Projet d'initiation

encadré par

Zied Ben Othmane

À la recherche de la réalité
Rapport intermédiaire de recherche

réalisé par

Younes Belkada

Jérôme Bonacchi

Romain Capuano

Arthur Zucker

Polytech Sorbonne

Mathématiques Appliquées et Informatique Numérique

1re année de cycle ingénieur

2018–2019

Paris, France

Table des matières

Table des figures

Introduction

Notre groupe de quatre étudiants de 1re année de cycle ingénieur fait partie de la spécialité Mathématiques Appliquées et Informatique Numérique de l'école Polytech Sorbonne à Paris. Dans le cadre de notre projet d'initiation, nous avons choisi un sujet proposé par Monsieur Zied Ben Othmane dénommé *À la recherche de la réalité.*

Objectifs

Ce projet consiste à analyser la vérité d'une requête à partir d'une base de données textuelle. Dans un premier temps, nous questionnons le système S et nous nous attendons à recevoir une réponse binaire de sa part. Dans un second temps, le système doit prendre en compte davantage de paramètres : les indices de confiance ou de crédibilité des sources donnant l'information, les relations etc. Dans cette deuxième partie, la réponse n'est plus binaire mais en score de vérité judicieusement calculé. Pour se faire, nous proposons de passer par les ontologies de données, les algorithmes de recherche d'information ainsi que les principes de normalisation ce qui facilitera la recherche de l'information par le biais des axiomes et des règles associées. Le projet termine par proposer un moyen de regrouper les connaissances c'est-à-dire les branches menant à un degré commun de vérité, et cela peut être résolu par des approches d'apprentissage bayésiennes non supervisées.

Plan de réalisation

1. étude bibliographique des divers aspects théoriques et pratiques de ce sujet ;
2. modélisation logique et mathématique du système, ses axiomes et ses règles ;
3. implémentions du système S mettant en œuvre la construction de l'ontologie à partir d'une base de données textuelle ;
4. développement des fonctions de question/réponse et de regroupement des connaissances.

1

Notions primordiales

1.1 *Chatbot*

En premier lieu, un *chatbot* ou dialogueur en français, est un programme informatique qui communique avec un utilisateur, en apportant des réponses à des demandes plus ou moins compliquées. Le *chatbot* doit être capable de discerner les synonymes, d'apporter des informations en rapport avec la demande et interpréter la question posée. Il peut être entraîné sur une base de donnée au préalable. Le but lors de la création d'un *chatbot*, est qu'il passe le célèbre test de Turing.

Les *chatbot* les plus connus de nos jours utilisent, pour répondre aux demandes formulées dans l'interface homme–machine, des langages de programmation créés spécialement pour l'implémentation de tels outils. Ceux-ci utilisent des programmes de reconnaissance des mots, alors que d'autres dialogueurs utilisent une intelligence artificielle.

1.2 Ontologie

> « L'ontologie est aux données ce que la grammaire est au langage. »[3]

Une ontologie, dans le domaine de l'informatique et des sciences, est une modélisation de données qui représente des concepts ainsi que leur relations. L'ontologie permet principalement d'organiser des metadonnées. La citations ci-dessus effectue une comparaison assez légitime puisque l'ontologie permet de structurer, d'ordonner et de mettre en forme des données. Nous nous sommes intéressés à cet outil car nos lectures bibliographiques portant sur notre sujet d'étude nous ont guidés vers l'utilisation d'ontologies.

L'ontologie d'un *chatbot* doit être capable de différencier les mots selon leur contextes, gérer les synonymes, créer des règles permettant à une intelligence artificielle de l'utiliser de manière efficace en définissant des relations entre différentes données.

Après de courtes recherche, il nous est apparu que le site protege.stanford.edu permettait la création d'ontologies de manière gratuite et en ligne, ce qui semble très pratique pour notre projet.

