^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}} \leq n\}$
\mathcal{Q}	$= nR.C$	$\{x \in \Delta^{\mathcal{I}} y \in C^{\mathcal{I}}, (x, y) \in R^{\mathcal{I}} = n\}$
\mathcal{R}	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$

TABLE A.5 – Familles de logique de description

Composition	Famille
\mathcal{AL}	Logique de base
\mathcal{ALC}	Logique minimale
$\mathcal{ALC} + \mathcal{N}$	Logique \mathcal{ALCN}
$\mathcal{ALC} + \mathcal{Q}$	Logique \mathcal{ALCQ}
$\mathcal{ALC} + \mathcal{F}$	Logique \mathcal{ALCF}
$\mathcal{ALC} + \mathcal{I}$	Logique \mathcal{ALCI}
$\mathcal{ALC} + \mathcal{R}^+$	Logique \mathcal{S}
$\mathcal{S} + \mathcal{H}$	Logique \mathcal{SH}
$\mathcal{SH} + \mathcal{I} + \mathcal{F}$	Logique \mathcal{SHIF}
$\mathcal{SH} + \mathcal{I} + \mathcal{Q}$	Logique \mathcal{SHIQ}
$\mathcal{SH} + \mathcal{O} + \mathcal{I} + \mathcal{N}$	Logique \mathcal{SHOIN}
$\mathcal{SH} + \mathcal{O} + \mathcal{I} + \mathcal{Q}$	Logique \mathcal{SHOIQ}

Définition 12 : Satisfaisabilité, Cohérence, Consistance

Une base de connaissances $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ est incohérente si au moins un des concepts C appartenant à sa TBox \mathcal{T} est insatisfiable : si pour chaque interprétation \mathcal{I} le nombre d'instances du concept C est égale à zéro, i.e. il n'existe pas de concept C tel que $\Sigma \models C \equiv \perp$.

Un concept C est satisfiable s'il existe au moins un modèle d'interprétation \mathcal{I} de \mathcal{T} pour lequel $C^{\mathcal{I}}$ est non vide. Dans ce cas \mathcal{I} est un modèle de C .

Une base de connaissances $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ est inconsistante s'il n'existe pas d'interprétation \mathcal{I} qui soit un modèle pour sa TBox \mathcal{T} et sa ABox \mathcal{A} . Elle est consistante dans le cas contraire.

L'ontologie est une structure de données largement répandue pour la réalisation de bases de connaissances. Le langage *OWL* est composé de trois sous-langages : *OWL-Lite*, *OWL-DL* et *OWLFull*. *OWL-Lite* correspond essentiellement à la famille de logiques *SHIF* alors que *OWL-DL* ("Description Logic"), correspond à la famille *SHOIN*. Ces deux logiques sont une augmentation de l'expressivité de la logique *SH*, elle-même basée sur la logique *S* à laquelle la gestion des hiérarchies de rôles a été ajoutée. Plus précisément, il s'agit des familles *SHIF(D)* et *SHOIN(D)*. La distinction est faite sur les deux types de rôles : les rôles qui lient deux individus et les rôles qui associent un individu avec un type primitif (entiers, chaîne de caractères, etc.), d'où le (*D*), pour Data Property. *OWL-Lite* est un sous-langage d'*OWL-DL*. *OWL-DL* est une version décidable du langage *OWL*. Il permet donc la réalisation de raisonnements logiques, contrairement à *OWL-Full*.

En 2009, le consortium W3C a officiellement lancé *OWL 2*, qui se distingue de la première version par un pouvoir expressif augmenté et une élimination de la décomposition en trois sous-langages. Ainsi *OWL-Full* n'existe plus, alors que *OWL-Lite* et *OWL-DL* sont considérés comme des profils de *OWL 2*, où certains constructeurs limitants sans pouvoir expressif ont été ôtés de *OWL 2*. Depuis 2012, *OWL 2* est la recommandation du W3C et permet l'expressivité de la logique *SROIQ(D)* Gagnon (2007). Le langage *OWL 2* a l'expressivité de la logique *SROIQ* qui est une augmentation de l'expressivité de la logique *SR* basée sur la logique *S* à laquelle contrairement la logique *SH* les inclusion de rôles *R* ont été ajoutées en plus des hiérarchies de rôles.

TABLE A.6 – Constructeurs OWL

Constructeurs	Syntaxe	Exemple
intersectionOf	$C_1 \sqcap C_2$	<i>Humain</i> \sqcap <i>Homme</i>
unionOf	$C_1 \sqcup C_2$	<i>Docteur</i> \sqcup <i>Avocat</i>
complementOf	$\neg C$	\neg <i>Homme</i>
oneOf	{ $a_1 \dots a_2$ }	{ <i>josé, nathan, michel</i> }
allValueFrom	$\forall R.C$	\forall <i>possedeEnfant</i> . <i>Docteur</i>
someValueFromFrom	$\exists R.C$	\exists <i>possedeEnfant</i> . <i>Avocat</i>
hasValue	$\exists R.\{a\}$	\exists <i>citoyenDe</i> .{ <i>France</i> }
minCardinality	($\geq nR$)	(≥ 2 <i>possedeEnfant</i>)
maxCardinality	($\leq nR$)	(≤ 1 <i>possedeEnfant</i>)
inverseOf	R^-	<i>possedeEnfant</i> $^-$

Les tableaux A.6 et A.7 présentent les constructeurs du langage *OWL* et les axiomes du langage *OWL 2* ainsi que leur correspondance en logiques de description.

La création d'ontologies est généralement assistée par des logiciels tels que Protégé² ou des frameworks dédiés tels que OWLAPI³, permettant de définir l'ontologie en fonction des axiomes *OWL 2*.

Différentes syntaxes recommandées par le W3C peuvent être utilisées afin de sériali-

2. <http://protege.stanford.edu/>

3. <http://owlapi.sourceforge.net/>

TABLE A.7 – Axiomes OWL 2

Axiome	Syntaxe	Exemple
subClassOf	$C_1 \sqsubseteq C_2$	$Humain \sqsubseteq Animal \sqcap Bipede$
equivalentClass	$C_1 \equiv C_2$	$Homme \equiv Humain \sqcap Male$
subPropertyOf	$R_1 \sqsubseteq R_2$	$possedeFille \sqsubseteq possedeEnfant$
equivalentProperty	$R_1 \equiv R_2$	$Cout \equiv Prix$
disjointWith	$C_1 \sqsubseteq \neg C_2$	$Male \sqsubseteq \neg Femelle$
sameAs	$\{a_1\} \equiv \{a_2\}$	$\{Paris\} \equiv \{Ville_Lumire\}$
differentFrom	$\{a_1\} \sqsubseteq \neg \{a_2\}$	$\{marie\} \sqsubseteq \neg \{josé\}$
transitiveProperty	R rôle transitif	$estAncetreDe$ rôle transitif
functionalProperty	$T \sqsubseteq (\leq 1R)$	$T \sqsubseteq (\leq 1estMereDe)$
inverseFunctionalProperty	$T \sqsubseteq (\leq 1R^-)$	$T \sqsubseteq (\leq 1aPourMere^-)$
symmetricProperty	$R \equiv R^-$	$estMarieAvec \equiv estMarieAvec$

ser une ontologie OWL, notamment en XML/RDF⁴, Turtle⁵, NTriples⁶ ou NQuads⁷. La plupart des triples stores prennent en compte ces formats.

