# Amélioration de la partie française de IBM Watson Analytics for Social Media

Renaud Mousnier-Lompré

# Amélioration de la partie française de *IBM Watson Analytics for Social Media*

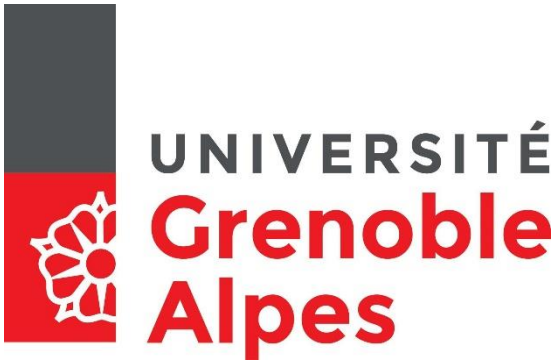
**MOUSNIER-LOMPRÉ**
**Renaud**

Sous les directions de
Georges ANTONIADIS (UGA) et Thilo GÖTZ (IBM)

UFR LLASIC - Département I3L

Mémoire de master 2 mention Sciences du langage - 20 crédits
Parcours : Industries de la langue

Année universitaire 2016-2017



# Amélioration de la partie française de *IBM Watson Analytics for Social Media*

**MOUSNIER-LOMPRÉ**
**Renaud**

Sous les directions de
Georges ANTONIADIS (UGA) et Thilo GÖTZ (IBM)

UFR LLASIC - Département I3L

Mémoire de master 2 mention Sciences du langage - 20 crédits
Parcours : Industries de la langue

Année universitaire 2016-2017

## Remerciements

Je tiens avant tout à remercier celle qui sera mon épouse dans quelques semaines, sans qui je n'aurais pas postulé pour ce stage, ni n'aurais ainsi surmonté les débuts houleux de ce séjour en Allemagne.

Même s'ils ne parlent pas français et ne liront pas ce rapport de stage, je voudrais mentionner mes nombreux collègues stagiaires, qui ont apporté une réelle valeur ajoutée à ces six mois à IBM, *via* nos relations d'amitié voire de complicité, nos débats sur les tâches communes, nos discussions sur le TAL ou encore nos incessants échanges culturels.

Je remercie également Thilo Götz et Verena Henrich, mes supérieurs directs, de m'avoir permis d'effectuer ce stage dans leur équipe, ainsi que pour la patience, la gentillesse, la compréhension et le sérieux dans la décontraction dont ils ont fait preuve au quotidien.

Merci aussi à Georges Antoniadis d'avoir accepté d'être mon tuteur à l'université, et de manière générale pour la qualité du master IDL : la variété tant de ses enseignements que de ses enseignants offrent des atouts que beaucoup d'autres formations ne proposent pas.

Pour finir, j'ai une pensée émue pour mes estimés collègues de la promotion 2017 du master IDL, en particulier Aymen, Doriane, Gülüm, Ieva, Louise, sans oublier Sylvain et William. Ça n'aura pas été trivial, nous ne sommes pas des spécialistes de la sauge, mais nous aurons fait de notre mieux pendant ces deux années pleines de bons souvenirs (et de moins bons, mais qui se bonifieront avec le temps).

UNIVERSITÉ Grenoble Alpes

**DÉCLARATION**

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

NOM : MOUSNIER-LOMPRÉ

PRENOM ; RENAUD

DATE : 08/09/2017

SIGNATURE : [redacted]

# Sommaire

# Introduction

Ce rapport de stage présente le travail que nous avons effectué à IBM Allemagne, dans le cadre du master Industries de la Langue de l'université Grenoble-Alpes. Au sein d'une équipe d'informaticiens et d'ingénieurs linguistes, nous avons pu mettre à profit nos connaissances acquises sur les bancs de l'université pour participer à l'amélioration du logiciel qu'IBM propose pour l'analyse de sentiments.

*IBM Watson Analytics for Social Media* est un logiciel en ligne destiné aux responsables du marketing et autres professionnels intéressés par la réputation numérique de leur entreprise et de leurs produits ou services. L'utilisateur peut effectuer des recherches sur les tendances d'opinion des internautes concernant une entité particulière, avec force détails sur la personne exprimant un sentiment positif ou négatif. Notre stage portait sur l'amélioration du code du produit lié au français, nous basant sur l'existant ou sur des versions de langues étrangères.

Dans une première partie, nous présenterons la multinationale qu'est IBM, ses différents secteurs d'activité et où se situe *Watson Analytics for Social Media* dans cet ensemble, avant d'aborder la façon dont le système fonctionne, ainsi que ses fonctionnalités pour l'utilisateur final. Nous détaillerons ensuite l'équipe dans laquelle nous avons travaillé, ainsi que le langage que nous utilisions en mettant l'accent sur l'étiquetage morpho-syntaxique qu'il propose.

La seconde partie expose les modules de la partie spécifique à chaque langue que traite le logiciel. Nous en ferons une description en précisant la façon dont nous l'avons enrichi, avant de présenter les résultats obtenus en gardant un œil critique sur notre travail.

# Contexte de travail

Notre stage s'est déroulé à Böblingen, dans les locaux de la section Recherche et Développement d'IBM Allemagne, une des nombreuses filiales du groupe IBM.

## *1. IBM*

International Business Machines, plus connu sous son acronyme IBM, est une société américaine fondée en 1911 aux États-Unis[1], spécialisée dans les services, matériel et logiciels informatiques. Elle compte actuellement 380 000 employés[2] répartis dans 170 pays[2] pour un chiffre d'affaires qui s'élevait en 2015 à 81,8 milliards de dollars américains[3].

L'histoire d'IBM est jalonnée de découvertes et de progrès[1], la recherche ayant une place primordiale dans l'entreprise. Parmi les innovations notables, on peut citer le langage FORTRAN en 1957, les ordinateurs System/360, la découverte de la géométrie fractale, des avancées dans la nanotechnologie, un investissement massif dans Linux et l'open source, *Deep Blue* qui bat en 1997 le champion du monde des échecs, *Watson* ceux du jeu télévisé de culture générale *Jeopardy!* en 2011, etc.

> Au cours de son histoire, IBM inventera rien de moins que le distributeur automatique de billets, le code barre, la bande magnétique qu'on trouve sur nos cartes bleues, le disque dur, la disquette et les barrettes de RAM, glanant au passage 5 prix Nobel et 6 prix Turing.[4]

Les travaux actuels de recherche d'IBM[5] concernent un plastique entièrement recyclable plus solide que l'os, des systèmes de détection de maladies, de l'information quantique dans le cloud, des avancées sur des nouvelles capacités de stockage de données ou encore des « environnements intelligents ».

À côté de ses occupations dans la recherche, IBM propose une large palette de services[6], allant du développement de logiciels aux solutions commerciales « intelligentes » ou dans le cloud, en passant par du consulting, des opérations numériques, de l'analyse de la finance, de ses risques et de la fraude, ou encore de l'analyse de *big data*.

---

[1] https://www-03.ibm.com/ibm/history/ibm100/us/en/icons/, dernière consultation le 28 août 2017
[2] http://fortune.com/2016/02/01/ibm-employee-performance-reviews/, dernière consultation le 28 août 2017
[3] https://www.ibm.com/ibm/fr/fr/, dernière consultation le 28 août 2017
[4] http://www.numerama.com/startup/ibm, dernière consultation le 28 août 2017
[5] http://research.ibm.com/#featured-projects, dernière consultation le 28 août 2017
[6] https://www.ibm.com/services/en-us/, dernière consultation le 28 août 2017

Ces services sont utilisés dans des domaines d'une grande variété[7] : défense et aérospatiale, transports, énergie, éducation, biologie, pétrochimie, santé, assurance, banque, électronique, télécommunications, médias et divertissements…

Toute entreprise relevant de ces domaines peut trouver de l'intérêt à garder un œil sur l'image que clients et simples internautes ont d'elle, ce qui est précisément ce que *IBM Watson Analytics for Social Media* (dorénavant « WASM ») permet de faire.

## *2. IBM Watson Analytics for Social Media*

L'objet de notre travail était l'amélioration de la partie française du code de WASM.

### *2.1. Présentation*

Le logiciel WASM fait partie de la famille des produits rattachés à *IBM Watson,* application d'intelligence artificielle utilisant le traitement automatique du langage et l'apprentissage automatique pour répondre à des questions posées en langue naturelle et non dans un langage formel, le tout dans le cloud.

WASM permet à son utilisateur de visualiser les tendances observées sur les médias sociaux grâce aux messages publics des internautes[8]. Ceux-ci sont traités automatiquement afin d'extraire des informations sur leur auteur, son comportement et son opinion vis-à-vis de l'entité sur laquelle porte la recherche. Grâce à une interface utilisateur basée sur Watson Analytics, le client peut poser à l'écrit des questions en langage naturel et se voir proposer la meilleure visualisation considérant le type de données de la réponse (lieu, date, nombres…).

De nombreux filtres permettent ensuite d'affiner les résultats, modifiant automatiquement leur affichage. Par exemple, l'utilisateur peut choisir de n'afficher que les messages écrits en italien à New York entre le 23 et le 27 mars 2017 sur des blogs ou des forums de discussion, par des femmes dont le pseudonyme ne contient pas de chiffres, qui s'intéressent au maquillage et qui ne pensent pas de bien de l'entité cherchée. Il s'agit donc d'un outil utile aux responsables marketing afin d'observer les attentes des consommateurs, de constater la réussite ou l'échec d'une campagne de publicité, de savoir ce que les internautes pensent de leurs produits et de ceux de leurs concurrents, etc. Si la majorité des clients de WASM sont des entreprises commerciales, il arrive néanmoins que des entités plus inattendues comme des institutions politiques ou des associations aient recours à cette application.

[7] https://www.ibm.com/industries/en-us/, dernière consultation le 28 août 2017
[8] https://www.ibm.com/us-en/marketplace/social-media-data-analysis, dernière consultation le 28 août 2017

### *2.2. Pipeline*

#### 2.2.1. Création d'une configuration par l'utilisateur

Le traitement d'une requête commence par la création d'une configuration de la part de l'utilisateur. Celui-ci commence par définir un *sujet,* en entrant tous les mots-clefs qui seront considérés comme étant pertinents pour la recherche. Celle-ci est insensible à la casse, mais il faut penser aux variations d'écriture comme la possible absence de diacritiques et des ligatures, voire d'espaces et d'apostrophes. Une liste de suggestions est proposée à l'utilisateur à mesure qu'il ajoute des mots-clefs, lui permettant d'avoir un aperçu du type de résultats qu'il va obtenir, grâce à l'analyse de cooccurrences notamment. S'il constate qu'un des éléments est ambigu, il a la possibilité d'y remédier en utilisant les deux autres champs, qui permettent d'exclure certains résultats s'ils apparaissent proches de certains autres termes, ou bien, inversement, de ne les inclure que s'ils sont entourés de ces termes.