2

Le développement d'ontologies

2.1 Où trouver des ontologies

Une des sources qui nous a été proposée[1] présente une variété de bibliothèques d'ontologies, c'est-à-dire des regroupements permettant la découverte, l'utilisation ou la publication d'ontologies. Cependant, le contenu de cet article n'est pas centré sur le développement d'ontologies, il ne nous est donc peu utile. Mais, les différentes bibliothèques présentée pourront nous permettre d'avoir une vision schématique de ce à quoi devrait ressembler une ontologie.

Nous avons donc effectués des recherches supplémentaires afin d'être en mesure de présenter les différentes étapes de la création d'une ontologie.

2.2 L'utilisation des ontologies par les *chatbots*

Un *chatbot* peut être construit de plusieurs manières différentes. Selon le besoin auquel il doit répondre, il peut être implémenté de façon à donner uniquement des réponses pré-enregistrées et pré-codées. Toutefois, il peut aussi être capable de créer sa réponse en fonction de données collectées, en utilisant par exemple une ontologie, apprenant par lui-même au fur et à mesure qu'on l'entraîne.

Le *chatbot* suis un modèle assez simple, il reçoit une information qu'il doit analyser, stoker pour ensuite créer et envoyer sa réponse. Le processus d'un *chatbot* se décompose de la manière suivante :

1. Transformer la requête reçue à l'aide du *natural language processing* dans une forme utilisable par la machine.
2. Stoker l'information transformée afin de pouvoir travailler avec ultérieurement.
3. Parcourir l'ontologie formée à l'aide d'une base de donnée afin de créer la décision à prendre, le contenue de la réponse.

4. Transformer la décision prise en langage humain : la réponse.

Le *chatbot* pose des questions en fonction des relations présentes dans l'ontologie et des données qu'il a récupérées[4].