A.2/ LES RAISONNEMENTS LOGIQUES

Les systèmes de représentation de connaissances basés sur les logiques de description permettent la réalisation de raisonnements. Le raisonnement est un processus qui permet de produire de nouvelles connaissances à partir des connaissances existantes, ou de vérifier des faits (connaissances) existants. Différents types de raisonnements peuvent être effectués sur un base de connaissances donnée. Les différents types de raisonnements sont nommés inférences logiques.

A.2.1/ LES INFÉRENCES

Parmi les différentes inférences logiques réalisables sur une base de connaissances, nous distinguons les trois cas suivants :

1. Les inférences qui permettent de vérifier la cohérence de la TBox.
2. Les inférences qui permettent de vérifier la consistance de la ABox.
3. Les inférences qui permettent de découvrir des connaissances implicites à partir des connaissances explicites contenues dans la base de connaissances.

La découverte de nouvelles connaissances correspond à :

1. la découverte de propriétés d'individus.
2. la découverte de relations entre individus,

4. <https://www.w3.org/TR/rdf-syntax-grammar/>

5. <https://www.w3.org/TR/turtle/>

6. <https://www.w3.org/TR/n-triples/>

7. <https://www.w3.org/TR/n-quads/>

3. la découverte de relations de subsomption ou d'équivalence entre concepts permettant de produire la hiérarchie des concepts (i.e. hiérarchisation).
4. la découverte de relations entre concepts et instances. Notamment la classification au plus spécifique concept pour chaque individu (i.e. réalisation).

Les notions de **hiérarchisation** et de **réalisation** sont employées avec le même sens dans l'architecture SHMC, présentée dans les chapitres 3 et 4 de ce manuscrit.

A.2.2/ APPROCHES DE RAISONNEMENT

Différents algorithmes ont été développés pour résoudre le problème d'inférence. Les algorithmes sont généralement divisés en deux catégories :

Le premier groupe contient des algorithmes appelés algorithmes structurels. Ces algorithmes comparent la structure syntaxique des concepts pour résoudre le problème de subsomption de concept dans quelques logiques de description primitives. Ces algorithmes, ne sont toutefois pas applicables pour les logiques de description plus complexes, notamment la négation et la disjonction.

Le deuxième groupe contient des algorithmes nommés algorithmes de tableaux. Les algorithmes de tableaux ont été donnés la première fois par Schmidt-Schauß et al. (1991), et sont aujourd'hui l'outil principal de résolution des problèmes de satisfaisabilité et de subsomption de concepts dans les logiques de description.

Les sections suivantes décrivent brièvement ces deux catégories d'algorithmes.

A.2.2.1/ RAISONNEMENT BASÉ SUR LA COMPARAISON STRUCTURELLE

Les algorithmes de raisonnement basé sur la comparaison structurelle ne sont applicables que sur des logiques primitives de type \mathcal{FL}^- (équivalent à \mathcal{ALC} sans la négation de concepts atomiques). Les algorithmes du calcul de subsomption sont basés sur la comparaison structurelle entre les expressions de concepts. L'idée de cette approche est que si deux expressions de concept sont faites de sous-expressions, alors elles peuvent être comparées séparément en comparant une sous-expression d'un concept avec toutes celles de l'autre. Afin de vérifier la subsomption dans les logiques \mathcal{FL}^- , l'algorithme s'exécute en deux phases. Premièrement, les concepts sont réécrits dans une forme normale (i.e. déplier les concepts et factoriser les rôles), puis leurs structures sont comparées :

- Toutes les conjonctions emboîtées sont égalisées : $A \sqcap (B \sqcap C) \Leftrightarrow A \sqcap B \sqcap C$. Toutes les conjonctions de quantifications universelles sont factorisées : $\forall R.C \sqcap \forall R.D \Leftrightarrow \forall(C \sqcap D)$. Les concepts réécrits sont logiquement équivalents avec les précédents, donc la transformation préserve la subsomption.
- Soient $C = C_1 \sqcap C_2 \sqcap \dots \sqcap C_m$ et $D = D_1 \sqcap D_2 \sqcap \dots \sqcap D_n$, alors D subsume C si et seulement si pour chaque D_i , il existe un C_j tel que :
 1. Si D_i est un concept atomique, ou est bien de la forme $\exists R$, alors $D_i = C_j$.
 2. Si D_i est un concept de forme $\forall R.D'$, $C_j = \forall R.C'$ (le même rôle atomique R), alors $C' \sqsubseteq D'$.

A.2.2.2/ RAISONNEMENT BASÉ SUR L'ALGORITHME DE TABLEAUX

Cette technique de raisonnement est basée sur les calculs de tableaux pour la logique des prédictats du premier-ordre. Les structures de tableaux obtenues en raisonnant avec un langage donné des logiques de description sont soigneusement analysées, et les vérifications redondantes dans les tableaux sont éliminées afin de donner une limite supérieure stricte sur la complexité de la méthode.

L'idée principale du calcul de tableau est de vérifier si une formule donnée F est une conséquence logique d'une théorie donnée T . On essaye de construire, en utilisant des règles de propagation, le modèle le plus générique de T où F est faux. Si le modèle est construit avec succès, alors la réponse est *NON*, i.e. F n'est pas une conséquence logique de T . Si la construction du modèle échoue, alors la réponse est *OUI*, i.e. il n'existe pas un modèle de T avec F faux, donc F est réellement une conséquence logique de T . Les règles de propagation proviennent directement de la sémantique des constructeurs.

D'une manière générale, l'algorithme de tableaux appliqué dans une logique de description essaye de prouver la satisfaisabilité d'une expression de concept D en démontrant l'existence d'une interprétation \mathcal{I} dans laquelle $D^{\mathcal{I}} \neq \emptyset$. Cette technique de raisonnement permet aujourd'hui de proposer des algorithmes de décision de satisfaisabilité et de sub-somption qui sont corrects et complets, y compris pour les langages les plus expressifs des logiques de description. En revanche, la complexité de l'inférence diffère selon l'expressivité du langage.

A.2.3/ COMPLEXITÉ DE L'INFÉRENCE

Le complexité du raisonnement et par conséquent, le temps de calcul et la quantité de mémoire nécessaires à sa réalisation, dépendent de la taille de la base de connaissances (i.e. quantité de données) ainsi que de son niveau d'expressivité.

Le site <http://www.cs.man.ac.uk/~ezolin/dl/> présente un aperçu non exhaustif de la complexité du raisonnement au niveau terminologique et assertional en fonction de l'expressivité de la logique utilisée Baader (2003), i.e. de l'ensemble des constructeurs qui définissent la logique.

Nous présentons ci-dessous un aperçu des différentes classes de complexité ainsi qu'un tableau associant la complexité à l'expressivité logique (A.8).

- **P** : la classe des problèmes de décision prenant en entrée un énoncé de problème et produisant en sortie une réponse positive ou négative (binaire) requiert un temps polynomial par rapport à la taille du problème pour obtenir une solution à l'aide une machine de Turing déterministe. On qualifie alors le problème de polynomial. Ce problème est de complexité $O(n^k)$ pour un k donné.
- **NP** : la classe des problèmes qui nécessitent un temps polynomial pour trouver une solution avec une machine de Turing non déterministe. Les calculs d'une machine de Turing déterministe forment une suite, tandis que les calculs d'une machine de Turing non déterministe forment un arbre, dans lequel chaque chemin correspond à une suite de calculs possibles.
- **PSpace** : la classe des problèmes de décision qui requièrent une quantité de mémoire polynomiale pour résoudre un problème avec une machine de Turing déterministe ou non déterministe.
- **ExpTime** : la classe des problèmes de décision solvables par une machine de

- Turing déterministe en un temps exponentiel par rapport à la taille du problème.
- **NExpTime** : la classe des problèmes de décisions solvables par une machine de Turing nondéterministe en un temps exponentiel par rapport à la taille du problème.