Après avoir créé plusieurs *sujets* de recherche, l'utilisateur peut ajouter des *thèmes,* de la même façon que pour les *sujets*. Les *thèmes* ne modifient en rien les documents obtenus mais en permettent des regroupements selon la présence ou non de certains mots-clefs. Si la configuration concerne les restaurants d'une même rue, les *sujets* seront probablement leurs noms, et les *thèmes* peuvent être « prix », « qualité » et « desserts », par exemple.

Une fois tous les mots-clefs enregistrés, l'utilisateur doit choisir la ou les langues des documents qu'il souhaite obtenir, à choisir entre les neuf langues proposées : allemand, anglais, arabe, chinois, espagnol, français, italien, portugais et russe. Si toutes n'ont pas la même profondeur d'analyse des informations concernant les auteurs des messages, l'analyse de sentiments, aspect le plus important du logiciel, est toujours disponible.

L'utilisateur sélectionne ensuite les sources qu'il souhaite voir utilisées. Les documents peuvent en effet provenir de différents types de médias sociaux : forums de discussion, blogs, articles de journaux, commentaires de vidéos, sites de critiques, Twitter ainsi qu'une liste prédéterminée de pages Facebook. Ces sources seront amenées à évoluer, incluant des sites propres à certaines langues.

Ne reste plus alors qu'à sélectionner la période sur laquelle s'étend l'analyse. Celle-ci peut s'étendre jusqu'à deux ans avant la date d'utilisation, permettant de préciser le moment où les messages doivent avoir été écrits, en indiquant la date et l'heure de début puis de fin de la période qui intéresse l'utilisateur.

Avant la soumission de la configuration est affichée une estimation du nombre de documents correspondant à la future recherche. L'utilisateur achetant l'analyse d'un certain nombre de documents, il peut vouloir se raviser et revoir certains éléments, comme le moment d'envoi ou les sources.

### 2.2.2. Traitement de la requête

Quand l'utilisateur est satisfait de sa configuration, celle-ci est envoyée sous la forme d'une requête aux fournisseurs de données partenaires, qui renvoient aux serveurs d'IBM certaines des métadonnées disponibles, ainsi que le contenu du message, épuré des balises HTML. Les différents fournisseurs ne disposant pas d'une même structuration des données, une étape d'uniformisation s'impose, qui permet ensuite le traitement des données linguistiques depuis des fichiers JSON (JavaScript Object Notation) similaires. Ce traitement est promptement réalisé grâce à la combinaison de la rapidité d'exécution de l'extracteur et de la puissance d'Apache Spark, qui manipule plusieurs documents en parallèle sur des machines ultra-performantes. Chaque centre de données d'IBM, présents aux quatre coins du globe, dispose en effet de sept ordinateurs dont l'usage est réservé à WASM, chacun avec 64 processeurs et 150 GB de RAM, sans oublier dix systèmes Spark par machine.

Le champ « texte » des fichiers JSON est dans un premier temps prétraité, pour marquer les mots-clefs correspondant à la recherche, mais aussi les suites de caractères correspondant aux entrées des listes des sentiments (cf. *infra*). Pendant l'analyse linguistique en AQL, des métadonnées sont ajoutées pour chaque document traité, comme le type d'utilisateur (entreprise, médias, personne privée…), ses intérêts, son rapport à l'objet de la recherche, et autres informations pouvant intéresser l'utilisateur. Étant l'objet même de notre travail à IBM, la fouille de ces éléments est décrite dans la deuxième partie de ce rapport de stage.

Après cette analyse du texte, les documents sont consolidés : les messages strictement identiques à d'autres sont supprimés, et les informations sur un même auteur, recueillies depuis plusieurs de ses messages, sont mises en commun puis associées à chacun de ces messages. C'est aussi à ce stade du traitement que les documents considérés comme indésirables sont supprimés. Subséquemment, une recherche des thèmes présents dans l'ensemble des documents est effectuée grâce à de l'apprentissage automatique, pour permettre à l'utilisateur de savoir ce dont on parle en même temps que de ce sur quoi porte sa

recherche (nous avons dû établir une liste de mots-outils à exclure des mots les plus fréquents).

S'ensuit l'envoi à l'interface utilisateur d'un fichier CSV contenant pour chaque message les informations à afficher, tant celles des fournisseurs de données que celles obtenues par analyse textuelle. Outre des possibilités variées de visualisation des résultats accompagnées d'un grand nombre de filtres, l'utilisateur peut voir les messages considérés comme positifs ou négatifs pour le concept cherché. Il s'agit en réalité de *snippets*, sortes de paragraphes constitués de trois phrases : une centrale contenant le mot-clef, entourée de celle qui la précède et la suit ; un même document peut donc contenir plusieurs *snippets*. L'utilisateur peut vérifier l'exactitude des résultats, et, selon la gravité de l'éventuelle erreur dans l'analyse des sentiments du *snippet*, en faire la remarque à IBM. S'il ne peut pas ajouter sa propre liste de mots-clefs pour les sentiments positifs et négatifs, il peut désactiver un terme à partir de ce *snippet* qui lui est proposé. Pour ce qui est de l'analyse de sentiments, WASM travaille sur ces *snippets* et non sur l'intégralité de chaque document, contrairement à la plupart des systèmes existants.

Cette personnalisation des résultats, l'affichage des données textuelles, la puissance de l'interface utilisateur avec le traitement automatique de l'écrit et les nombreux diagrammes évolutifs incluant les *thèmes* et les informations sur les internautes, sans oublier les nombreuses langues et sources disponibles, sont autant d'atouts de WASM par rapport à la concurrence.

## *3. Travail sur WASM*

### *3.1. L'équipe*

L'équipe de WASM est divisée en deux, l'une à Ottawa au Canada travaillant sur l'interface utilisateur, l'autre à Böblingen en Allemagne se concentrant sur l'analyse de texte. Cette dernière, où nous avons effectué notre stage, se compose ainsi :

- un chef de projet, décideur des questions techniques ;
- un manager orienté ressources humaines et relations avec les autres équipes ;
- deux ingénieurs en informatique aux tâches orientées vers les bases de données et les transferts d'information, travaillant principalement en Scala et Java ;
- une ingénieure à mi-temps chargée de la détection de messages indésirables ;
- trois linguistes-informaticiens (nos encadrants et supérieurs directs) se concentrant principalement sur l'aspect linguistique de l'application ;

- un étudiant en licence d'informatique dans un programme d'IBM en alternance ;
- un nombre variable de stagiaires, le plus souvent étudiants en TAL ou en linguistique avec des connaissances en informatique, bien qu'il soit arrivé que des philosophes prennent part au travail sur WASM.

Entre le début du projet en 2010 et notre stage en 2017, entre cinquante et cent stagiaires ont collaboré au projet, chacun travaillant sur la langue dont il est locuteur natif ou spécialiste, selon les besoins d'IBM ; un collègue américain a ainsi été chargé d'apporter son savoir linguistique tant en anglais qu'en mandarin et en arabe.

Le français est une langue souvent apprise à l'école en Allemagne, ce qui explique que certains ingénieurs aient des bases plus ou moins solides dans la langue de Corneille. Ces connaissances étaient estimées acceptables pour proposer une version française de WASM, et bien qu'elles ne fussent pas erronées, elles se sont avérées largement insuffisantes (cf. *infra*). Ainsi, nous sommes le premier stagiaire natif d'un pays francophone à travailler à temps plein sur le code français. Si les performances du logiciel étaient convenables à notre arrivée, elles ne pouvaient être qu'améliorées, ce qui était l'objet de notre stage.

### *3.2. Conditions de travail*

Tout au long du stage, nous avons joui d'excellentes conditions de travail, tant grâce à IBM qu'aux collègues stagiaires et à l'ambiance d'entraide et de collaboration qui régnait.

#### 3.2.1. Générosité d'IBM

L'entreprise dispose de deux types de contrat, selon que le stage est obligatoire ou non, ce dernier cas étant plus avantageux (rémunération et jours de congé). Afin d'optimiser notre situation, nous nous sommes vu proposer deux contrats : l'un de quatre mois correspondant à la durée minimale du stage obligatoire dans le cadre du master Industries de la Langue à l'UGA, l'autre de deux mois, volontaire. Ainsi, nous travaillions 38 heures par semaine, rémunéré entre 1100 et 1300 € mensuels, et entre 1,5 et 2,5 jours de congés par mois, soit un total de treize jours sur les six mois de notre stage à IBM.

Jusqu'à quelques semaines après notre arrivée, le vendredi était le jour du télétravail. Si cela a finalement été annulé pour les employés, conséquence d'une politique globale par IBM de « retour au bureau[9] », il a été permis aux stagiaires de maintenir cette pratique. En effet, nous ne participions pas aux nombreuses réunions de l'équipe, et nous dispositions

[9] https://qz.com/924167/ibm-remote-work-pioneer-is-calling-thousands-of-employees-back-to-the-office/, dernière consultation le 28 août 2017

d'une messagerie instantanée interne pour communiquer entre nous, voire du téléphone dans des cas le nécessitant.

De manière générale, nos supérieurs nous faisaient confiance. Nous organisions notre temps à notre convenance, pouvant finir nos heures le samedi et le dimanche, sans contrôle sur notre quantité de travail. Sauf cas exceptionnels d'urgence liée à des dates butoir, on nous laissait tout le temps dont nous avions besoin pour réaliser nos tâches, sans jamais nous mettre de pression. Bien que certains (rares) stagiaires profitent de cette largesse, la plupart sont honnêtes, travaillant parfois même plus que les heures pour lesquelles ils sont payés.

### 3.2.2. Ambiance au sein de l'équipe

Tous les stagiaires étant jusqu'à un certain degré spécialistes de linguistique, et donc curieux à propos des langues et des cultures étrangères, une ambiance détendue a régné au sein de l'équipe. Nous nous entraidions volontiers et échangions notre savoir-faire, que l'objet soit une langue commune ou non.

Nous sommes arrivés à IBM en même temps que deux autres personnes. Un stagiaire se trouvait déjà dans l'équipe mais n'était pas affecté aux mêmes tâches que nous et ne pouvait nous aider que sur les locaux d'IBM et la vie en Allemagne. Nous avons ainsi dû apprendre un nouveau langage par nous-mêmes, sans le moindre tutoriel, avec pour seules références celles du site d'IBM[10] et du logiciel Eclipse, brutes et froides, nullement pédagogiques. Nous avons donc mis en commun nos « bonnes pratiques » et résumé les erreurs habituelles, afin que nos successeurs n'aient à subir les mêmes lectures et désagréments que nous. Cela s'est avéré extrêmement utile, et ce document d'aide ne cesse d'être amélioré, chacun y ajoutant ses astuces.