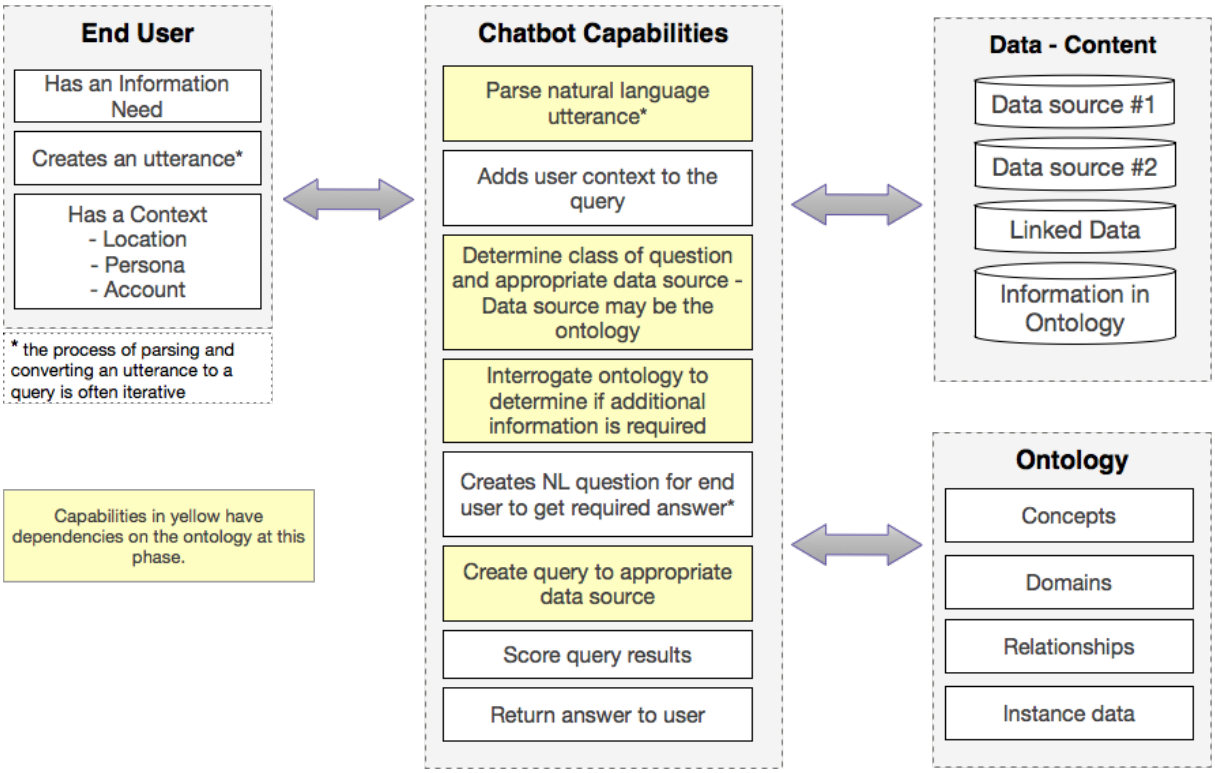

FIGURE 2.1 – Relations entre l'utilisateur, le *chatbot*, les données et l'ontologie.

Les ontologies peuvent être implémentées en Python à l'aide du module `Seth`[I].

I. http://seth-scripting.sourceforge.net/

3

Prétraitements linguistiques

Ce chapitre est composé de nombreuses citations qui sont issues du livre *Recherche d'information : applications, modèles et algorithmes* de Amini et Gaussier[5].

3.1 Segmentation

« La segmentation [...] consiste à séparer une suite de caractères en [...] mots. [...]

Une simple segmentation par des espaces et des signes de ponctuation conduit à des résultats d'indexation médiocres. Par exemple, pour le français nous avons :

— les composés lexicaux à trait d'union
— les composés lexicaux à apostrophe
— les expressions idiomatiques
— les formes contractées
— les sigles et les acronymes »[5]

3.2 Normalisation

« La normalisation de mots est le processus qui transforme tous les mots d'une même famille sous une forme normale ou canonique de façon à ce que l'appariement entre les termes d'une requête et ceux de l'index puissent avoir lieu, et ce malgré les différences qui peuvent exister entre les séquences de caractères de leurs mots associés. [...]

Il existe deux types : la normalisation *textuelle*, ou *surfacique*, et la normalisation *linguistique*. »[5]

3.2.1 Normalisation textuelle

> « La normalisation textuelle rend les mots d'une même famille sous leur forme canonique en effectuant quelques transformations superficielles sur leurs séquences de caractères. »[5]

En bref, il faut enlever les points et les traits d'unions apparaissant entre les mots, convertir la première lettre des phrases et enlever tous les diacritiques[I].

3.2.2 Normalisation linguistique

> « La normalisation linguistique consiste à ramener un mot fléchi sous sa forme canonique. Il existe deux types de normalisation linguistique : la *racinisation* et la *lemmatisation*.
>
> La racinisation se rapporte au procédé qui cherche à regrouper les différentes variantes morphologiques d'un mot autour d'une même racine. […]. Le risque est qu'on peut aussi ramener des documents qui contiennent les mots ayant les mêmes racines que les mots de la requête mais qui sont sémantiquement totalement différents de ces derniers. […] La lemmatisation fait une analyse linguistique poussé destinée à enlever les variantes flexionnelles des mots afin de les ramener sous leur forme lemmatisée ou encyclopédique. À l'opposé de la racinisation, le résultat de la lemmatisation n'est autre que des mots de l'encyclopédie ; elle ne conduit ainsi pas à l'agrégation de mots très différents. »[5]

Par exemple, le logiciel *TreeTagger* développé par l'université de Stuttgart permet de segmenter (et de normaliser par lemmatisation) des textes de plusieurs langues européennes[II]. Snowball est un langage conçu pour créer des algorithmes de normalisation. Le site internet dédié[III] met à disposition un algorithme de normalisation utilisé pour le français[IV].

3.2.3 Algorithme de Carry

L'algorithme de Carry est expliqué dans *Carry, un algorithme de désuffixation pour le français*[6] de Paternostre, Francq, Lamoral et al.