TABLE A.8 – Complexité du raisonnement en fonction de la logique de description et donc de son expressivité

Type d'inférence	Logique				
	<i>ALC</i>	<i>S</i>	<i>SH / SHIF</i>	<i>SHOIN</i>	<i>SROIQ</i>
Satisfaisabilité de concept	PSpace complet	PSpace complet	EXPTIME complet	NExpTime complet	NExpTime complet
Consistance de la ABox	PSpace complet	N/A	EXPTIME complet	NExpTime complet	NExpTime complet

A.2.4/ LES RAISONNEURS POUR OWL

Actuellement, il existe plusieurs raisonneurs libres, conçus pour raisonner sur les logiques de description et compatibles avec le standard du W3C, i.e. OWL/RDF(S). Parmi ceux-ci, on peut citer FaCT++⁸, Hermit⁹, RacerPro¹⁰, et Pellet¹¹. Certains raisonneurs ne peuvent raisonner qu'au niveau terminologique, alors que des moteurs comme Pellet et RacerPro permettent de raisonner aussi sur les instances de concepts. Nous les présentons ci-dessous :

- FaCT++ ("Fast Classification of Terminologies") est un raisonneur *OWL-DL* supportant un sous ensemble de *OWL 2 (SHOINQ(D))*. Il a été implémenté en C++ et utilise un algorithme de tableaux optimisé pour de meilleures performances.
- Hermit est un raisonneur basé sur l'algorithme hypertableaux. Il a une expressivité allant jusqu'à (*SHOINQ(D)*).
- RacerPro ("Renamed Abox and Concept Expression Reasoner") est un raisonneur utilisant les hypertableaux calculus optimisé pour la logique de description *SROIQ*. Il peut traiter des ontologies *OWL-Lite* et *OWL-DL*, mais ignore les types définis par l'utilisateur et les concepts énumérés.
- Pellet : est le premier raisonneur à supporter entièrement *OWL-DL*. Pellet est aujourd'hui compatible avec le langage *OWL 2* et intègre diverses techniques d'optimisation, y compris pour les nominaux, les requêtes conjonctives, etc. Pellet est codé en JAVA et fonctionne sur la base de tableaux.

8. <http://owl.man.ac.uk/factplusplus/>

9. <http://hermit-reasoner.com/>

10. <http://franz.com/agraph/racer/>

11. <http://clarkparsia.com/pellet/>

TABLE A.9 – Caractéristiques des raisonneurs

Raisonneur	FaCT++	Hermit	RacerPro	Pellet
Expressivité	$SROIQ(\mathcal{D})$	$SROIQ(\mathcal{D})$	$SHIQ$	$SROIQ(\mathcal{D})$
Règles	- N/A	+ SWRL	+ SWRL	+ SWRL
Langage	C++	JAVA	JAVA	JAVA
Méthode	tableau	hypertableau	tableau	tableau
Raisonnement sur <i>ABox</i>	+	+	+	+

B

LE PARADIGME MAPREDUCE

MapReduce est un paradigme de programmation distribué défini par Dean et al. (2008). L'objectif de ce paradigme est de tirer partie d'une montée en charge horizontale des capacités de calcul pour effectuer une tâche coûteuse en ressources (processus, espace de stockage, mémoire vive, réseau ...), i.e. par incrémentation du nombre de machines disponibles et division de cette tâche parmi les machines. Le paradigme permet alors d'effectuer des tâches sur des données de taille importante, réparties sur un système de fichier distribué commun à l'ensemble des machines disponibles. L'ensemble des machines forme un cluster de calcul distribué.

B.1/ MAPREDUCE ET ARCHITECTURE HADOOP

Les processus MapReduce sont des processus de traitement de données par lots (batch processing). Chaque tâche est divisée en lots appelés jobs (MapReduce). Hadoop¹ est une implémentation du paradigme MapReduce, développée en java. Il s'agit de l'implémentation la plus utilisée du paradigme. Hadoop définit deux composants essentiels pour répondre aux spécificités du paradigme. Le système de fichier distribué, HDFS (Hadoop Distributed File System), et un ensemble de librairies permettant l'exécution de tâches MapReduce. A partir de ces deux composants, de nombreux outils de traitement du Big Data se sont développés. La majorité de ces outils sont des "surcouches", qui fonctionnent en interne sur la base de tâches MapReduce et sur l'utilisation de HDFS.

La figure B.1 décrit comment le fonctionnement de l'architecture Hadoop lors du lancement de jobs MapReduce sur un cluster de machines.

Comme décrit ci-dessus, les données sont divisées arbitrairement en sections nommées *InputSplits* de taille fixe (64Mo par défaut). Chaque nœud du cluster traite de séquentiellement une partie des sections. Ce traitement est défini dans le Mapper. Les résultats intermédiaires des Mapper sont envoyés aux Reducers, qui agrègent ces résultats par clé et effectuent des traitements additionnels. La figure B.2 décrit le fonctionnement interne d'une tâche MapReduce.