Nous avons personnellement été le mentor de deux nouveaux stagiaires, présentant notre savoir avec toute la pédagogie dont nous sommes capables, acquise dans notre vie passée d'enseignant de français. Cela nous a valu des remerciements officiels lorsque cette fonctionnalité a été mise en place au sein d'IBM, *via* une carte électronique de reconnaissance d'aide à un collègue.

Pour aller plus loin dans le partage des connaissances entre stagiaires, des groupes intra- et extra-IBM ont été créés pour organiser des événements et partager des documents liés au traitement automatique du langage, confidentiels ou non. Des exposés hebdomadaires ont

[10] https://www.ibm.com/support/knowledgecenter/SSPT3X_4.2.0/com.ibm.swg.im.infosphere.biginsights.aqlref. doc/doc/aql-overview.html, dernière consultation le 28 août 2017

également été instaurés, où chacun pouvait présenter scripts (soupçonneur de fautes d'orthographe), concepts (linguistique de corpus), frameworks (Apache Spark) et autres éléments liés à l'informatique ou à la linguistique, utiles ou non pour notre travail sur WASM. Nos supérieurs nous ont autorisés à prendre sur nos heures de travail pour ces « formations » considérées comme utiles à notre culture générale de futurs ingénieurs linguistes.

Nos encadrants sont restés fidèles à eux-mêmes, tranquilles, patients et compréhensifs, quand nous avons effacé le répertoire *Home* de notre ordinateur suite à une erreur de manipulation du Terminal, un `rm -rf *` dans le mauvais dossier. Grâce au cache de notre éditeur de texte et à notre habitude de régulièrement sauvegarder notre travail sur Git, nous n'avions heureusement rien perdu de très important, mais il a fallu passer du temps à réinstaller et paramétrer à nouveau de nombreux logiciels et fonctionnalités, occasion de partager des souvenirs d'expériences similaires devenues sujettes à plaisanteries.

## *4. Méthodes et outils*

WASM utilise une approche à base de lexique et de règles, sans apprentissage automatique pour l'analyse textuelle. Le système extrait les colonnes « texte » et « bio » (cette dernière correspondant à la section « À propos de moi » de blogs, forums, Twitter, etc.), puis procède à une étape de tokenisation suivie d'une désambiguïsation morphosyntaxique à partir de laquelle nous formons des chunks.

### *4.1. AQL*

WASM a recours à InfoSphere® BigInsights™ Text Analytics, un « puissant système d'extraction d'information structurée depuis des textes peu ou pas structurés »[11]. L'extraction en elle-même est réalisée en AQL (Annotation Query Language[12]), langage créé par IBM, que nous utilisions dans le logiciel Eclipse. En voici les éléments principaux à retenir.

#### 4.1.1. Présentation générale

Les données en AQL sont stockées sous forme de tuples, dont un ensemble forme une *vue*. Pour qu'une vue ne pose pas de problème de compilation, ses composants doivent avoir un type, des noms de colonnes un ordre identiques. Bien que les données puissent être des

[11] https://www.ibm.com/support/knowledgecenter/en/SSPT3X_2.1.2/com.ibm.swg.im.infosphere. biginsights. text.doc/doc/ana_txtan_intro.html, dernière consultation le 28 août 2017
[12] https://www.ibm.com/support/knowledgecenter/SSPT3X_4.2.0/com.ibm.swg.im.infosphere.biginsights.aql ref. doc/doc/aql-overview.html, dernière consultation le 28 août 2017

listes scalaires, des booléens, des décimaux, des entiers, une chaîne de caractères ou encore des spans, nous n'utilisions que ces trois derniers.

Les *spans* sont des intervalles entre lesquels se trouvent un ou plusieurs caractères, bien qu'il puisse se trouver des situations dans lesquelles ne rien avoir en [0-0] s'avère utile. Par exemple, la suite de caractères `Élégance et robustesse` correspond au span [0-22], avec `et` en [9-11] : $^{0}$É$^{1}$l$^{2}$é$^{3}$g$^{4}$a$^{5}$n$^{6}$c$^{7}$e$^{8}$ $^{9}$e$^{10}$t$^{11}$ $^{12}$r$^{13}$o$^{14}$b$^{15}$u$^{16}$s$^{17}$t$^{18}$e$^{1}$ $^{9}$s$^{20}$s$^{21}$e$^{22}$. Toute extraction depuis un document se fait sur des spans, permettant de facilement trouver l'emplacement du résultat pour des vérifications, mais aussi de consolider lesdits résultats (cf. *infra*).

AQL travaille chaque document un à un, dont le contenu est représenté dans la vue *Document*, qui sert de base à toute extraction ou sélection. Avec un fichier JSON en entrée, on peut par exemple choisir de ne s'intéresser qu'à certaines colonnes de *Document* (texte, bio, nom de l'auteur) au détriment d'autres (source, URL…).

#### 4.1.2. Déclarations

Tout fichier .aql doit commencer par la déclaration du module auquel il appartient. Le code spécifique à chaque langue de WASM contient cinq modules : *accountTypes*, *demographics*, *behavior*, *sentiment* et *common*, dont l'utilité et le contenu seront détaillés dans la deuxième partie de ce rapport.

```
module demographics;
```

La déclaration principale en AQL est *create*, qui permet de créer une vue, un tableau ou un dictionnaire. Un dictionnaire est une liste d'éléments, dont une correspondance dans une chaîne de caractères nécessite des frontières de token autour d'elle ; un tableau est un ensemble de colonnes transformables en dictionnaires, dont chaque élément d'une ligne permet de retrouver ceux situés sur la même ligne. Les tableaux étant peu utilisés et les dictionnaires simples à utiliser, nous ne détaillerons que les vues.

La création d'une vue nécessite de choisir entre une sélection et une extraction. Sélectionner appelle une, plusieurs ou toutes les colonnes d'une ou plusieurs autres vues déjà existantes (dont *Document*), pour les manipuler ou simplement pour les réordonner. Ainsi, si on souhaite dans un fichier .aql travailler sur le nom de l'auteur de chaque document, on pourra sélectionner la colonne « authorName » dans la vue NomAuteur. Cette vue servira ensuite de base de travail, à la place de *Document*.

```
create view NomAuteur as
  select
    x.authorName as nom
  from
    Document x
;
```

Pour travailler sur du texte en lui-même, on utilise l'extraction d'une expression régulière ou d'un schéma, suite d'expressions régulières ou d'éléments de vues. Si on choisit généralement *Document* comme source, on peut aussi préciser une certaine colonne de vue, comme dans l'exemple suivant :

```
create view FemaleNameCandidateEndInA as
  extract
    regex /.*a/
      with flags 'CASE_INSENSITIVE'
      in 1 token on x.nom
      as match
  from
    NomAuteur x
;

create view GroupePrepositionnel as
  extract
    pattern (<prep.match>) (<dp.match>)
      return
        group 0 as match and
        group 1 as prep and
        group 2 as dp
  from
    Prep prep,
    DP dp
;
```

Notons qu'il est possible d'additionner ou de soustraire des vues les unes aux autres, pour peu qu'elles partagent les mêmes ordre, type et nom de colonne. Les deux vues précédemment créées ont une colonne en commun, *match*. Pour rassembler leurs résultats en une seule vue, il faut se séparer des colonnes *GroupePrepositionnel.prep* et *GroupePrepositionnel.dp* :

```
create view DeuxVuesEnUne as
  (select x.match from FemaleNameCandidateEndInA)
    union all
  (select x.match from GroupePrepositionnel)
;
```

Les éléments créés (vues, dictionnaires et tableaux) sont directement accessibles dans tous les autres fichiers .aql du même module, *Document* faisant figure d'exception car

accessible depuis n'importe quel endroit. Pour permettre aux autres modules d'utiliser une vue, par exemple, celle-ci doit être exportée depuis son module d'origine, puis importée dans les modules la nécessitant.

```
export view NomAuteur;
```

```
import view NomAuteur from module demographics as NomAuteur;
```

Dans le cadre de la programmation, la déclaration d'affichage des résultats est extrêmement utile. Celle-ci affiche le contenu de la vue, chaque colonne correspondant à un de ses champs, avec une colonne supplémentaire indiquant le numéro du document dont est issu le résultat, et chaque ligne présentant un résultat, sans limite de quantités.

```
output view GroupePrepositionnel;
```

Voici un exemple de l'affichage résultant de la ligne ci-dessus.

| **input** | **match (SPAN)** | **prep (SPAN)** | **dp (SPAN)** |
|---|---|---|---|
| test.json_1 | de mon père [40-51] | de [40-42] | mon père [43-51] |
| test.json_1 | à Philippe [74-84] | à [74-75] | Philippe [76-84] |

#### 4.1.3. Filtrage

Il est possible d'affiner les résultats en utilisant des fonctions, soit créées personnellement en Java, soit natives à AQL, ces dernières se divisant comme suit :

- *fonctions de regroupement de plusieurs valeurs* : nombre d'entrées similaires, moyenne, valeur minimale ou maximale, somme, etc. ;
- *fonctions scalaires*, qui renvoient une valeur non booléenne : Chomp, CombineSpans, GetBegin et End, GetLemma, GetLength, GetString, GetText, Left et RightContext, SpanBetween, ToLowerCase, etc. ;
- *fonctions prédicatives*, renvoyant un booléen : Contains, ContainsDict, ContainsRegex, Equals, Follows, FollowsTok, GreaterThan, MatchesDict, MatchesRegex, Not, Or, Overlaps, etc.

Ainsi, si on veut sélectionner parmi les pseudonymes qui finissent en *-a* ceux qui commencent par un A, font plus de trois caractères tout en n'étant pas « Andrea », une des possibilités est la suivante, grâce au mot-clef *where* dans le cas d'une sélection :

```
create view FemaleNameCandidateEndInA02 as
  select
    x.match
  from
    FemaleNameCandidateEndInA x
  where
    GreaterThan(GetLength(x.match),3) and
    MatchesRegex(/a.*/, 'CASE_INSENSITIVE', x.match) and
    Not(Equals(ToLowerCase(GetText(x.match)),'andrea'))
;
```

Lors d'une extraction, c'est le mot-clef *having* qui est utilisé. La principale différence avec *where* est que la restriction ne peut porter que sur le résultat de l'extraction : la correspondance entière ou une de ses sous-parties, délimitées grâce à des parenthèses capturantes. Il est donc impossible d'utiliser un schéma en restreignant les vues qui le composent ; elles doivent être auparavant filtrées en créant une vue sélectionnant les valeurs qui seront utilisées dans le schéma.

```
create view GroupePrepositionnel as
  extract
    pattern <prep.match> <dp.match>
      as match
  from
    Prep prep,
    DP dp
  having
--  Equals(GetText(prep.match),'à') -- problème de syntaxe
    MatchesRegex(/à.*/, match) -- colonne propre à la vue
;
```

### *4.2. Étiquetage morpho-syntaxique d'AQL*

L'extracteur AQL dispose d'un désambiguïseur morpho-syntaxique, dont la qualité laisse à désirer pour la plupart des langues prises en charge. Malgré son appartenance à IBM, les conditions de sa création sont mystérieuses pour quiconque ne travaillant pas dans l'équipe AQL, et toute proposition d'aide par des ingénieurs linguistes chevronnés est refusée.