La désuffixation[V] est un procédé permettant de limiter l'ensemble de mots sur lequel travailler. En effet, la langue française comprend exactement 225 508 mots (les variations de

I. Signe qui est ajouté à une lettre de l'alphabet pour en modifier la prononciation.
II. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
III. http://snowballstem.org/
IV. http://snowballstem.org/algorithms/french/stemmer.html
V. Création d'un mot nouveau par suppression du suffixe.

conjugaisons et les pluriels sont pris en compte), l'anglais comprend 197 820 mots. Plutôt que de travailler sur l'intégralité des mots, l'objectif est de les regrouper par familles, ces dernières sont ensuite représentées par un mot de base. La méthode *Mmorph* est la méthode dite classique. Elle se base sur la morphologie d'un langage pour en extraire les familles de mots. Cependant, il existe deux principaux algorithmes produisant des ensembles de familles de mots plus précis : la méthode de Porter en anglais et la méthode de Carry en français. Les principes de ce dernier sont à disposition sur le site *The Paul Otlet Institute*[7]. De plus, ces algorithmes respectent plusieurs règles, en effet on ne peut pas retirer des suffixes dès qu'ils sont identifiés. Ces algorithmes peuvent être comparés sur plusieurs critères : la précision et le rappel.

Rappel

Pour une forme de base donnée, le rappel est calculé comme suit : nombre de mots avec lesquels il partage le même radical et présents sous la même forme de base / nombre de mots avec lesquels il partage la même forme de base. Le rappel est moyenné sur toutes les formes de base présentes dans Mmorph.

Précision

Pour un radical donné, la précision est calculé comme suit : nombre de mots avec lesquels il partage la même forme de base et présents sous le même radical (après désuffixation) / nombre de mots avec lesquels il partage le même radical. La précision est moyennée sur tous les radicaux découverts par les algorithmes de désuffixation.

3.3 Filtrage par antidictionnaire

> « Un antidictionnaire […], ou liste de *mots vides*, est une liste de mots qui tendent à être présents avec une fréquence élevée dans tous les documents d'une collection et qui n'apportent que peu d'information sur le contenu. […] La distribution des mots vides est identique dans tous les documents d'un corpus, ceci quelles que soient les thématiques auxquelles ils appartiennent, et les mots vides n'ont ainsi pas un pouvoir discriminant élevé pour ces tâches. […]
>
> Il existe des antidictionnaires indépendants du domaine de la collection que l'on considère et d'autres qui en sont dépendants.
>
> — Les antidictionnaires dépendants d'un domaine sont construits en triant d'abord les mots suivant leur nombre total d'apparitions dans la collection et en sélectionnant ensuite à la main les mots les plus fréquents et à faible contenu informatif. […]
> — Pour les antidictionnaires généraux, […] on se repose soit sur la catégorie grammaticale des mots […], soit sur un ensemble de collections de différents

domaines d'une langue donnée, ensemble sur lequel on effectue le travail présenté au point précédent.

Une fois cette liste constituée, on filtre les documents du corpus en enlevant les mots présents dans cet antidictionnaire. […] Après le filtrage des documents par un antidictionnaire, l'ensemble des *termes* restants constitue le vocabulaire de la collection.

Une des principales motivations au filtrage des mots vides a été depuis longtemps le gain d'espace disque ou mémoire que ce filtrage entraînait. […] Les systèmes de recherche d'information modernes ne filtrent pas les documents et indexent tous les mots vides, même si leur prise en compte au moment de l'interrogation varie »[5]

Des antidictionnaires généraux pour la langue française sont à disposition sur les sites suivants :

— https://www.ranks.nl/stopwords/french
— https://countwordsfree.com/stopwords/french
— http://snowball.tartarus.org/algorithms/french/stop.txt

4

Représentation documentaire et recherche d'information

Ce chapitre est composé de nombreuses citations qui sont issues du livre *Recherche d'information : applications, modèles et algorithmes* de Amini et Gaussier[5].