1. <http://hadoop.apache.org/>

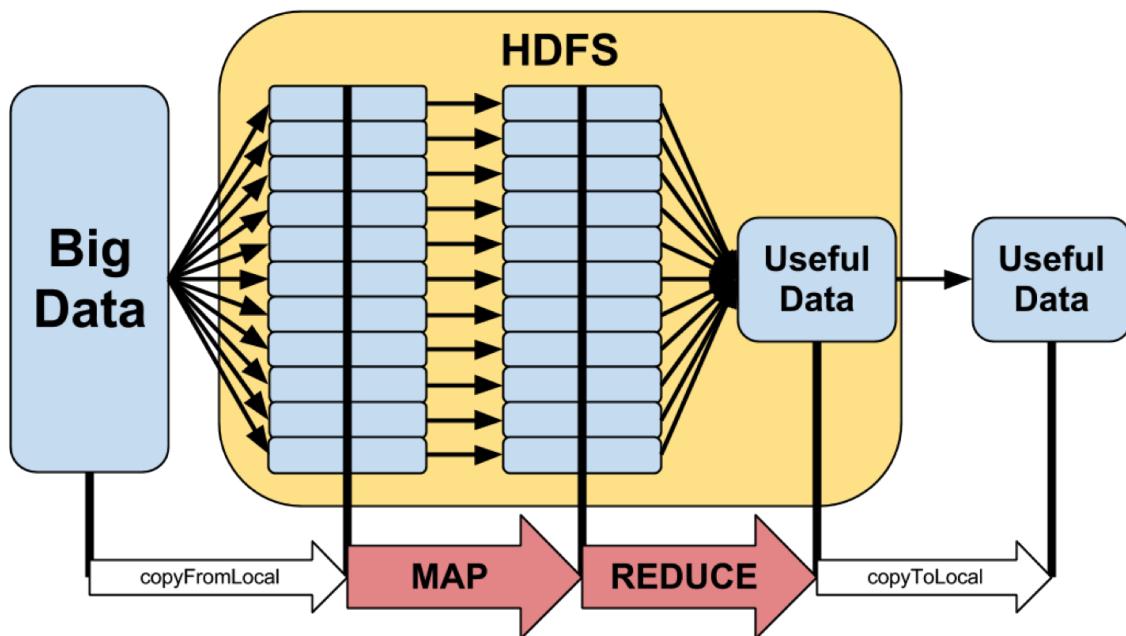
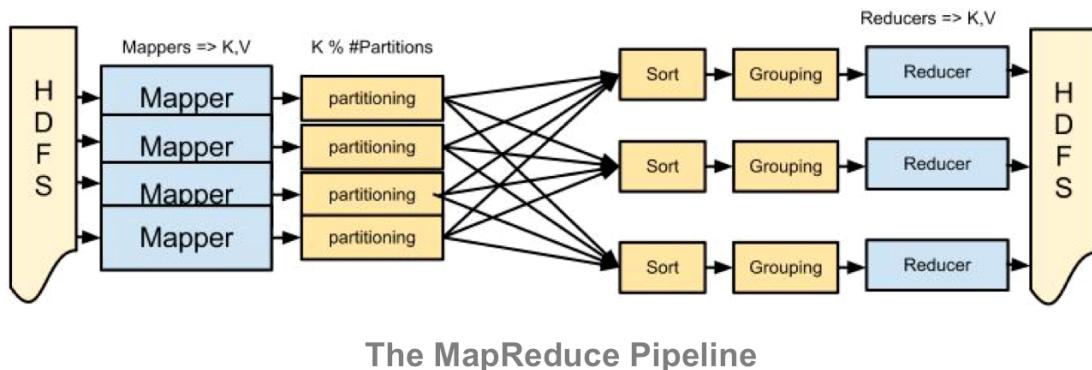


FIGURE B.1 – Fonctionnement Hadoop MapReduce



The MapReduce Pipeline

A mapper receives (Key, Value) & outputs (Key, Value)
A reducer receives (Key, Iterable[Value]) and outputs (Key, Value)
Partitioning / Sorting / Grouping provides the Iterable[Value] & Scaling

FIGURE B.2 – Principe d'une tâche MapReduce. Source : <http://www.glenneklockwood.com/data-intensive/hadoop/overview.html>

B.2/ IMPLÉMENTATION DE TÂCHES MAPREDUCE

Cette section décrit en pseudo-code l'implémentation de tâches (jobs) MapReduce. Un exemple minimaliste est d'abord présenté au travers de l'algorithme Word Count. Les tâches de construction de la matrice (Vectorisation) et de création des règles de classification (Résolution) à partir de la matrice sont ensuite décrrites. L'implémentation de création des relations hiérarchiques (Hiérarchisation étant similaire à l'algorithme de création des règles, il ne sera pas détaillé).

B.2.1/ WORD COUNT

L'algorithme Word Count est l'exemple minimal de tâche MapReduce. L'algorithme lit un ensemble de termes en entrée, regroupe les termes identiques et somme les occurrences de chaque terme.

Le mapper (fonction map) convertit les données initiales en un ensemble de tuples (clé,valeur) représentant une occurrence d'un terme, i.e. un tuple de la forme (terme,1)².

TABLE B.1 – Exemple Word Count, mapper

Entrée : Données initiales	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
Sortie : tuples	(BUS,1), (CAR,1), (BUS,1), (CAR,1), (TRAIN,1), (CAR,1), (BUS,1), (CAR,1), (TRAIN,1), (BUS,1), (TRAIN,1),(BUS,1), (BUS,1), (CAR,1), (CAR,1), (CAR,1), (BUS,1), (TRAIN,1)

Dans cet exemple on remarque que le mapper joue également un rôle de normalisation des données d'entrée (transformation en majuscules). Les traitements effectués sur les données d'entrée sont libres, et peuvent être effectués au niveau du mapper, ou avant l'envoi des valeurs intermédiaires entre le mapper et le reducer.

Le reducer (fonction reduce) lit la sortie des mapper, regroupe les tuples et émet un ensemble réduit de tuples transformés. Dans cet algorithme le reducer somme les occurrences de chaque terme.

TABLE B.2 – Exemple Word Count, reducer

Entrée : tuples	(BUS,1), (CAR,1), (BUS,1), (CAR,1), (TRAIN,1), (CAR,1), (BUS,1), (CAR,1), (TRAIN,1), (BUS,1), (TRAIN,1),(BUS,1), (BUS,1), (CAR,1), (CAR,1), (CAR,1), (BUS,1), (TRAIN,1)
Sortie : tuples (somme)	(BUS,7), (CAR,7), (TRAIN,4)

2. Source : <https://dzone.com/articles/word-count-hello-word-program-in-mapreduce>

Ci-dessous une implémentation du Mapper et du Reducer sont décrits.

Data : key = ID, value = texte

```
// key est un identifiant unique correspondant au segment de données (InputSplit). Value
est le contenu du segment (les termes).
```

Segmentation et normalisation des termes (value)

Liste \leftarrow termes normalisés

Function Map(key,value)

```
while Liste non vide do
```

```
    next  $\leftarrow$  liste.next()
    emit (key = next, value = 1)
```

```
end
```

Algorithme 1 : Mapper

Data : key = terme, value = listeOccurrences

total \leftarrow 0

Function Reduce(key,value)

```
while listeOccurrences non vide do
```

```
    next  $\leftarrow$  listeOccurrences.next()
```

```
    total  $\leftarrow$  total + next // next est systématiquement égal à 1 dans cet exemple.
```

```
Cependant une aggrégation des valeurs par clé peut être faite sur les données
intermédiaires entre le mapper et le reducer, auquel cas cette même fonction
reduce peut être utilisée. Dans l'implémentation Hadoop, le reducer
intermédiaire est un combiner.a
```

```
end
```

```
emit (key = terme, value = total)
```

Algorithme 2 : Reducer

a. <http://hadooptutorial.info/combiner-in-mapreduce/>

B.2.2/ VECTORISATION (ALGORITHME STRIPES)

L'implémentation de l'approche stripes décrite dans le chapitre 3 correspond au pseudo-code ci-dessous.

Dans la méthode map, les cooccurrences sont d'abord regroupées pour chaque terme distinct traité par le mapper. Le terme et ses cooccurrences sont alors amis au Reducer sous la forme d'un couple clé/valeur, i.e. (*Term t, Stripe s*) dans l'algorithme. Chaque

valeur s (stripe) contient la liste des termes cooccurents du terme t .

```

Data : key = ID, value = texte
// key est un identifiant unique correspondant au segment de données (InputSplit). Value
// est le contenu du segment (les termes).
Segmentation et normalisation des termes (value)
Liste ← termes normalisés
Function Map(key,value)
    for i=0, i < Liste.size, i++ do
        terme ← Liste[i]
        stripe ← HashMap vide
        for j=0, j< Liste.size, j++ do
            valeurPrecedente ← stripe.get(Liste[j])
            if valeurPrecedente not null then
                | stripe.put(Liste[j], valeurPrecedente + 1)
            else
                | stripe.put(Liste[j], 1)
            end
        end
        emit (key = terme, value = stripe)
    end

```

Algorithme 3 : Mapper

```

Data : key = terme, value = stripe
incrementingMap ← HashMap vide
Function Reduce(key,value)
    incrementingMap.clear()
    foreach HashMap stripe ∈ value do
        | addAllOccurrences(stripe)
    end
    emit (key = key, value = incrementingMap)
Function addAllOccurrences(HashMap stripe)
    Set<key> clefs ← stripe.keySet()
    foreach key cle ∈ clefs do
        countFromStripe ← stripe.get(cle)
        if incrementingMap.containsKey(cle) then
            | totalCount ← incrementingMap.get(cle)
            totalCount ← totalCount + countFromStripe
        else
            | incrementingMap.put(cle, countFromStripe)
        end
    end

```

Algorithme 4 : Reducer

B.2.3/ RÉSOLUTION (RÈGLES)

Dans le mapper, une portion (InputSplit) de la matrice de cooccurrence est lue de façon séquentielle, i.e. une liste de couples de termes associés à leur proportion conditionnelle $P_C(term_i|term_j)$. La fonction ID génère un identifiant unique servant à regrouper l'ensemble des valeurs pour une même clé construite à partir des deux termes

$<term_i, term_j>$ (voir algorithme Stripes).