#### 4.2.1. Étiquettes

AQL dispose de plusieurs listes de catégories grammaticales, qui peuvent varier selon les langues. Pour le français, comme pour toutes les langues indo-européennes en dehors de l'anglais, les catégories sont les suivantes[13] :

[13] https://www.ibm.com/support/knowledgecenter/SSPT3X_4.2.5/com.ibm.swg.im.infosphere.biginsights.aqlref. doc/doc/parts-of-speech.html, dernière consultation le 28 août 2017

| | | | | | |
|---|---|---|---|---|---|
| CC | conj. de coordination | MD | modal | RB | adverbe |
| CD | nombre cardinal | NN | nom commun | SYM | symbole |
| DT | déterminant | NNP | nom propre | UH | interjection |
| IN | prép. ou conj. de subordination | PRP | pronom | UKW | inconnu |
| JJ | adjectif | QT | quantificateur | VB | verbe |
| | | | | WH | mots en *wh-* |

**Tableau 1**. Étiquettes de l'étiqueteur morpho-syntaxique d'AQL pour le français.

On extrait ces catégories grâce à des vues semblables à celle-ci :

```
create view Nom as
  extract
    parts_of_speech 'NN' and 'NNP'
      with language 'fr'
      on d.text as match
  from
    Document d
;
```

À la vue de cette liste, on est en droit de se poser des questions quant au choix des étiquettes. AQL propose 16 catégories pour le français, à comparer aux 42 pour l'anglais ou aux 212 dont dispose le French TreeBank[14], utilisé notamment par l'analyseur de Stanford CoreNLP. Outre le fait que l'existence et l'utilité des mots en *wh-* en français restent à démontrer, des étrangetés apparaissent au premier coup d'œil : prépositions et conjonctions de subordination ne font qu'un, la ponctuation se trouve perdue dans les symboles, les quantificateurs ne font pas partie des déterminants, pas plus que les nombres cardinaux avec les adjectifs. Cette liste discutable l'est encore davantage quand on se rend compte que les classes des cardinaux, modaux et mots en *wh-* ne présentent aucun résultat, ce qui baisse à treize le nombre de catégories effectives pour le français.

### 4.2.2. Analyse morphologique

Suite à cette brève critique des étiquettes, intéressons-nous dans un premier temps à l'analyse morphologique, observée après désambiguïsation, mais dans un premier temps sans la prendre en compte.

[14] French TreeBank - Guide morpho-syntaxique : http://www.llf.cnrs.fr/sites/sandbox.linguist.univ-paris-diderot.fr/files/statiques/french_treebank/guide-morpho-synt.02.pdf, p. 54, dernière consult. le 28 août 2017

L'annexe 1[15] présente quelques-unes des nombreuses erreurs d'étiquetage, observées sur un ensemble de 1000 documents comptant 166 176 tokens totaux. Sur les 15 228 différents tuples token-catégorie, le taux d'erreur s'élève à 27 %, indépendamment donc des fréquences, et malgré le fait que nous ayons accepté certaines acceptions hautement improbables, comme *bref* en nom commun. En voici le tableau récapitulatif :

| **catégorisation par AQL** | **impossibles** | **possibles** | **total** | **taux d'erreur** |
|---|---|---|---|---|
| conj. de coordination | 37 | 16 | 53 | 70% |
| noms propres | 685 | 748 | 1433 | 48% |
| déterminants | 45 | 72 | 117 | 38% |
| pronoms | 67 | 113 | 180 | 37% |
| noms communs | 2525 | 4970 | 7495 | 34% |
| symboles | 71 | 154 | 225 | 32% |
| prépositions et conj. de sub. | 37 | 88 | 125 | 30% |
| adverbes | 145 | 352 | 497 | 29% |
| adjectifs | 506 | 1635 | 2141 | 24% |
| interjections | 2 | 7 | 9 | 22% |
| quantificateurs | 1 | 25 | 26 | 4% |
| inconnus | 0 | 1 | 1 | 0% |
| verbes | 0 | 2926 | 2926 | 0% |
| **cumul** | **4 121** | **11 107** | **15 228** | **27%** |

**Tableau 2.** Étiquetages impossibles (fréquence non prise en compte).

Observons qu'AQL associe une catégorie grammaticale à tout token, sans exception, considérant ceux complètement inconnus comme des adjectifs, noms communs ou propres, apparemment selon leur entourage. Cela explique en partie pour ces classes les taux d'erreur aussi élevés, sans compter la difficulté même pour un annotateur humain de distinguer certains noms propres d'un pur charabia. Concernant les conjonctions de coordination, tous les *que* sont fautivement inclus parmi elles, ce qui explique la majeure partie des erreurs de cette classe.

Grâce à l'apprentissage automatique, certains mots mal orthographiés, sans accents ou avec des coquilles se voient attribuer la bonne étiquette. Néanmoins, certains mots correctement écrits, dont on peut pourtant relativement aisément dresser la liste des natures possibles, sont parfois classés dans des catégories impossibles en français : *que* peut donc figurer parmi les conjonctions de coordination, *depuis* être avec les noms communs, *beaucoup*

[15] Annexe 1, *Extrait de résultats fautifs du désambiguïseur morphosyntaxique d'AQL*, p. 37

les adjectifs, *hein* les prépositions et conjonctions de subordination, etc. Les performances variables de l'apprentissage automatique associées à l'absence d'erreur pour les verbes (hors désambiguïsation) nous laisse dubitatif quant à l'algorithme d'étiquetage.

#### 4.2.3. Désambiguïsation morpho-syntaxique

AQL semble considérer qu'un token est une simple suite de caractères entre deux symboles de ponctuation, sans donc possibilité d'unités polylexicales, ce qui implique des erreurs de tokenisation comme *ainsi / que*. Cela, ajouté au fait que de nombreuses étiquettes pourtant impossibles sont acceptées, explique en partie les mauvais résultats de l'analyse morpho-syntaxique.

En annexe 2[16] se trouve la matrice de confusion de l'étiquetage du texte de présentation du master Industries de la Langue de l'Université Grenoble-Alpes accompagnée des tests (vrais positifs, vrais négatifs, etc.) pour chacune des catégories grammaticales ; *idem* en annexe 3[17] avec des messages recueillis sur un forum de discussion. Ces deux documents, de quelque 500 tokens chacun, permettent de comparer les performances de l'étiqueteur en fonction la qualité *a priori* du texte, soit littéraire avec quelques coquilles possibles, soit d'un style proche de l'oral avec tout type de fautes (orthographe, conjugaison, syntaxe, typographie et tant d'autres). Nous avons ignoré les tokens du type *c* (*c'est*) ou *g* (*j'ai*) car ils font l'amalgame de deux tokens aux catégories distinctes, et seule une catégorie *ad hoc* conviendrait.

Voici ci-dessous une autre perspective des résultats, mis côte à côte. Concernant le rappel, il est globalement similaire entre le texte travaillé et non travaillé. Seuls les rappels des prépositions ou conjonctions de subordination (IN) et des adjectifs (JJ) présentent une grande différence entre les deux documents à l'avantage du texte travaillé, mais notons les adverbes (RB) qui sont mieux repérés dans le texte au style relâché. En dehors de cela, les rappels sont relativement similaires, la différence pour les verbes se situant dans les participes passés, qui ne constituent pas une erreur trop importante.

Pour ce qui est de la précision, celle du texte travaillé est bien supérieure en comparaison à l'autre, avec des scores respectivement de 90 % et 75 %. Dans le détail des catégories, les conjonctions de coordination (CC) souffrent d'une faible précision à cause de *que*, tout comme les adverbes (RB), souvent confondus avec des prépositions ou des adjectifs.

---

[16] Annexe 2, *Matrice de confusion de l'étiquetage désambiguïsé d'un texte travaillé*, p. 38

[17] Annexe 3, *Matrice de confusion de l'étiquetage désambiguïsé d'un texte non travaillé*, p. 39

Étonnamment, les déterminants (DT) sont mieux identifiés le texte relâché, à l'inverse des noms communs (NN) ; cela s'explique par le fait qu'AQL identifie les nombres comme des noms, et qu'ils sont bien plus utilisés dans ce type d'écrits que dans un texte où l'on est attentif.

| | Précision | | Rappel | |
|---|---|---|---|---|
| | Travaillé | Non travaillé | Travaillé | Non travaillé |
| CC | 0.75 | 0.59 | 1.00 | 0.95 |
| CD | 0.00 | 0.00 | 0.00 | 0.00 |
| DT | 0.89 | 0.97 | 0.97 | 1.00 |
| IN | 1.00 | 0.98 | 0.86 | 0.70 |
| JJ | 0.77 | 0.68 | 0.77 | 0.68 |
| NN | 0.93 | 0.57 | 0.93 | 0.86 |
| NNP | 1.00 | 0.00 | 0.64 | 0.00 |
| PRP | 0.85 | 0.93 | 0.73 | 0.91 |
| RB | 0.67 | 0.90 | 0.73 | 0.80 |
| SYM | 1.00 | 1.00 | 1.00 | 0.99 |
| VB | 0.73 | 0.95 | 1.00 | 0.90 |

**Tableau 3.** Précision et rappel du désambiguïseur d'AQL selon le caractère « travaillé » du texte.

La qualité de l'analyseur morpho-syntaxique varie donc selon les catégories grammaticales, mais dépend également du type de texte étudié. Malheureusement, les messages des internautes sont bien trop souvent criblés de fautes pires les unes que les autres, s'éloignant beaucoup du français écrit standard et n'aidant pas les linguistes-informaticiens dans leurs fouilles de textes.

## Mise en œuvre du cahier des charges

Notre rôle pendant ces six mois de stage a consisté à améliorer la partie française de WASM. Si la plupart des modules disposaient déjà de la majeure partie du code, il manquait pour certains d'importants blocs, et tous nécessitaient des corrections à différents niveaux. Comme nous l'avons déjà mentionné, nous sommes la première personne ayant le français pour langue maternelle à travailler sur ce produit, et nous nous sommes employé à mettre notre connaissance linguistique au service de l'équipe dans l'optique d'augmenter encore les performances de WASM. Nous avons aussi nettoyé et uniformisé l'ensemble du code, qui contenait des styles différents (appellations, indentation…) et des vues utilisées nulle part.