« En recherche documentaire, le but est de trouver les documents qui contiennent les termes d'une requête donnée. »[5]

4.1 Modèle vectoriel

« Avec cette représentation, on associe, à chaque document d d'une collection C, un vecteur $\mathbf{d}$ dont la dimension correspond à la taille du vocabulaire. L'espace vectoriel considéré est donc un espace de termes dans lequel chaque dimension est associée à un terme de la collection.

$$\forall d \in C, \mathbf{b} = (w_{id})_{i \in \{1,\dots,V\}} \tag{4.1}$$

Dans ce cas, w_{id} est le poids que le terme i du vocabulaire a dans le document d. […] Avec cette représentation, l'ordre d'apparition des termes dans les documents n'est pas pris en compte. »[5]

4.2 Pondération des termes

« Une approche simple et classique pour estimer le poids $(w_{id})_{i=1}^{V}$ du vecteur associé au document d est de calculer le nombre d'occurrences des termes du vocabulaire dans le document. Cette approche connue en anglais sous le nom de *term frequency* (tf) définit ainsi chaque poids associé à un terme par le nombre de fois

où ce terme apparaît dans un document :

$$\forall i \in \{1, \dots, V\}, w_{id} = \mathrm{tf}_{t_i,d} \tag{4.2}$$

Cela dit, cette représentation fréquentielle introduit un biais problématique, les documents longs tendent à comporter davantage de répétitions de termes que les documents courts. […] Ceci étant, avec cette approche, tous les termes du vocabulaire apparaissant dans un document avec la même fréquence recevront le même poids. [Les termes qui figurent dans peu de documents] ont un pouvoir de discrimination plus grand que celui des termes qui apparaissent dans beaucoup de documents. […] Dans le calcul des poids, on prend en compte le nombre de documents dans lesquels un terme t du vocabulaire apparaît, appelé *document frequency* en anglais et usuellement noté par df_t, en multipliant le nombre d'apparitions du terme t dans un document d donné par son idf_t :

$$\mathrm{idf}_t = \ln \frac{N}{\mathrm{df}_t} \tag{4.3}$$

où N est le nombre de documents de la collection. […]

En recherche d'information, le codage le plus courant des documents […] est défini comme :

$$\forall i \in \{1, \dots, V\}, w_{id} = \mathrm{tf}_{t_i,d} \times \mathrm{idf}_{t_i} \tag{4.4}$$

[…] La forme générale du codage des termes est ainsi :

$$\forall i \in \{1, \dots, V\}, w_{id} = n_d \times p_{\mathrm{tf}_{t_i,d}} \times p_{\mathrm{df}_{t_i}} \tag{4.5}$$

où $p_{\mathrm{tf}_{t_i,d}}$ et $p_{\mathrm{df}_{t_i}}$ sont des poids fondés sur la fréquence du terme t_i dans d. »[5]

4.3 Index inversé

« L'utilisation d'une structure qui fait correspondre chaque terme du vocabulaire à la liste des documents qui le contiennent est la façon la plus rapide pour trouver un terme d'une requête dans une collection de documents. Cette structure s'appelle communément un index inversé.

[…] Les deux types de structure de données les plus utilisés pour la construction de l'index inversé sont des *tables de hachage* ou des *arbres*. »[5]

Les étapes de l'indexation sont :

«

1. L'extraction de paires d'identifiants (terme, document), en opérant une passe complète sur la collection. […]
2. Le tri de paires suivant les clés d'identifiants de termes puis les clés d'identifiants de documents.
3. Le regroupement des paires en construisant pour chaque identifiant de terme, la liste des identifiants de documents dans lesquels le terme apparaît. »[5]

4.4 Modèles de recherche

« Les modèles de recherche sont des programmes qui aident les utilisateurs à trouver les informations qu'ils recherchent dans une collection de documents textuels ou éventuellement multimédias. Pour une demande d'information donnée, le but de ces modèles est de retourner un sous-ensemble de documents de la collection qui pourraient contenir l'information recherchée. »[5]