Data : key = $ID(term_i, term_j)$, value = $P_C(term_i|term_j)$

Function Map(key,value)

```

if  $\alpha < P_C(term_i|term_j) \& term_i \subseteq Label$  then
    emit (key =  $ID(term_i hasAlphaTerm)$ , value =  $term_j$ )
else
    if  $\beta < P_C(term_i|term_j) \& term_i \subseteq Label$  then
        emit (key =  $ID(term_i hasBetaTerm)$ , value =  $term_j$ )
    end
end

```

Algorithme 5 : Mapper

Dans le reducer, les tuples émis par les différents mappers sont regroupés par clé, composée d'un identifiant unique construit à partir du label (tête de la règle) et du type de relation. La valeur regroupe l'ensemble des relations hasAlpha|hasBeta pour ce label.

Data : key = $ID(term_i hasAlphaTerm/hasBetaTerm)$, value = $Liste<term_j>$

```

total <- 0 Tri par valeur décroissante de  $Liste<term_j>$  // Le nombre de relations de type
beta est limité par le seuil de classement termes (n). Le tri est nécessaire pour
appliquer cette restriction.

```

Function Reduce(key, value)

```

while Liste non vide do
    lire prochain élément de la liste
    if ID contient hasAlphaTerm then
        emit (key =  $term_i hasAlphaTerm$ , value =  $term_j$ )
    else
        if total < n then
            total <- total + 1
            emit (key =  $term_i hasBetaTerm$ , value =  $term_j$ )
        end
    end
end

```

Algorithme 6 : Reducer

C

PRODUCTION DES DOCUMENTALISTES

Cette annexe présente le travail des documentalistes de l'entreprise, qui comprend notamment un processus de veille d'information, et un processus rédactionnel d'articles synthétiques. La description de ces processus est basée sur les travaux de thèse de Werner (2015), qui a définit cette description auprès des documentalistes de l'entreprise Actualis.

Les documentalistes de l'entreprise Actualis sont des experts du domaine de la veille informationnelle et stratégique. Au sein de l'entreprise ils sont en charge de la veille et la production d'articles de synthèse. Ils ont grandement participé à la création des vocabulaires contrôlés, présentés dans le chapitre 7 lors de la mise en place du système de recommandation d'articles Werner (2015). Ce système repose en partie sur le savoir-faire métier de ces experts.

C.1/ MÉTHODE DE TRAVAIL DES DOCUMENTALISTES

Cette annexe présente quelques aspects du travail de documentaliste, et s'intéresse en particulier aux points concernés par la mise en place du nouvel outil de veille (chapitre 7) dans l'entreprise. Les documentalistes rédigent les articles qui sont par la suite proposés aux clients. Leur travail consiste à :

1. Veiller des sources d'informations.
2. Sélectionner les articles intéressants proposés par ces sources.
3. Extraire les éléments pertinents des articles sources. Cela peut inclure des tâches supplémentaires comme : recouper les informations entre différentes sources si différentes sources sont disponibles. Approfondir la recherche dans certains cas où l'information partielle ou trop limitée.
4. Synthétiser cette information sous la forme d'articles concis proposés aux clients.

Les sections suivantes détaillent les quatre points ci-dessus.

C.1.1/ SOURCES ET SUPPORTS DE L'INFORMATION ÉCONOMIQUE

Chaque documentaliste est en charge de la veille d'un espace géographique des sources d'information correspondantes. Une veille performante et de qualité doit prendre en compte la multiplicité des sources d'information ainsi que leur évolution. Les supports utilisés par les documentalistes de l'entreprise sont les suivants :

- La presse papier (presse quotidienne régionale, nationale, presse professionnelle, presse hebdomadaire)
- La presse web
- Le web en général, qu'il s'agisse de sites d'actualités ou de sites vitrines
- Les outils de relations presse, i.e. les dossiers de presse et les communiqués de presse
- La transmission orale de l'information

Ces différentes sources transmettent chaque jour des quantités importantes d'information qu'il faut sélectionner, filtrer, pour ne garder que les plus pertinentes.

C.1.2/ SÉLECTION DES INFORMATIONS

Le processus de sélection d'une information dépend en partie de sa source. Une source traitant uniquement d'informations économiques ne sera pas traitée de la même façon qu'une source proposant de l'information généraliste. Mais les deux types sont veillés. L'analyse de sources spécialisées, plus proche du domaine de travail de l'entreprise sera ainsi plus rapide à analyser qu'une source généraliste. Les informations proposées par les différentes sources sont parcourues à la recherche de mots clés. Une partie de ce travail est fait de façon automatisée par les logiciels de veille déjà en place, qui font un premier filtrage automatique des sources d'informations numériques à l'aide de règles définies par les documentalistes. Les règles sont des combinaisons plus ou moins élaborées de mots clés, par exemple : *entreprise* ou *société* et *Nord* ou *Pas-de-Calais* sauf chasse. En ce qui concerne les sources imprimées, ou orales, par contre, le travail est intégralement fait par le documentaliste. Lorsque le documentaliste recherche des informations sur un support papier, il dispose, pour permettre son analyse, d'un titre, d'un chapô¹ et du texte intégral. L'analyse est donc rapide puisque le nombre de mots-clés à sa disposition est important. La sélection s'opère alors en deux temps :

- lecture diagonale de l'ensemble du texte
- sélection ou rejet

Si le documentaliste est face à un support web, il ne dispose dans la majorité des cas que d'un titre et d'un chapô. La sélection est alors plus complexe. Elle s'opère en quatre temps :

- analyse du titre
- clic sur le titre
- analyse du texte
- sélection ou rejet du texte

L'analyse des contenus s'effectue en outre selon plusieurs niveaux de lecture. Les documentalistes cherchent en premier lieu des termes typiquement liés à l'univers économique, comme *entreprise*, *société*, *PME*, *investissement*, *zone d'activité*, ou encore des mots faisant référence à des événements (i.e. thèmes) ou des noms des principaux secteurs d'activités économiques.

Un vocabulaire plus lointain est aussi pris en compte. L'expérience montre que ces termes pourraient générer des réponses pertinentes. Ainsi, des termes comme *pôle*, *conseil communautaire*, *conseil municipal*, *projet* ou encore *activité*, génèrent une attention particulière de la part des documentalistes. Combinés à un mot-clé économique, ces termes provoquent une analyse complète du contenu.