### *1. Nos découvertes en AQL*

Avant d'entrer dans le vif du sujet, nous allons présenter deux découvertes que nous avons personnellement faites concernant le langage AQL, qui permettent respectivement à tout développeur de se simplifier la tâche pour les schémas, et aux linguistes voulant sélectionner des apostrophes de le faire sans qu'aucune ne leur échappe.

#### *1.1. Les vues facultatives des schémas*

À notre arrivée, on nous avait prévenu d'un des défauts d'AQL, qui obligeait à créer de nombreuses vues au lieu d'en indiquer certaines comme facultatives dans un schéma. La structure de vue suivante est utilisée depuis notre découverte, qui relève presque de la sérendipité :

```
create view GroupeNominal1 as
  extract
    pattern <det.match> <adj.match>? <nom.match>
      as match
  from
    Determinant det,
    Adjectif adj,
    Nom nom
;
```

Auparavant, l'absence de tout élément facultatif dans l'ensemble du document créait une vue vide. On utilisait deux vues (l'une avec un adjectif et l'autre sans), unies par la suite en une seule. Ainsi, dans le tweet « Tu es bien la fille de ton père ! », *la fille* n'était pas reconnue, malgré sa compatibilité apparente avec le schéma, car aucun adjectif ne figure dans le document.

Cet exemple est simple, mais il peut arriver que les schémas fassent appel à autrement plus de vues, avec plusieurs champs facultatifs. Cela permet d'économiser de nombreuses lignes de code et créations de vues, dont des quantités trop importantes n'aident pas la bonne compréhension et maintenance du code. Quant à la raison de l'ignorance de l'équipe vis-à-vis de ces vues facultatives, elle s'explique tout bonnement par le manque de communication entre l'équipe de maintenance du langage et celles qui l'utilisent.

### *1.2. Les trois apostrophes*

En cherchant la raison pour laquelle certaines vues n'affichaient pas les résultats attendus, nous nous sommes rendu compte que les documents ne présentaient pas une seule apostrophe, mais plusieurs, considérées par Unicode comme des entités distinctes. Nous avons donc modifié dictionnaires et expressions régulières en ajoutant la possibilité d'avoir ces autres apostrophes[18], mais si les dictionnaires ne sont pas gênés par cela, une des trois apostrophes utilisées pose un problème dans les expressions régulières. Voici les vues créées (à la mise en page simplifiée) pour tester ces apostrophes, avec les résultats en dessous.

```
create view A1 as extract regex /l['’']/ on d.text as m from Doc d;
create view A2 as extract regex /\bl['’']/ on d.text as m from Doc d;
create view A3 as extract regex /l['’']\b/ on d.text as m from Doc d;
create view A4 as extract regex /\bl['’']\b/ on d.text as m from Doc d;
create view A5 as extract regex /l['’']/ on 1 token in d.text as m from Doc d;
create view A6 as extract regex /l(['’]\b|')/ on d.text as m from Doc d;
create view A7 as extract regex /\bl(['’]\b|')/ on d.text as m from Doc d;
```

| | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
|---|---|---|---|---|---|---|---|
| ’ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| ' | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| ’ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ |

**Tableau 4.** Tests des apostrophes dans une expression régulière (vert = repérée ; rouge = non repérée).

La restriction à un seul token (A5) bloque toutes les apostrophes finales, alors que seule la troisième est la seule à ne pas être reconnue par une séparation de token `\b` la suivant (A3 et A4). Selon les situations, nous utilisions donc la structure de A5 ou A6.

## *2. Common*

Le module *Common* contient toutes les vues ayant un lien avec la grammaire, potentiellement nécessaires à tous les autres modules (par conséquent *a priori* non spécifiques

[18] https://fr.wikipedia.org/wiki/Apostrophe_(typographie), dernière consultation le 28 août 2017

à aucun d'entre eux). C'est là que sont extraites les classes grammaticales, créées les vues sélectionnant chacune des classes nous intéressant, définis les chunks, listés certains pronoms personnels ou possessifs, etc. De ce fait, presque toutes les vues de *Common* sont exportées, prêtes à être importées dans les autres modules.

### *2.1. Travailler avec l'étiqueteur morpho-syntaxique d'AQL*

Comme nous l'avons souligné dans la première partie de ce présent rapport, l'étiquetage morpho-syntaxique d'AQL n'est pas d'une fiabilité à toute épreuve et nous avons dû recourir à différents stratagèmes pour pallier cela.

Tout d'abord, nous avons établi des dictionnaires pour les catégories grammaticales fermées dont nous avions besoin : déterminants, nombres cardinaux, pronoms personnels clitiques, prépositions, etc. Certaines classes ont été laissées de côté, comme les conjonctions de subordination, car elles ne sont pas importantes dans notre traitement linguistique. Pour ce qui est des classes ouvertes, dont nous avons grandement besoin, nous avons établi pour chacune une liste de tokens considérés à tort par AQL comme étant de la catégorie en question, ajoutant chaque entrée dans la liste des tokens qui auraient dû être considérés comme lui appartenant. Les faux positifs étaient donc soustraits aux résultats extraits grâce à la déclaration *extract parts_of_speech*, et les faux négatifs étaient ajoutés.

Notons que toute entrée de dictionnaire comprenant apostrophe, ligature ou caractère comprenant un diacritique était dupliquée pour permettre de prendre en compte sa variabilité (*idem* pour les expressions régulières) : *aujourd'hui* se voit ainsi adjoindre *aujourd hui* et *aujourdhui*, *aujourd'hui* et *aujourd'hui* (avec deux autres types d'aspostrophes) ; *sœur* nécessite la création de *soeur* ; *hétérogénéité*, celle d'*heterogeneite*.

Dans ce dernier cas, l'expression régulière serait la suivante : `/h[ée]t[ée]rog[ée]n[ée]it[ée]e?/`, en prenant en compte la possibilité qu'un e soit ajouté. Pour un dictionnaire, l'affaire se complique. Nous avons dans un premier temps entré *hétérogénéité*, *heterogénéité*, avec toute la combinatoire. Cela s'est rapidement avéré coûteux, peu maintenable et peu utile, la précision primant sur le rappel. Nous avons donc finalement opté, dans la plupart des cas, pour deux entrées : l'une bien orthographiée, l'autre « simplifiée ».

### *2.2. Améliorations majeures*

Parmi les problèmes les plus importants que nous avons résolus, citons les chunks, la négation et les indicateurs de première personne, problèmes essentiellement dus à l'absence de linguiste francophone natif.

Ce que nous appelons *chunk* est un groupe structuré de tokens, différent du *groupe* en ce qu'il n'est pas récursif, afin d'éviter de trop nombreux faux positifs, bien qu'AQL ne permette pas de récursivité (ce qui explique sa vitesse). Les chunks existants produisaient de bons résultats en terme de précision, mais leur rappel était assez faible. Nous avons appliqué notre savoir linguistique pour améliorer tous ces chunks, qui présentent maintenant des structures plus englobantes, comme l'ajout des clitiques dans les chunks verbaux.

Pour ce qui est de la négation, le *ne* explétif revêtait parfois une importance plus grande que l'autre élément de la négation, accentué (*pas*, *jamais*, etc.). De nos jours, il est pourtant en voie de disparition à l'oral, style dont relèvent souvent les textes issus des médias sociaux. Ne pas considérer la possibilité que ce *ne* puisse être absent créait un grand nombre de faux négatifs, dont le nombre est maintenant considérablement réduit.

Enfin, la restriction aux sujets *je* et *nous* pour la première personne du singulier laissait à désirer, utilisée pour vérifier que le verbe est bien relatif à l'auteur (sans possibilité de vérifier la présence de guillemets, souvent utilisés hors du discours direct, parfois même absents de celui-ci). Nous avons donc fortement agrandi la liste de ces termes, que nous ne pouvons pas dévoiler pour des raisons de confidentialité mais qui viendraient à l'esprit de tout spécialiste se penchant sur la question.

### *2.3. Résultats*

Par manque de temps, nous n'avons pas pu réaliser de réelle évaluation sur chacun des points importants du code, mais voici les différences de quantité de résultats, calculées sur un même ensemble de mille documents. La seule diminution apparente, la baisse du nombre de chunks adverbiaux, est due aux opérations de redéfinition de ce qu'est un adverbe, qui comprenait auparavant de nombreuses erreurs. AP correspond aux chunks d'adjectifs, DP aux chunks commençant par un déterminant, VP par un verbe, et XP étant une vue regroupant tous les chunks le plus larges possible.

| | AdvP | AP | DP | VP | XP |
|---|---|---|---|---|---|
| 2016 | 35 577 | 47 850 | 230 097 | 25 127 | 429 239 |
| 2017 | 34 233 | 48 819 | 233 551 | 70 038 | 494 824 |
| amélioration | -3,78% | 2,03% | 1,50% | 178,74% | 15,28% |

**Tableau 5.** Taux d'amélioration des principales vues du module *Common*.

L'étiqueteur morpho-syntaxique représente la principale faiblesse de WASM, et le passage après notre stage vers le système de Stanford CoreNLP permettra sans aucun doute

d'encore meilleurs résultats, surtout concernant les classes ouvertes que sont les noms communs, les adjectifs et les adverbes.

## *3. Demographics*

Le module *Demographics* cherche à découvrir le sexe de l'auteur du message en question, si cette personne est mariée et si elle a des enfants, et ce tant dans le corps du texte que dans la biographie de l'auteur, le cas échéant. Si le code existait déjà pour fouiller le texte, celui pour le champ « Biographie » restait entièrement à créer.

### *3.1. Description*

La recherche d'informations sur le caractère marié de l'internaute porte essentiellement sur son message et sa biographie. Une vue existe pour tester le nom d'utilisateur, mais elle est presque inutile pour le français, présentant une probabilité de capture de résultats presque nulle. Dans le texte et la biographie, on cherche des structures simples comme *mon mari*, en filtrant celles qu'on ne souhaite pas retenir, car non représentatives de la situation actuelle de la personne. Nous avons largement amélioré ces vues, permettant des résultats de meilleure qualité.

Concernant les éléments indiquant si la personne a un enfant ou non, il en va de même que pour la recherche précédente, avec une vue dont l'espoir d'apporter de l'eau au moulin du système est très bas. Un ensemble d'expressions régulières et de schémas tentent de repérer un maximum de structures linguistiques caractéristiques du fait que le référent a un enfant. Nous les avons donc revues et perfectionnées, filtrant notamment des résultats probables mais erronés.

Pour détecter le sexe de l'auteur du message, on dispose de son pseudonyme en plus de ce qu'il a écrit. Si son nom d'utilisateur contient un nom et un prénom, notre dictionnaire de prénoms peut le trouver grâce à la séparation de tokens entre les deux éléments. Néanmoins, si le pseudonyme est *ClaudiaRieu04*, le dictionnaire est inefficace car ne fonctionne qu'avec des tokens. Quand bien même cela ne serait-il pas le cas, de trop nombreux faux positifs surviendraient : des prénoms courts ou étrangers peuvent être aisément contenus dans d'autres mots (*suppléant, janvier, Andréa*), que la capitalisation n'aide pas à identifier du fait de son utilisation stylistique variée, en particulier dans les pseudonymes.