4.4.1 Modèles booléens

« Les modèles booléens de recherche sont fondés sur la logique booléenne et la théories des ensembles. […] Ils ont été conçus de façon à répondre exactement aux requêtes formées par des expressions combinant des termes et des opérateurs logiques *ET*, *OU* et *SAUF*. »[5]

4.4.2 Modèles vectoriels

« Les modèles de recherche vectoriels […] [assignent] […] des scores de similarité à toutes les paires formées par une requête et un document d'une collection donnée. L'hypothèse de base des modèles vectoriels est que les requêtes peuvent être représentées dans le même espace vectoriel que les documents. […]

Ainsi, pour chaque requête q on considère sa représentation vectorielle $\mathbf{q} \in \mathbb{R}^V$. Le but d'un modèle de recherche est alors de prédire pour la requête fixée un ordre sur les documents de la collection C en utilisant une fonction de score s. »[5]

Modèle vectoriel de Salton

« Si on considère un document d de la collection C et une requête d, avec des représentations vectorielles respectives $\mathbf{d}$ et $\mathbf{q}$, la mesure cosinus entre ces deux vecteurs est :

$$s(q, d) = \frac{\langle \mathbf{d}, \mathbf{q} \rangle}{\|\mathbf{d}\| \times \|\mathbf{q}\|} \tag{4.6}$$

Dans ce cas, plus l'angle entre les vecteurs d'un document et de la requête est petit […], plus le document est supposé être pertinent par rapport à la requête. »[5]

4.4.3 Modèles probabilistes

Dans "Bayesian modelling and inference on mixtures of distributions"[8], Mengersen, Marin et Robert nous présentent comment aujourd'hui, grâce à des analyses et méthodes complexes, nous pouvons aboutir à une prédiction proches de la réalité.

Les modèles de Bayes sont appliqués aux modèles de mélange qui sont des modèles statistiques permettant de modéliser différentes sous-populations dans une population globale sans que celles-ci soient identifiées dans les données par une variable observée. Ces modèles sont fréquemment utilisés pour des applications liées à l'intelligence artificielle.

Principe d'ordonnancement probabiliste

> « Ce principe […] considère chaque paire constituée d'un document et d'une requête (d, q) comme la réalisation d'une expérience aléatoire qui consiste à tirer simultanément ces observations d'un ensemble prédéfini de documents et de requêtes. À chaque tirage (d, q) on associe alors une variable aléatoire $R_{d,q}$ qui est égale à 1 si le document d est pertinent par rapport à la requête q et 0 sinon. »[5]

5

Graphes pondérés et multigraphes

Dans “Analysis of weighted networks”[2], Newman nous présente comment un graphe pondéré peut être représenté comme un multigraphe non pondéré.

> « Un graphe pondéré peut être représenté mathématiquement par une matrice d'adjacence avec des entrées qui ne sont pas simplement zéro ou un, mais égaux au poids des arêtes à la place. »[2]

$$A_{ij} = \text{poids des connections de } i \text{ à } j \tag{5.1}$$

Pour convertir un graphe pondéré en un multigraphe, il suffit que :

> « chaque arête avec un poids n soit remplacée par n arêtes parallèles chacune de poids 1 connectant les mêmes sommets. La matrice d'adjacence reste inchangée et n'importe quelle techniques pouvant être appliquée normalement au graphe non pondéré peut maintenant être appliquée aussi aux multigraphes. »[2]

> « le k_i d'un sommet i dans un graphe pondéré est la somme des poids des arêtes incidentes : »[2]

$$k_i = \sum_j A_{ij} \tag{5.2}$$

6

Traitement automatique du langage naturel

La notion de traitement automatique du langage naturel, ou *natural language processing* en anglais (abrégée en *NLP* par la sutie), a été introduite dans les années 1950 par Alan Turing qui créa le *Test de Turing* pour définir le critère d'intelligence d'une machine. Nous pourrons d'ailleurs faire passer notre futur *chatbot* par ce test si nécessaire (ou par simple curiosité).

