Stratégies d'enrichissement sémantique d'analyse d'opinions : contributions logicielles et linguistiques

Nadezhda Rumiantceva

Stratégies d'enrichissement sémantique d'analyse d'opinions : contributions logicielles et linguistiques.

Rumiantceva
Nadezhda

Sous la direction de Ponton Claude

Réalisé au sein de la société : Eloquant
Sous la direction de Kalitvianski Ruslan et Padró Muntsa

UFR LLASIC
Département I3L

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

Année universitaire 2019-2020

Stratégies d'enrichissement sémantique d'analyse d'opinions : contributions logicielles et linguistiques.

Rumiantceva
Nadezhda

Sous la direction de Ponton Claude

Réalisé au sein de la société : Eloquant
Sous la direction de Kalitvianski Ruslan et Padró Muntsa

UFR LLASIC
Département I3L

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

Année universitaire 2019-2020

Remerciements

Je voudrais tout d'abord remercier tous ceux qui m'ont accueillie si chaleureusement au sein de l'entreprise Eloquant : Muntsa, Mathieu, Ruslan, David, Myriam et Emmanuelle. Leurs conseils, leur soutien, l'envie d'aider et de tout expliquer dans les moindres détails, mais surtout la bonne humeur tous les jours, tout cela a rendu ces six mois de stage inoubliables ! L'équipe Sémantique est très soudée mais tellement ouverte à accueillir de nouvelles personnes, ils m'ont mise à l'aise dès les premiers jours ! Je garderai toujours une très belle image de cette entreprise.

J'aimerais également remercier monsieur Kraif et monsieur Ponton pour leur soutien, leur aide, leur gentillesse et leur foi en moi tout au long du Master. J'ai toujours pu compter sur eux pour avoir des conseils.

Une autre personne à qui je veux dire un énorme merci est l'une des personnes les plus importantes de ma vie ! C'est ma maman Liudmila qui m'a donné cette incroyable opportunité de venir faire mes études et ma vie en France. Elle a toujours fait de son mieux pour que je puisse atteindre de beaux résultats. Merci beaucoup, maman !

Finalement, je remercie de tout mon cœur mon amie Capucine et mon conjoint Ludovic. J'ai toujours pu confier absolument tout à ces deux magnifiques personnes, partager avec elles mes pires angoisses et les meilleurs moments de ma vie ! Elles sont à mes côtés quoiqu'il arrive et c'est grâce à elles que je deviens de plus en plus forte tous les jours.

Absolument toutes les personnes évoquées sur cette page ont joué un rôle très important à différents moments de mon parcours universitaire, et je tiens à eux énormément. Elles ont participé à mon développement professionnel et personnel, et je les remercie du fond de mon cœur.

Déclaration anti-plagiat

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.

RUMIANTCEVA
Nadezhda
03/09/2020

[redacted]

Sommaire

Introduction

1 Préambule

Dans le cadre de la deuxième année du Master Sciences du langage : Industries de la langue à l'Université Grenoble Alpes, nous avons effectué un stage de six mois au sein de l'entreprise Eloquant, et plus précisément dans l'équipe R&D [1] Sémantique. Le contrat de 35h/semaine a été établi du 02 mars 2020 jusqu'à 28 août 2020. Le travail s'est déroulé en partie en présentiel, en partie en télétravail, étant donné la situation sanitaire dans le pays.

L'encadrement académique a été assuré par l'enseignant-chercheur au laboratoire LIDILEM [2] Claude Ponton, et, dans l'entreprise, par l'ingénieur-chercheur Ruslan Kalitvianski et la directrice de l'équipe R&D Muntsa Padro.

Dans ce mémoire, nous utiliserons le *nous de modestie* sauf la partie **Remerciements** et la section **Bilan personnel**, étant, bien évidemment, plus personnelles.

Pour des raisons de confidentialité de l'entreprise Eloquant, les noms des catégories utilisées, les chiffres exactes des mesures, ainsi que les détails des algorithmes utilisés seront anonymisés. Ils sont accessibles dans la version complète du mémoire.

2 Présentation de la société

2.1 Présentation générale

2.1.1 Spécialistes de la relation client

L'entreprise Eloquant est un spécialiste de la relation client depuis l'année 2001. Ils ont deux sites : le principal à Gières (38610) et le second à Levallois (92300). Le site de Gières gère le développement des produits de l'entreprise, dont les logiciels proposés par l'équipe Sémantique.

Les clients d'Eloquant sont des entreprises qui souhaitent analyser les interactions avec leurs propres clients. Parmi ces entreprises, nous trouvons des assurances, des opérateurs téléphoniques, des banques, etc.

Eloquant propose actuellement des produits dans 123 pays, dont 320 clients actifs en Europe, dans 12 langues.

L'entreprise est certifiée ISO 27001 et assure la sécurité des données de ses clients.