Les prénoms ne sont donc pas cherchés à l'intérieur des tokens, mais dans le cadre de correspondance totale uniquement, nécessitant de recourir à des expressions régulières pour

les autres indices relatifs au sexe de la personne, comme son titre (*KingJack*), la façon dont il se considère dans la famille (*mamie*), etc. Nous avons guère modifié les vues relatives à cela, ne voyant aucun moyen de les améliorer, mais nous avons complété et filtré les expressions régulières afin de réduire le nombre de faux positifs.

Les autres sources se situent dans le texte et la biographie, cette dernière étant une copie des vues portant sur le texte, adaptées aux spécificités des biographies, principalement en ce qui concerne la syntaxe. Outre les termes habituels comme *Je suis une femme*, une grande source d'amélioration a été l'établissement de dictionnaires complétant ceux dont on disposait déjà dans le module et grâce auxquels on pouvait déterminer le sexe de la personne. En observant les structures spécifiques aux biographies sur Twitter, nous avons observé certaines régularités, que nous avons mises à profit pour établir des dictionnaires de termes qui indiquent sans ambiguïté l'appartenance à un sexe ou à l'autre.

Enfin, une dernière possibilité pour identification du sexe de l'auteur est de recouper les informations obtenues grâces aux autres vues du module, celles indiquant si la personne est mariée ou a un enfant. En effet, le texte prouvant ces derniers éléments peut contenir certains mots sans équivoque, comme dans *Je suis le papa d'un petit bébé*.

### *3.2. Résultats*

Les résultats détaillés pour le module *Demographics* se trouvent en annexe 4[19], présentant une hausse globale de 10 % du nombre d'éléments trouvés suite à notre travail. Dans le détail, on observe des augmentations de 11 % pour les hommes mais seulement 3 % pour les femmes, alors que celle des parents est de 16 % et celle des personnes mariées s'envole à 156 %. De précédentes analyses poussées des résultats ont montré que l'explication des rares cas où une baisse avait lieu tenait en réalité à des faux négatifs de l'ancienne version du code.

Ce module est le premier sur lequel nous ayons travaillé. Si les améliorations sont visibles, elles peuvent encore être perfectionnées grâce à quelques simples astuces et à une certaine structure à laquelle nous n'avons pensé qu'après avoir commencé le module suivant, et que nous n'avons pas eu le temps d'implémenter.

19 Annexe 4, *Comparatif des résultats de Demographics avant et après nos modifications*, p. 40

## *4. Behavior*

Le module *Behavior* vise à déterminer le rapport qu'entretient l'auteur du message avec l'objet qui intéresse l'utilisateur de WASM. Autrement plus compliqué que *Demographics* car nécessitant des chunks de bonne qualité et donc un travail parallèle sur *Common*, ce module nous a fait nous confronter à des problématiques et donc des vues relativement complexes, nous permettant d'appréhender AQL dans sa quasi-totalité.

### *4.1. Description*

Cinq profils d'utilisateurs ont été déterminés pour ce module : possesseur ou utilisateur (*user*), utilisateur potentiel (*prospective user*), ancien utilisateur, potentiel ou avéré (*potential churner*), personne qui recommande (*recommender*) ou qui déconseille (*detractor*).

Nous basant sur les vues existantes que nous avons améliorées, nous avons ajouté d'autres structures, voire beaucoup pour ce qui est des anciens utilisateurs, sans néanmoins guère de résultats probants. Parmi les pistes pour obtenir davantage de résultats se trouvent la fouille du champ « Biographie » ainsi que la simple recherche hors chunks, qui sont étonnamment nullement prises en compte dans le code existant, et que nous n'avons pas pensé à exploiter, bien que le temps nous eût manqué de toute façon.

Le plus gros de notre travail a consisté à faire la liste des verbes possibles pour chaque structure, puis à les conjuguer, à faire vérifier la présence ou l'absence de négation, etc. Pour réaliser nos tests, nous devions modifier non seulement le champ « Texte » du document, mais également les spans du mot-clef correspondant à un des *sujets* qui intéressent l'utilisateur, ce qui rendait la tâche autrement plus fastidieuse que pour le module *Demographics*, qui n'est pas lié aux *sujets*.

### *4.2. Résultats*

Les résultats détaillés pour le module *Behavior* se trouvent en annexe 5[20], présentant une hausse globale de 24 % du nombre d'éléments trouvés. Dans le détail, on observe des augmentations de 24 % pour les utilisateurs de l'entité sur laquelle porte la recherche, 37 % pour les personnes qu'elle intéresse, et 33 % pour les utilisateurs qui se sont détournés ou sont susceptibles de le faire. Notons la baisse de 25 % pour les personnes recommandant l'usage ou l'appropriation de l'objet, et l'absence totale de résultats pour les détracteurs, quelle que soit la version du code.

[20] Annexe 5, *Comparatif des résultats de Behavior avant et après nos modifications*, p. 41

Ces résultats s'expliquent notamment par la faiblesse du nombre de résultats (211 pour l'ancien code, 262 pour l'actuel, sur quelque 10 000 documents totaux), qui font très vite varier les pourcentages en cas de léger changement, mais surtout par le fait que nous étions trop près du code, ne pensant pas suffisamment globalement. D'autres documents présentaient des résultats prouvant la non nullité de la recherche des détracteurs et que la nouvelle version du code permet parfois de découvrir davantage qu'auparavant de personnes recommandant un produit.

De manière générale, la recherche du comportement des internautes peut (et devrait) être encore améliorée. Cela nécessiterait un travail de linguistique de corpus, analysant les messages un à un, repérant la façon dont on s'exprime pour indiquer qu'on possède un produit, qu'on le recommande, etc. Nous avons passé plus de temps que nous ne l'aurions souhaité sur ce module, et ne pouvions nous permettre de nous attarder davantage. Bien qu'ils soient positifs dans l'ensemble, ces résultats mitigés ne nous satisfont pas, mais l'impossibilité que nous fassions mieux dans les délais impartis relativise cette frustration.

## *5. Interests*

Le code de la partie *Interests* est contenue dans le module *Behavior*, mais il s'agit du seul lien qui les unit, tant ils sont différents. Comme son nom l'indique, *Interests* cherche à identifier les centres d'intérêt de l'utilisateur. Rien n'existait pour le français, aussi nous sommes-nous inspiré du code anglais pour l'adapter à la langue de Corneille. Notons également que deux autres stagiaires ont travaillé sur cette partie peu après que nous avons fini notre adaptation vers le français, et que des questions quotidiennes créaient des réunions incessantes et des améliorations constantes, pour arriver après bien trop de temps (le module d'analyse de sentiments étant plus important) au code actuel. Néanmoins, sans elles, la qualité générale serait légèrement moindre, et nous aurions baissé les bras face à un algorithme qui nous semblait trop difficile à mettre en place, que nous avons finalement réussi à implémenter, et qui a par la suite servi de modèle pour toutes les langues.

### *5.1. Description*

Le cœur de la partie *Interests* se situe dans un tableau, qui liste des sources d'intérêt, leur catégorie et la façon dont elles se comportent dans la phrase : sont-elles suffisamment spécifiques pour que leur simple occurrence fasse comprendre que l'auteur du message est intéressé par elles (après filtrages), ou bien nécessitent-elles des schémas particuliers ?

Le code est presque similaire pour le texte et la biographie, cherchant les sources d'intérêt par leur fréquence ou leur apparition au sein de schémas. Ceux-ci sont simples, aussi avons-nous fait de notre mieux pour les compléter le plus possible, sans prendre le risque de baisser la précision. Nous ne pouvons révéler ce en quoi consistent les grandes améliorations que nous avons apportées, mais nous pouvons expliquer la façon dont nous avons porté le tableau à plus de 13 000 entrées.

Nous servant du tableau anglais comme base, nous avons traduit une à une chaque entrée, sans oublier de vérifier son aspect spécifique ou général. Nous avons ensuite développé la liste en ajoutant les variantes possibles (diacritiques, fautes d'orthographe, etc.), ainsi qu'en ayant recours à Wikidata[21], une base de connaissances utilisée par les sites de la Wikimedia Foundation, dont Wikipedia. Du fait de l'utilisation que nous faisions de ces données, la licence nous permettait d'en récupérer une partie, ce que nous avons pu faire grâce à une interface de requêtes[22] où nous codions en SPARQL.

Pour obtenir une liste des top models, nous entrions ainsi la requête suivante, dont nous exportions les résultats avant de les manipuler pour n'en extraire que certains éléments.

```
SELECT ?item ?itemLabel
WHERE {
  ?item wdt:P106 wd:Q865851.
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "[AUTO_LANGUAGE],fr".
  }
}
```

Dans l'interface utilisateur, le client peut filtrer les résultats selon les intérêts des internautes. Comme mentionné, nous avons étendu le tableau en créant une entrée *creer des bijoux* après *créer des bijoux*. Cela augmente le rappel mais oblige l'utilisateur à cocher ou décocher toutes les variantes d'une même entrée (si elles apparaissent toutes deux dans les documents), qui n'apparaissent parfois pas côte à côte. Pour parer à cela, nous avons ajouté au tableau une colonne avec la forme standard du concept, commune à toutes les variantes voire à des mots différents qui expriment une réalité similaire, comme *fabriquer des bijoux* ou *création de bijoux*, pour reprendre l'exemple précédent. L'utilisateur n'a plus qu'une entrée à cocher ou décocher, *fabrication de bijoux,* qui correspond parfois de très nombreuses entrées du tableau.

[21] https://www.wikidata.org, dernière consultation le 28 août 2017
[22] https://query.wikidata.org, dernière consultation le 28 août 2017

En voyant la liste des formes normalisées, on peut se demander ce que certaines entrées peuvent apporter à l'utilisateur. Y a-t-il un intérêt réel à savoir que l'internaute en question parle de tel homme politique ou joue à tel jeu vidéo plutôt qu'à un autre ? Si cela est source d'intérêt, le client peut créer une configuration spécifique. Proposer un maximum d'intérêts potentiels peut éventuellement s'avérer utile, mais il est beaucoup plus probable qu'un degré moindre de précision le soit davantage : si on aime un musicien, on s'intéresse aussi à son style de musique, qui n'est pas moins important que la personne en elle-même. De même, les catégories auxquelles doivent appartenir les sources d'intérêts peuvent parfois sembler inutiles si on ne dispose pas de tous les intérêts de tous les utilisateurs, certains en parlant aisément et d'autres non. Il nous semble donc que l'importance accordée à ces intérêts doit être relative, utilisée comme informations bonus et non comme base pour un plan marketing.