La *NLP* est une notion fondamentale et dont la compréhension est absolument nécessaire afin de pouvoir comprendre la philosophie du *chatbot*. Cette notion, largement utilisée dans le domaine de l'I.A., se définit comme étant une interface entre la machine et l'utilisateur qui peut être appliquée dans divers domaines tels que la reconnaissance vocale.

Les connaisseurs[9] définissent cette notion comme étant à mi-chemin entre la machine et l'humain et permettant à la machine de comprendre ce que l'humain souhaite dire à cette dernière. Elle peut prendre différentes formes et être utilisée par différents moyens et pour des fins variés.

Voici quelques exemples de *NLP* que nous pouvons utiliser pour notre projet :
- **OpenNLP :** est une bibliohthèque *opensource* de machine learning que nous pouvons utiliser pour faire de la segmentation de texte (utilisée également pour du *machine learning*) et du traitement de texte poussé.
- **Natural Language Toolkit :** est une bibliohthèque *opensource* en Python qui permet de faire du traitement de texte poussé.

Ces outils peuvent être utilisés pour par exemple résumer un texte en gardant les phrases importantes et en ignorant le reste, analyser la subjectivité d'un texte pour en extraire son opinion, ses sentiments etc. On voit bien que les applications sont variés et très étendus.

La compréhension et l'analyse du langage humain par une machine devient une chose faisable à travers la *NLP*, mais comprendre les subtilités de cette dernière semble être compliquée.

Bibliographie

Articles

[1] M. d'Aquin et N. F. Noy, "Where to publish and find ontologies? a survey of ontology libraries", *Web semantics (Online)*, t. 11, p. 96–111, 1er mar. 2012, ISSN : 1570-8268. DOI : 10.1016/j.websem.2011.08.005. adresse : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3293483/ (visité le 10/03/2019).

[2] M. E. J. Newman, "Analysis of weighted networks", *Phys. Rev. E*, t. 70, p. 056 131, 5 nov. 2004. DOI : 10.1103/PhysRevE.70.056131. adresse : https://link.aps.org/doi/10.1103/PhysRevE.70.056131.

Autres sources

[3] *Ontologie (informatique)*, in *Wikipédia*, Page Version ID: 156460030, 4 fév. 2019. adresse : https://fr.wikipedia.org/w/index.php?title=Ontologie_(informatique)&oldid=156460030 (visité le 14/03/2019).

[4] fmuxu0, *How taxonomies/ontologies help chatbots connect people with information*, in *Scope eKnowledge*. adresse : https://www.scopeknowledge.com/SmartContent/how-taxonomies-ontologies-help-chatbots-connect-people-with-information/ (visité le 20/03/2019).

[5] M.-R. Amini et É. Gaussier, *Recherche d'information : Applications, modèles et algorithmes*, 2e éd. Eyrolles, 2016, 274 p., ISBN : 978-2-212-67376-0.

[6] M. Paternostre, P. Francq, J. Lamoral, D. Wartel et M. Saerens, *Carry, un algorithme de désuffixation pour le français*. 2002. adresse : https://docplayer.fr/15614748-Carry-un-algorithme-de-desuffixation-pour-le-francais.html.

[7] P. Francq, *Wikics*, in *The Paul Otlet Institute*. adresse : http://www.otlet-institute.org/wikics/Articles.html (visité le 20/03/2019).

[8] K. Mengersen, J. M. Marin et C. Robert, "Bayesian modelling and inference on mixtures of distributions", in *Essential Bayesian models. Handbook of statistics: Bayesian thinking - modeling and computation*. Elsevier, 2011, t. 70, ISBN : 9780444537324. adresse : https://www.ceremade.dauphine.fr/~xian/mixo.pdf.

[9] *Introduction to natural language processing*, in *Algorithmia*. adresse : https://blog.algorithmia.com/introduction-natural-language-processing-nlp/ (visité le 20/03/2019).