1. Texte court coiffant un article, permettant d'amener le lecteur à entrer dans l'article.

Ces mots clés spécifiques au domaine permettent d'identifier la nature de l'information et de la qualifier de façon concise et pertinente. La détection de mots clés pertinents et la qualification de l'information est une des motivations de ces travaux de thèse, notamment de l'architecture SHMC (chapitres 2 à 4).

Un mot-clé seul, même appartenant à l'univers de l'économie, ne suffit pas à provoquer la sélection de l'information. Il induit, tout au plus, une lecture un peu plus approfondie du contenu. Pour que s'opère la sélection, il faut que le contenu combine plusieurs termes pertinents. Le processus de sélection est donc basé sur une recherche booléenne. L'analyse booléenne n'est pas la même avec tous les supports. En effet, les niveaux de lecture étant différents en fonction du support, l'analyse l'est également. Ainsi, un support à vocation généraliste avec une couverture nationale (comme *Le Monde* ou *Le Figaro*) va induire une première recherche avec des mots-clés relevant de la géolocalisation (exemple : *Nord-Pas-de-Calais*), puis dans un deuxième temps, une recherche avec des mots-clés typiquement liés à l'économie (*entreprise* ou *investissement*).

Un support avec une couverture nationale, mais à vocation économique (*La Tribune* ou *Les Echos*) va plus simplement induire une recherche géolocalisée. Un support local, mais généraliste (comme *La Voix du Nord*, *Ouest France*, *Le Parisien*, etc.) n'imposera pas de géolocalisation puisqu'il est local et va donc induire une recherche de mots-clés typiquement liés à l'économie.

Un support local économique (*La Gazette Nord-Pas-de-Calais*, *Bref Rhône-Alpes* ou *La Lettre API*) implique une simple recherche de vocabulaire économique, puis une hiérarchisation de l'information en fonction de sa valeur ajoutée pour les clients. Il est donc beaucoup plus aisé d'analyser un support à la fois local et économique puisqu'il réunit tous les paramètres de la requête et effectue en quelque sorte un pré-tri des informations. Il est en revanche beaucoup plus complexe d'analyser un support à la fois national et généraliste, puisqu'il implique d'opérer plusieurs requêtes (vérifier plusieurs critères), à la fois géographiques et économiques.

C.1.3/ EXTRACTION DES INFORMATIONS

L'objectif une fois une information sélectionnée est d'en rédiger une synthèse. Pour cela il faut extraire de la source d'information, par exemple un article de journal en ligne, les éléments à conserver. Pour cela, les principes des *5W* ou en français *QQOQCP* est utilisé. Les éléments conservés sont ceux qui répondent aux questions : *Qui ? Quoi ? Où ? Quand ? Comment ? Pourquoi ?*

Des suppléments d'informations par rapport à ce qui est présent dans la source sont parfois ajoutés. Il est possible que le documentaliste puisse répondre à ce qui est nommé le *2^{ème} Quoi*. Une deuxième information trouvée sur un autre support montre que l'entreprise en question dans l'article réalise simultanément une deuxième action. Il faut donc **recouper** les informations. Si l'entreprise a fait l'objet d'un article récemment publié, un rappel de ce qui a été dit dans le précédent article est introduit afin de développer le *Qui*. Dans certains cas il peut tout simplement s'agir de renvoyer l'utilisateur vers un document, via un lien hypertexte par exemple. Enfin, l'article source est parfois incomplet, imparfait, voire difficile à comprendre, mais il a tout de même été sélectionné comme pertinent à la phase précédente. Il est donc nécessaire de trouver plus d'informations en **recroisant** différentes sources, voire parfois en contactant directement les protagonistes.

C.1.4/ RÉDACTION DE LA SYNTHÈSE

La synthèse débute la plupart du temps par une rapide description du *Qui* (i.e. nom, activité, éventuels éléments historiques), et généralement du *Où*. Ensuite, les rédacteurs introduisent le *Quoi* (i.e. action, raison d'être de la synthèse). Puis, ils détaillent le *Quoi* en apportant la réponse aux questions *Quand*, *Comment*, *Pourquoi* et *Où* (si l'action ne se déroule pas dans le même lieu que celui cité en introduction). Enfin, ils ajoutent très souvent des éléments complémentaires. La plupart du temps concernant le *Qui* mais sans rapport avec le *Quoi* (le chiffre d'affaires, l'effectif de l'entreprise, etc.).

Le titre est introduit par un ou plusieurs termes définissant généralement les secteurs d'activités concernés par l'information. Un des objectifs de l'architecture SHMC est d'améliorer la qualification de l'information que représente ces termes. Ces termes sont suivis d'une phrase répondant aux questions : *Qui*, *Quoi*, *Où*, *Quand* et parfois *Pourquoi*.

Spécifier le secteur ainsi que beaucoup d'informations dans le titre a pour objectif de permettre à l'utilisateur de savoir rapidement si l'article correspond à ses attentes.

Le *Qui* prend à la fois en compte, l'entreprise en elle-même (sa taille, son importance, son chiffre d'affaires) et le secteur économique concerné par son activité. Le *Quoi*, concerne la raison d'être de l'article, c'est-à-dire, généralement l'événement économique (i.e. le thème). La pertinence des différents secteurs d'activité et de certains événements économiques intéressent certains lecteurs plus que d'autres.

D

IMPLEMENTATION DE L'ANALYSEUR SYNTAXIQUE

Pour construire le modèle de classification, le processus SHMC extrait les termes potentiellement pertinents des données, en se basant sur des méthodes statistiques.

Dans la littérature, cette phase correspond à la sélection de caractéristiques (features), ici appliquée l'extraction de termes dans un corpus de textes. Le domaine de la Recherche d'Information a vu de nombreuses propositions d'améliorations de cette phase, notamment pour les systèmes d'indexation basés sur les mots clés. L'étape d'extraction de termes dans un système de classification a un impact non négligeable sur la qualité des classifications effectuées en sortie du système Kboubi et al. (2010).

Il est donc primordial d'optimiser et d'évaluer cette phase du processus. La qualité intrinsèque des labels est complexe à évaluer : la sélection des labels (attributs de classe) est dépendante de la configuration du processus, et leur qualité peut être considérée comme subjective.

Les sections suivantes décrivent les choix effectués pour implémenter la phase d'extraction de termes du processus SHMC.

D.1/ OBJECTIFS

Le processus SHMC effectue une sélection des termes en fonction de plusieurs paramètres. Ces paramètres sont décrits dans le tableau 3.5, et correspondent aux paramètres des fonctions de la librairie Mahout utilisées dans le processus (notamment les fonctions SequenceFilesFromLuceneStorageDriver et SparseVectorsFromSequenceFiles¹). La première consiste à convertir l'index Lucene issu de Solr en fichiers de séquence. La seconde est la fonction qui utilise l'ensemble des paramètres déterminants pour la sélection des labels (3.5).

La première étape afin d'améliorer la qualité intrinsèque des termes et des labels (attributs de classe) est de supprimer autant que possible le bruit résultant de la sélection statistique effectuée par le système.

- La sélection statistique des termes doit prendre en compte différentes mesures pour réduire le bruit en fonction de la pondération des termes par fréquence d'apparition.

1. <https://mahout.apache.org/users/basics/creating-vectors-from-text.html>

partition. Les termes extraits de basse et haute fréquence sont souvent indésirables.