Étant donné que cette partie du code n'existait pas auparavant, nous ne pouvons pas réaliser de comparatif, et une évaluation serait trop longue. Néanmoins, nous sommes extrêmement confiant quant à la qualité de cette partie-ci, qui ne peut être améliorée qu'en rappel, les rares erreurs de précision venant d'une utilisation étonnante de la langue par l'auteur du message.

## *6. Sentiment*

L'analyse de sentiments est la raison d'être de WASM, les autres modules venant se greffer à celui-ci pour lui ajouter des fioritures et se démarquer de la concurrence. Ce module vise à déterminer, à partir du *snippet* dans lequel le mot-clef apparaît, la polarité des sentiments qui s'y rattrachent : positif, négatif ou neutre.

### *6.1. Description*

La détection de sentiments de WASM ne se fait pas par apprentissage automatique, qui nécessiterait trop de données pour être efficace et ne serait pas aisé à entretenir, mais par règles. Il faut donc des dictionnaires de mots-clefs indiquant du positif ou du négatif ; la liste des entrées positives comptait 1666 entrées, celle des négatives 2287. Nous les avons portées respectivement à 14 483 et 19 088.

Pour ce faire, nous avons commencé par ôter les entrées erronées, ayant parfois plutôt leur place dans l'autre liste. Ensuite, nous avons traduit une à une les entrées des listes anglaises plutôt que de nous fier à un traducteur automatique, ce qui nous permettait de penser à des synonymes en visualisant la page de propre à chaque mot traduit. Enfin, nous avons

utilisé DISCO[23], une application Java proposant des mots sémantiquement proches de ceux fournis en entrée, à partir d'un corpus de messages authentiques et non retravaillés, permettant notamment de recueillir les fautes les plus fréquentes. Ce brillant outil nous a permis de substantiellement accroître les dictionnaires, avant de réutiliser les mêmes expressions régulières que pour *Interests*, ajoutant les variantes de chaque entrée et imaginant les diverses probables fautes d'orthographe non proposées par DISCO, qu'il faudrait pouvoir néanmoins détecter.

Une collègue stagiaire a développé une fonction permettant d'utiliser des expressions régulières dans les dictionnaires, ce qui permet de grandement en réduire le nombre de lignes et d'en faciliter la modification. Néanmoins, nous avions déjà dressé nos listes et n'avions pas le temps de tout modifier, sachant que le rendu effectif final serait le même. De plus, créée pour les déclinaisons du russe, cette fonction ne porte que sur les fins de mots alors que le français nécessiterait la possibilité d'utiliser des expressions régulières sur toute partie du mot, ce qui complexifierait chaque entrée et aurait probablement un impact sur la rapidité du traitement, ce qui n'est pas le cas avec la version actuelle.

Quant au code, nous n'avons eu besoin d'effectuer que peu de changements, celui-ci étant globalement de très bonne qualité. Il cherche à afficher le plus grand groupe de mots possibles auquel on peut attribuer une polarité positive ou négative, allant donc au-delà d'une simple correspondance d'entrées du dictionnaire dont le contexte est inspecté pour vérifier la présence ou l'absence de négation. Certains enchaînements de groupes de même polarité en créent un plus grand, de même pour certains groupes de polarités différentes, selon la nature des chunks en question.

Partout où des vues concernaient des groupes nominaux, nous avons créé un équivalent pour les groupes prépositionnels, distincts d'une simple préposition mais utiles dans certains cas pour proposer le groupe le plus grand possible plutôt qu'une série de groupes à un seul token.

Parmi les mots porteurs de sentiment se trouvent les termes latents, qui dépendent d'un modifieur pour exprimer du positif ou du négatif. Ainsi, si *consommation d'énergie* est neutre, il devient positif quand associé à *faible*, mais négatif avec *importante*. Nous avons amélioré ces dictionnaires et incorporé les verbes dans le code, qui avaient été préparés mais non implémentés.

[23] http://www.linguatools.de/disco/disco_en.html, dernière consultation le 28 août 2017

Nous avons également été chargé de créer les règles concernant les comparatifs. Si A est plus beau que B et que nous nous intéressons à B, le système doit indiquer que la polarité de *beau* est négative pour B. Cela n'existait pas pour le français, et c'est le code italien qui nous a aidé. En effet, les codes anglais et allemands ne correspondaient pas à notre vision du problème, et y répondaient par des voies qui ne nous convenaient pas. La version italienne des comparatifs ressemblait trait pour trait à ce que nous aurions écrit si nous avions dû créer le code *ex nihilo*. En réalisant des tests, nous nous sommes rendu compte d'erreurs dans le fichier italien, que nous avons donc modifié pour éviter qu'un tiers des résultats dus à la comparaison soient faux.

### *6.2. Résultats*

Les résultats du module d'analyse de sentiment se trouvent en annexe 6[24]. La hausse globale s'élève à 73 %, plus précisément 65 % pour les sentiments positifs et 91 % pour les négatifs. Ces chiffres correspondent à nos attentes, étant donné que tous les changements que nous avons effectués portaient sur le rappel en plus de la précision. Le seul moyen de faire encore progresser ces résultats serait de travailler à temps plein sur l'enrichissement des listes, le code n'ayant probablement besoin que d'éventuelles modifications mineures.

[24] Annexe 6, *Comparatif des résultats de Sentiment avant et après nos modifications*, p. 42

# Conclusion

À travers ce stage de six mois au sein de l'équipe de *IBM Watson Analytics for Social Media,* nous avons fait l'expérience du travail dans une grande entreprise, pionnière dans de nombreux domaines liés à l'informatique et les nouvelles technologies. Nous avons côtoyé de grands professionnels, certains disposant d'une vingtaine de brevets à leur actif, mais également des stagiaires informaticiens, linguistes ou informaticiens-linguistes, tous issus de cultures et universités différentes, proposant une équipe diverse collaborant de manière efficace.

Après avoir découvert un nouveau langage formel, l'AQL, nous avons amélioré chaque module, de plus en plus complexes, nécessitant des réflexions et prises de recul toujours plus grandes. Si nous disposions de davantage de temps, nous reviendrions sur nos premières lignes de code pour les rendre plus simples et efficaces, dans les modules *Demographics* et *Behavior*.

Ces différentes tâches de fouille de texte nous convenaient parfaitement, à nous qui sommes particulièrement friands d'expressions régulières. Nous avons pris un réel plaisir à travailler sur le code de WASM, tout comme le dévoilement progressif des plus subtils aspects d'AQL a été intéressant à suivre, d'autant plus qu'on ne semble jamais vraiment avoir tout fini d'apprendre dessus, même après six mois intensifs.

En plus des modules présentés, au cours desquels nous avons pu utiliser une grande partie de nos connaissances fraîchement acquises à l'université (expressions régulières, SPARQL, Python, Java, bash, etc.), nous avons pu avoir un aperçu de la façon dont fonctionne un système de détection de courrier indésirable et de classification d'auteur (proche de *Demographics*, avec principalement des dictionnaires qu'il a fallu agrandir), mais nous avons aussi été amené à effectuer des évaluations, avec lesquelles nous sommes maintenant bien plus à l'aise que par le passé.

En somme, ce fut une expérience vraiment enrichissante, qui convenait tout à fait à notre profil ainsi qu'à la présentation initiale du stage. Nous avons fait de belles rencontres, nous avons fait l'expérience du télétravail, vers lequel nous nous dirigeons dans un avenir très proche, ainsi que celle de l'appartenance à une équipe d'ingénieurs de haut niveau dans une entreprise de pointe. Nous sommes dans l'ensemble satisfait de notre travail, estimant avoir mené à bien la tâche qui nous avait été confiée, tout en restant conscient qu'il est toujours possible de faire mieux mais que seule l'expérience permet d'avancer plus vite.

# Table des annexes

# Annexe 1
# Extraits de résultats fautifs de l'analyseur morphosyntaxique d'AQL

| | |
|---|---|
| CC - conj. de coordination | toutes conjonctions, pas uniquement de coordination (principalement *que*) |
| CD - nombres cardinaux | 0 résultat |
| DT - déterminants | *- + 257k 300m a aas alerte après autres both ça çà chaque contre court en entre es étroite fin forte meilleurs même moi neuf où perdu peu plein puis rempli rendu si soit t tel telle telles tous toutes trop vu y* |
| IN - prépositions et conj. de subordination | *ab ad ah bah eh es euh foutre ha hein hello hum lol lui model na o ok ouais plupart point post priori proche quo* |
| JJ - adjectifs | *\| ��€ 10j 10k 273e2038b3ee6381cb407b6d 3a3 8millions 9aeuarar admitted ag al apres archétype aricept arimidex asteroide avenue avocat baclofène beaucoup belgique biais c ça cb742b1 ccadrcs cnil copain créancier créanciers cri cul demain depuis deux jai jaiden khomri membres smartphone voire xanax y* + suites de chiffres + mots inconnus |
| MD - modaux | 0 résultat |
| NN - noms communs | *abaisse abondés activable actuel aigue anticoncurrentiel antidiabétiques apparait après c'est-à-dire cassé décotées dédiaboliser dématérialisé densifie depuis déresponsabilisant Wgr3MhkNhn* + ponctuation + noms propres + mots inconnus |
| NNP - noms propres | mots inconnus avec majuscule initiale |
| PRP - pronoms | *+ 231cv a ab ah aipas après au autres avec bordeaux chaque demi deux devant entre es frites hue mal mieux mon même n' n26 net non o orange pas point post pour pourquoi puis punaise sans sauf si tapie trois uns vite voilà vu* |
| QT - quantificateurs | aucun mot ne pouvant être un quantificateur, dont les cardinaux |
| RB - adverbes | *a \| 200e 2eme 2fa 30euros 360no2 3h 4ans 5000e 72h 8k ab ah arriteè4g aucun aucune b2c2 ce certain certains cinq claranova dans déjà deux directeur es euh ex fier grâce hein hélas hello hors leur lol lui ma matin merde mes minima moi mon n4dz na nos notre nul oh or où outre-mer pendant personne pouce quatre que quel quelle quelles quelques quoi s' sa sans sauf sept sh3ng soi-disant son sous suivant sur t thedune1 tienne tiens ton trois turbo un une uv13a5h vu y* |
| SYM - symboles | *000emplois 13e 15k 20d 400e 4eme 4G 4hpjllibl 4i 4x 5k4p94i 5tn 7sS 8d 9yjr A4 a5i Acquires AFP aSdér ass AV b1 BNA Buys C C6E CAC CEAPC dEtat DIRECTMEDICA DOI Err:520 iB4aH iubsd3 June l5 Level LEVETO M3391193 M6 miamglou m² nlaar4 ns4s o4o oHwsv4V4ty P PC PER Predissime qetA r56 rD Renan Rpgmaker rrscR4 RS2K SEVESTRE SIPA sr15z sr3 Synthroid t7 TMI tolya92 Traceur1 v VEU érMa*<br>+ suite de caractères spéciaux |
| UH - interjections | *cependant olivier* |
| UKW - inconnu | *t* (*a-t-il*) |
| VB - verbes | aucun mot ne pouvant être un verbe |
| WH - mots en *wh-* | 0 résultat |