- Quelque soit le jeu de données utilisé, il est normal de constater un bruitage de termes non pertinents. Les différentes spécificités des jeux de données sont susceptibles d'introduire un bruit supplémentaire : les jeux de données peuvent ne pas être adaptés aux traitements effectués par le système. Dans l'évaluation en contexte non supervisé (chapitres 3 et 4), le jeu de données utilisé est issu d'archives du site Wikipedia (dumps d'articles de la version française).

Ce jeu de données concerne des connaissances générales, et n'introduit pas de bruit lié à une quelconque spécificité du domaine. En revanche, nous avons pu observer que du bruit était introduit par le simple fait que ce jeu de données est issu de documents web. Les documents web ont la particularité d'apparaître sous un format (semi)structuré, en opposition à des données textuelles, et doivent donc être formatés (parsés). Cette étape engendre naturellement du bruit lorsque la source n'est pas parfaitement constituée (balises html, css, éléments de page récurrents, instructions de code ...). Ce bruit doit être traité.

- Enfin, l'utilisation d'un jeu de données français crée du bruit supplémentaire. Les causes peuvent être la gestion des caractères spéciaux, des élisions, de cas particuliers du dictionnaire, et plus généralement de la syntaxe française. Il est donc nécessaire d'appliquer des traitements spécifiques au français. Le positionnement par rapport à l'état de l'art est plus difficile, puisque la majorité des travaux se concentrent sur l'analyse de documents anglais. Naturellement, les ressources-/méthodes disponibles afin de traiter des documents en français sont plus rares et moins complètes que celles disponibles pour l'anglais Vivaldi et al. (2007) Dubremetz (2013). WordnetMiller (1995) par exemple, est une des ressources lexicales les plus utilisées dans la littérature. Or, cette ressource est incomplète au niveau de sa traduction française. Des travaux récents tels que WoneFPradet et al. (2014) tentent de palier cette incomplétude, mais ne sont pas intégrés à la version standard de Wordnet. Pour d'autres ressources telles que VerbNet Schuler (2005), ou FrameNetBaker et al. (1998), il n'existe pas de traduction standard, bien que des traductions françaises existent Mouton et al. (2009) Hadouche et al. (2010) Danlos et al. (2015).

Lors de l'étape d'indexation, un analyseur syntaxique basé sur la bibliothèque Lucene d'Apache² définit l'ensemble des traitements appliqués aux données brutes, pour effectuer l'extraction de termes. L'analyseur tronque le texte (tokenisation) et applique un ensemble de filtres sur les mots clés extraits. Les différents types de filtres sont communément utilisés dans la littérature dans les applications de traitement automatique du langage : normalisation de longueur (mots-clés trop courts/longs), suppression des mots vides, stemmer spécifique à une langue, etc. Paramétrier l'analyseur de façon efficiente est donc un point important pour corriger le bruit observé.

D'après nos observations préliminaires, l'analyseur syntaxique du français par défaut (classe issue de la bibliothèque Lucene, i.e. "org.apache.lucene.analyzers.FrenchAnalyser"), des adaptations sont nécessaires au niveau de l'analyseur afin d'améliorer les filtres. En outre, la liste des mots vides (stopwords) doit être mise à jour, et l'utilisation du stemming (racinisation) est remise en question pour notre cas d'utilisation.

2. Tous les analyseurs de la bibliothèque Lucene dérivent de la classe http://lucene.apache.org/core/4_3_0/core/org/apache/lucene/analysis/Analyzer.html

Un analyseur personnalisé a été implémenté afin d'effectuer des traitements adaptés au problème et aux jeux de données à analyser, tels que ceux utilisés pour les évaluations présentées dans les chapitres 3 et 4. Cette implémentation permet de définir et de configurer les filtres qui seront appliqués aux termes extraits de jeu de données.

Les optimisations apportées à l'analyseur syntaxique doivent prendre en compte les spécificités de la tâche à effectuer, notamment l'objectif du processus (construction d'un modèle de classification), et les caractéristiques du jeu de données. Les sections suivantes décrivent les différentes optimisations apportées à l'analyseur syntaxique depuis les tests préliminaires. La plupart des modifications apportées sont basées sur des observations empiriques, et n'ont pas fait l'objet d'une évaluation poussée. Il serait intéressant d'évaluer la qualité des termes en fonction de différents analyseurs syntaxiques, i.e. de différents traitements.

D.2/ TOKENISATION

La tokenisation est le processus de décomposition d'un flux de texte en mots, phrases ou entités appelés tokens. Ce traitement est généralement le premier dans la chaîne, ainsi les traitements suivants ne considèrent les tokens plutôt que le texte dans son ensemble. Un tokenizer de la bibliothèque Lucene³ est utilisé pour effectuer la tokenisation. Ce tokenizer est basé sur l'algorithme de segmentation de texte unicode⁴.

D.3/ STEMMING

Porter (2001) définit le stemming (racinisation) comme une des composantes d'un système d'indexation/de recherche d'information. Le stemming est décrit comme complexe, et étroitement lié aux autres composantes de traitement du langage. Il n'existe pas de solution appropriée à tous les systèmes : le choix des traitements effectués dépend des objectifs du système. La question de l'utilisation d'un algorithme de stemming dans notre cadre applicatif se pose.

Nos premières observations montraient une faible performance du stemmer standard pour la tâche souhaitée, à savoir regrouper les termes issus d'une même racine. Le stemmer standard de la bibliothèque Lucene pour le Français, basé sur l'algorithme SnowballAgichtein et al. (2000). L'algorithme de stemming étant basé exclusivement sur des règles syntaxiques, et non sur un thésaurus/dictionnaire, la racinisation est limitée. L'impact du stemmer sur le nombre de termes générés est significatif (de l'ordre de 1% à 10%, suivant le jeu de données), mais nous n'avons pas pu juger de l'aspect positif ou négatif de cette variation sur la qualité des termes. L'algorithme tend à rendre les termes extraits plus confus en effectuant une sur-racinisation (certains termes ne peuvent plus être identifiés clairement). Un algorithme de stemming moins agressif basé sur Savoy (2006) est implémenté dans la bibliothèque Lucene⁵. Les cas de sur-racinisation sont amoindris en comparaison de l'algorithme Snowball. Les bénéfices de l'utilisation de cet

3. https://lucene.apache.org/core/4_4_0/analyzers-common/org/apache/lucene/analysis/standard/UAX29URLEmailTokenizer.html

4. <http://unicode.org/reports/tr29/>

5. "org.apache.lucene.analysis.fr.FrenchLightStemmer"

algorithme sont détaillés dans Perret (2005). Dubremetz (2013) utilise le même algorithme de stemming, pour la reconnaissance de chiasmes dans des textes français. Leurs résultats montrent que l'utilisation du stemmer n'améliore pas le nombre de chiasmes retrouvés. L'algorithme est trop agressif et tend à sur-raciner, de telle sorte que deux termes issus de la même racine ne sont pas rapprochés. Par la suite nous avons considéré que l'utilisation d'un stemmer n'était pas adapté à notre cas d'utilisation.

D.4/ SUPPRESSION DES MOTS-VIDES

Plusieurs listes de mots-vides libres sont disponibles et ont été concaténées afin de créer une liste plus complète Savoy (1999) Perret (2005)⁶. D'autres mots vides ont été ajoutés manuellement à la liste d'après des observations faites sur les résultats.

La liste complète comporte près de 500 mots-clés. Les données utilisées pour les tests sont génériques, la liste des mots-clés n'est donc pas spécifique à un domaine particulier.

D.4.1/ FLEXIONS, SYNONYMES

Un système de recherche d'information basé uniquement sur des mots clés donne des résultats différents pour deux termes, sans prendre en compte leurs relations sémantiques.

Les relations sémantiques entre deux termes sont variées. Jiang et al. (1997) définit une liste non exhaustive des différentes catégories de relations. Les relations les plus communément citées sont les relations hiérarchiques (relations de type "IS-A", hyponymie/hyperonymie, ensemble/partie, etc.), associatives (cause-effet), et d'équivalence (synonymie). Nazarenko et al. (2009) distingue également la relation morphologique (flexion/lemme, exemple : drapeau, drapeaux), que l'on peut regrouper avec les relations d'équivalence.

L'analyseur syntaxique permet d'utiliser un dictionnaire de synonymes afin de lier différentes entrées ayant le même sens vers un seul et même terme dans l'index final. Cette fonctionnalité est utile pour les systèmes de recherche d'information pour deux raisons :

- Améliorer l'indexation des items pour la recherche
Si le système inclut une gestion de la synonymie, une requête portant sur un terme peut retourner des résultats sur son/ses synonyme. Cette fonctionnalité est souvent appliquée par extension de requête : lorsqu'une requête est reçue, les termes synonymes aux termes de la requête sont ajoutés à celle-ci. On se rapproche ainsi de la recherche sémantique : la gestion des relations de synonymie est un premier pas vers une recherche basée sur la sémantique. Cette observation s'applique également à la gestion des relations morphologiques des termes, i.e. les relations entre les lemmes⁷ et leurs flexions⁸.
- Réduire la taille de l'index : la taille de l'index inversé est fonction du nombre de termes qu'il contient. Par conséquent la diminution du nombre de termes total par

6. <http://snowball.tartarus.org/algorithms/french/stop.txt>

7. Le lemme, ou lexie ou item lexical, est l'unité autonome constituante du lexique d'une langue. C'est la forme d'un mot la plus simple utilisée comme entrée dans les dictionnaires. En français, il s'agit de la forme canonique, i.e. l'infinitif pour les verbes, et le masculin singulier pour un adjectif, etc

8. La forme fléchie d'un mot permet de représenter une fonction grammaticale (conjugaison), un nombre (pluriel/singulier), etc

affiliation de termes "équivalents", d'un point de vue sémantique, réduit la dimension de l'index.

D.4.2/ RESSOURCE UTILISÉE

Dicollecte est un effort collaboratif et open-source de développement de ressources linguistiques adaptées au français, à destination des moteurs de recherche et logiciels de traitement de texte libres. Un lexique des lemmes et de leurs formes fléchies est disponible. Le lexique est mis à jour régulièrement, et compte près de 500000 relations lemme-flexion.

Des cas particuliers sont à prendre en compte. Le lexique Dicollecte comprend parfois plusieurs lemmes possibles pour une même flexion. Exemple : le terme télescope peut à la fois représenter l'objet (nom commun), ou une déclinaison du verbe télescopier. Ce cas particulier représente un nombre important d'entrées du lexique (entre 4 et 5%).

Le lexique de Dicollecte est utilisé pour construire le dictionnaire de synonymes, qui est intégré à l'analyseur syntaxique.

Une autre ressource disponible est un thesaurus qui référence des relations de synonymie. Il est important de noter qu'une relation de synonymie entre deux termes différents est souvent dépendante du contexte (deux termes peuvent être synonymes dans un contexte mais pas dans un autre), et n'a pas toujours la même intensité. Exemple tiré du thesaurus de Dicollecte : le terme porte peut être affilié d'une part au sens de barrière, portique, clôture, mais aussi au sens de solution, exutoire, dans un contexte approprié. L'utilisation de ces synonymes dans un système d'indexation peut donc avoir un impact négatif sur la précision des résultats.

Modéliser les relations de synonymie au niveau des termes est un problème complexe, demandant des traitements particuliers (étude du contexte et/ou de la syntaxe). Le thesaurus de synonymes n'est pas intégré à la solution actuelle, uniquement l'intégration des relations morphologiques est prise en compte dans le processus.

D.5/ FILTRES REGEX

Le bruit engendré par la nature des documents web (encodage des caractères, balises html et caractères spéciaux non échappés...) peut le plus souvent être géré par l'ajout de filtres au cas par cas. Pour les balises par exemple, il convient d'ajouter la liste des balises les plus communes à la liste des stopwords. Pour d'autres éléments gênants tels des éléments de style (CSS), l'utilisation de filtres basés sur des expressions régulières peut aider à supprimer une partie du bruit. Par exemple, pour filtrer toutes les valeurs numériques portant sur des dimensions ou des tailles de fonte/éléments graphiques, une expression régulière peut être employée pour les séries de chiffres terminant par une unité de mesure (.px, .em, .pt...). Les traitements doivent être adaptés aux jeux de données en fonction de la qualité de leur structuration.

Abstract:

Competitive intelligence regroups several activities from which information analysis, discovery, cross-referencing and synthesis are key, high-level tasks realized by experts of the domain. In a web context, where available data is huge, diverse and rapidly evolving, automating these tasks becomes critical. Creating novel tools able to solve these high-level tasks can only be based on machine learning to replicate complex decision processes, and must secondly take into account precise domain knowledge in the learning process to better reflect the decisions of experts.

This thesis work aims to answer to the issues of classification and information research on the web within the domains of Big Data and Semantic Web. Solutions to these problems are proposed, through the learning of an ontology-based predictive model, from high volumes of data. The integration of domain knowledge in the learning process allows adapting the approach to actual processes of production, thus facilitating the realization of complex tasks done by the experts of the domain.

Keywords: Classification, Big Data, Ontologies, Machine-learning, Web Mining, Competitive Intelligence

Résumé :

La veille stratégique regroupe plusieurs activités parmi lesquelles l'analyse, la découverte, le croisement et la synthèse de l'information sont des tâches de haut niveau effectuées par des experts dans leur domaine. Dans un contexte web, le volume grandissant des données, leur grande variété et la vitesse à laquelle les données évoluent, font apparaître un besoin d'automatiser tout ou partie de ces tâches. Créer des outils de résolution automatique de ces tâches demande d'une part la mise en place d'un processus d'apprentissage automatique permettant d'imiter des processus de décision complexes, et d'autre part la prise en compte des connaissances métier des experts dans le processus d'apprentissage.

Ces travaux tentent de répondre aux problématiques de classification et de recherche d'information sur le web, à l'intersection des domaines du Big Data et du Web Sémantique. L'apprentissage d'un modèle prédictif à partir de grands volumes de données, décrit par une ontologie, propose des solutions à ces problématiques. L'intégration de connaissances métier dans le processus d'apprentissage et de recherche d'information permet l'adaptation de l'approche à un processus de production métier, facilitant ainsi les tâches complexes réalisées par les experts du domaine.

Mots-clés : Classification, Big Data, Ontologies, Apprentissage automatique, Web Mining, Veille Stratégique



■ École doctorale SPIM - Université de Bourgogne/UFR ST BP 47870 F - 21078 Dijon cedex

■ tél. +33 (0)3 80 39 59 10 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