## Annexe 2
## Matrice de confusion de l'étiquetage désambiguïsé d'un texte travaillé

| étiquettes réelles | étiquettes proposées par AQL | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | CC | DT | IN | JJ | NN | NNP | PRP | RB | SYM | VB | Total |
| CC | 18 | | | | | | | | | | 18 |
| CD | | | | | 2 | | | | | | 2 |
| DT | | 67 | | | | | 2 | | | | 69 |
| IN | 3 | 8 | 92 | 1 | | | | 3 | | | 107 |
| JJ | | | | 37 | 3 | | | 1 | | 7 | 48 |
| NN | | | | 7 | 157 | | | | | 5 | 169 |
| NNP | | | | | 5 | 9 | | | | | 14 |
| PRP | 3 | | | 1 | | | 11 | | | | 15 |
| RB | | | | 2 | 1 | | | 8 | | | 11 |
| SYM | | | | | | | | | 77 | | 77 |
| VB | | | | | | | | | | 33 | 33 |
| Total | 24 | 75 | 92 | 48 | 168 | 9 | 13 | 12 | 77 | 45 | 563 |

| VP | FP | FN | VN | Précision | Rappel |
|---|---|---|---|---|---|
| 18 | 6 | 0 | 539 | 0,75 | 1,00 |
| 0 | 0 | 2 | 561 | 0,00 | 0,00 |
| 67 | 8 | 2 | 486 | 0,89 | 0,97 |
| 92 | 0 | 15 | 456 | 1,00 | 0,86 |
| 37 | 11 | 11 | 504 | 0,77 | 0,77 |
| 157 | 11 | 12 | 383 | 0,93 | 0,93 |
| 9 | 0 | 5 | 549 | 1,00 | 0,64 |
| 11 | 2 | 4 | 546 | 0,85 | 0,73 |
| 8 | 4 | 3 | 548 | 0,67 | 0,73 |
| 77 | 0 | 0 | 486 | 1,00 | 1,00 |
| 33 | 12 | 0 | 518 | 0,73 | 1,00 |
| 509 | 54 | 54 | 5576 | | |

# Annexe 3
## Matrice de confusion de l'étiquetage désambiguïsé d'un texte non travaillé

| étiquettes réelles \ étiquettes proposées par AQL | CC | DT | IN | JJ | NN | NNP | PRP | RB | SYM | VB | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CC | 18 | | | | | | | | | | 18 |
| CD | | | | | 2 | | | | | | 2 |
| DT | | 67 | | | | | 2 | | | | 69 |
| IN | 3 | 8 | 92 | 1 | | | | 3 | | | 107 |
| JJ | | | | 37 | 3 | | | 1 | | 7 | 48 |
| NN | | | | 7 | 157 | | | | | 5 | 169 |
| NNP | | | | | 5 | 9 | | | | | 14 |
| PRP | 3 | | | 1 | | | 11 | | | | 15 |
| RB | | | | 2 | 1 | | | 8 | | | 11 |
| SYM | | | | | | | | | 77 | | 77 |
| VB | | | | | | | | | | 33 | 33 |
| Total | 24 | 75 | 92 | 48 | 168 | 9 | 13 | 12 | 77 | 45 | 563 |

| VP | FP | FN | VN | Précision | Rappel |
|---|---|---|---|---|---|
| 18 | 6 | 0 | 539 | 0,75 | 1,00 |
| 0 | 0 | 2 | 561 | 0,00 | 0,00 |
| 67 | 8 | 2 | 486 | 0,89 | 0,97 |
| 92 | 0 | 15 | 456 | 1,00 | 0,86 |
| 37 | 11 | 11 | 504 | 0,77 | 0,77 |
| 157 | 11 | 12 | 383 | 0,93 | 0,93 |
| 9 | 0 | 5 | 549 | 1,00 | 0,64 |
| 11 | 2 | 4 | 546 | 0,85 | 0,73 |
| 8 | 4 | 3 | 548 | 0,67 | 0,73 |
| 77 | 0 | 0 | 486 | 1,00 | 1,00 |
| 33 | 12 | 0 | 518 | 0,73 | 1,00 |
| 509 | 54 | 54 | 5576 | | |

## Annexe 4
## Comparatif des résultats de *Demographics* avant et après nos modifications

| | config 1 | | | config 2 | | | config 3 | | | config 4 | | | config 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| male | 71 | 76 | 7% | 202 | 241 | 19% | 259 | 270 | 4% | 188 | 218 | 16% | 254 | 259 | 2% |
| female | 26 | 26 | 0% | 110 | 111 | 1% | 142 | 143 | 1% | 115 | 122 | 6% | 39 | 37 | -5% |
| married | 2 | 3 | 50% | 1 | 1 | 0% | 2 | 4 | 100% | 4 | 3 | -25% | | 2 | |
| parent | 7 | 9 | 29% | 1 | 1 | 0% | 16 | 19 | 19% | 22 | 19 | -14% | 4 | 3 | -25% |
| **total** | **106** | **114** | **8%** | **314** | **354** | **13%** | **419** | **436** | **4%** | **329** | **362** | **10%** | **297** | **301** | **1%** |

| | config 6 | | | config 7 | | | config 8 | | | config 9 | | | config 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| male | 164 | 187 | 14% | 117 | 131 | 12% | 84 | 116 | 38% | 140 | 147 | 5% | 167 | 186 | 11% |
| female | 25 | 27 | 8% | 53 | 54 | 2% | 73 | 78 | 7% | 26 | 28 | 8% | 66 | 69 | 5% |
| married | 2 | 2 | 0% | 4 | 13 | 225% | | 7 | | | | | 1 | 6 | 500% |
| parent | 6 | 13 | 117% | 14 | 17 | 21% | 27 | 27 | 0% | 1 | 1 | 0% | 2 | 7 | 250% |
| **total** | **197** | **229** | **16%** | **188** | **215** | **14%** | **184** | **228** | **24%** | **167** | **176** | **5%** | **236** | **268** | **14%** |

## Annexe 5
## Comparatif des résultats de *Behavior* avant et après nos modifications

| | config 1 | | | config 2 | | | config 3 | | | config 4 | | | config 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| user | 4 | 9 | 125% | 19 | 23 | 21% | 23 | 21 | -9% | 51 | 63 | 24% | 5 | 6 | 20% |
| prosp. user | | | | 1 | 9 | 800% | 3 | 4 | 33% | 16 | 6 | -63% | 1 | 3 | 200% |
| pot. churner | | | | | | | | | | 1 | 3 | 200% | | 1 | |
| recommender | | | | | | | | | | 8 | 6 | -25% | | | |
| detractor | | | | | | | | | | | | | | | |
| **total** | **4** | **9** | **125%** | **20** | **32** | **60%** | **26** | **25** | **-4%** | **76** | **78** | **3%** | **6** | **10** | **67%** |

| | config 6 | | | config 7 | | | config 8 | | | config 9 | | | config 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| user | 10 | 16 | 60% | 3 | 6 | 100% | | 1 | | 11 | 12 | 9% | 50 | 62 | 24% |
| prosp. user | | 2 | | | | | | | | 1 | 2 | 100% | 2 | 7 | 250% |
| pot. churner | | | | 1 | | -100% | | | | 1 | 0 | -100% | | | |
| recommender | | | | | | | | | | | | | | | |
| detractor | | | | | | | | | | | | | | | |
| **total** | **10** | **18** | **80%** | **4** | **6** | **50%** | **0** | **1** | | **13** | **14** | **8%** | **52** | **69** | **33%** |

# Annexe 6
# Comparatif des résultats de *Sentiment* avant et après nos modifications

| | config 1 | | | config 2 | | | config 3 | | | config 4 | | | config 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| positive | 9 737 | 17 700 | 82% | 2 138 | 3 243 | 52% | 10 986 | 16 061 | 46% | 7 168 | 11 635 | 62% | 2 236 | 7 508 | 236% |
| negative | 4 341 | 7 452 | 72% | 1 366 | 2 130 | 56% | 4 321 | 7 308 | 69% | 3 101 | 5 028 | 62% | 1 081 | 3 360 | 211% |
| **total** | **14 078** | **25 152** | **79%** | **3 504** | **5 373** | **53%** | **15 307** | **23 369** | **53%** | **10 269** | **16 663** | **62%** | **3 317** | **10 868** | **228%** |

| | config 6 | | | config 7 | | | config 8 | | | config 9 | | | config 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| version | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff | av | mtnt | diff |
| positive | 2 176 | 2 872 | 32% | 7 798 | 12 853 | 65% | 6 788 | 11 406 | 68% | 4 885 | 7 508 | 54% | 2 850 | 2 872 | 12% |
| negative | 1 330 | 1 481 | 11% | 3 620 | 8 791 | 143% | 3 167 | 8 510 | 169% | 1 863 | 3 360 | 80% | 1 473 | 1 481 | 11% |
| **total** | **3 506** | **4353** | **24%** | **11 418** | **21 654** | **90%** | **9 955** | **19 916** | **100%** | **6 748** | **10 868** | **61%** | **4 323** | **4 353** | **12%** |

# Table des matières

**MOTS-CLÉS** : analyse de sentiments, fouille de texte, fouille de données, AQL, IBM

## RÉSUMÉ

Ce rapport de stage détaille notre expérience à IBM Allemagne, où nous avons pendant six mois travaillé sur le logiciel *IBM Watson Analytics for Social Media*. Celui-ci permet à son utilisateur de sonder indirectement l'opinion des internautes *via* leurs messages écrits publiquement sur certains sites Internet. À partir d'une simple requête, le système collecte les données correspondantes, les analyse, en extrait non seulement les sentiments vis-à-vis de l'objet de la recherche, mais également de nombreuses informations potentiellement intéressantes à l'utilisateur. L'objet de notre stage portait sur l'amélioration du code pour le français. Nous travaillions en AQL, langage de requêtes créé par IBM, en utilisant des dictionnaires et des expressions régulières.

**KEYWORDS** : sentiment analysis, text mining, data mining, AQL, IBM

## ABSTRACT

This internship report details our experience at IBM Germany, where we worked for six months on the software *IBM Watson Analytics for Social Media*. It allows its user to indirectly survey Internet users' opinion through their public written posts on certain websites. Thanks to a simple query, the system gathers corresponding data, analyses it, extracts not only the feelings expressed towards the searched entity, but also many pieces of informations that can potentially interest the user. The objective of our internship was to improve the code for French language. We worked in AQL, query language created by IBM, while using dictionaries and regular expressions.