2.1.2 Les différentes offres de l'entreprise

Jusqu'à l'année 2020, Eloquant proposait trois produits : Dialogue, Écoute et Sémantique. À partir de cette année, les deux premiers produits ont été fusionnés sur la même plateforme qui englobe les solutions Centre de Contact et Mesure Satisfaction Client. Ces deux offres sont complémentaires et flexibles. [3]

1. Recherche et développement.
2. https://lidilem.univ-grenoble-alpes.fr/ (consulté le 03/09/2020).
3. Les descriptions des produits sont prises du site officiel d'Eloquant : https://www.eloquant.com/ (consulté le 03/09/2020)

Centre de Contact Ce produit permet de gérer tous les canaux de l'interaction sur une même interface : les conversations téléphoniques, emails, chats, SMS, applications mobiles, réseaux sociaux, etc. ELOQUANT manipule donc différents types de données langagières, qui peuvent toutes se prêter à l'analyse sémantique.

Mesure Satisfaction Client Cette solution permet de créer des enquêtes de satisfaction et de les envoyer via un canal adapté : email, SMS, QR-code, téléphone, Web, etc. Le recueil d'information et l'analyse sémantique des opinions permettent d'améliorer le service client.

L'équipe CX [4] assure un accompagnement des entreprises tout au long du projet, et l'équipe R&D SÉMANTIQUE (une parmi plusieurs équipes de la R&D) développe des analyses génériques ou des analyses sur mesure pour certains clients de l'entreprise.

2.2 La SÉMANTIQUE

2.2.1 L'équipe

L'équipe SÉMANTIQUE est composée de quatre membres : le responsable IA-Sémantique Mathieu RUHLMANN, l'ingénieur-chercheur Ruslan KALITVIANSKI et les ingénieurs-linguistes David GRACEFFA et Myriam EL HELOU. L'équipe est en étroite collaboration avec l'équipe CX, via le médiateur principal Emmanuelle DUSSERRE, Consultante CX-Sémantique. Enfin, l'équipe SÉMANTIQUE fait partie du grand pôle R&D dont la directrice est Muntsa PADRO.

2.2.2 Les analyses de la SÉMANTIQUE

L'équipe SÉMANTIQUE propose une analyse de *verbatims* [5] sur plusieurs aspects :

— l'extraction des concepts : il s'agit de noms ou groupes nominaux significatifs extraits selon le modèle proposé par Sclano & Velardi (2007) ;
— l'extraction des entités nommées pertinentes pour le client ;
— la catégorisation multi-labels (ou, dans certains cas, multi-classes) des *verbatims*, selon une catégorisation définie avec le client ;
— l'extraction et l'analyse des *feedbacks* [6].

De plus, pour certains clients, la SÉMANTIQUE propose d'autres analyses spécifiques. Par exemple, l'équipe a commencé le développement de l'analyse des transcriptions automatiques de conversations téléphoniques (Dusserre et al., 2020) qui inclut les traitements supplémentaires suivants :

— l'identification des rôles des locuteurs : « conseiller » ou « client » ;
— L'anonymisation des noms, adresses email, lieux ou numéros.

4. *Customer Experience.*
5. Texte rédigé par un client des clients de l'entreprise ELOQUANT. Il peut s'agir d'une réponse à une question ouverte d'enquête de satisfaction, mais aussi d'un mail, d'un commentaire laissé sur un site, etc.
6. Il s'agit d'une forme d'analyse d'opinions, dont nous verrons les détails et une définition plus précise dans le Chapitre **2**.

2.2.3 La restitution des résultats de l'analyse sémantique au client

La restitution se fait via une plateforme Web qui affiche, de manière graphique :

— les métadonnées qui accompagnent généralement les données analysées (date du *verbatim*, lieu concerné, etc.) ;

— les résultats de l'analyse sémantique via une quinzaine de visualisations interactives sous forme d'histogrammes, de « camemberts » ou de nuages de mots.

L'illustration donnant un exemple de quelques analyses de *verbatims* sur la plateforme de restitution se trouve en annexe **A**.

L'intérêt de cette plateforme est non seulement de rendre l'analyse appréhensible en un coup d'oeil, mais également de permettre d'appliquer des filtres. Par exemple, d'afficher en quelques clics tous les *commentaires négatifs* (*feedbacks*) concernant l'*accueil* (catégorisation) dans une *boutique parisienne* (métadonnée) sur *la période janvier-mars* (métadonnée).

3 Différents sujets de travail

Cette section présente brièvement les différentes tâches et leurs évolutions, qui nous ont été confiés au cours du stage. Il s'agit de l'analyse des conversations téléphoniques, du ré-étiquetage des *feedbacks*, de la détection de l'intensité et encore quelque travaux transverses.

3.1 Analyse des conversations : sujet initial

Le sujet initial de notre stage était l'analyse des conversations téléphoniques issues des services clients. Comme dit précédemment, ELOQUANT s'intéresse à l'analyse sémantique de transcriptions automatiques des audios issues de centres d'appels. Ce type de données présente de nombreuses particularités par rapport aux *verbatims* textuels, dont nous parlerons en détail dans la section **1.4** de l'état de l'art. Il s'agissait de l'analyse des transcriptions automatiques d'appels téléphoniques produites par un éditeur extérieur. Il était prévu de développer des briques d'analyse génériques à partir de données audio et de leurs transcriptions provenant d'une dizaine de nos clients. Plus précisément, les tâches principales concernaient :

— la mise en place d'une catégorisation automatique des tours de paroles, puis des conversations complètes ;

— l'extraction des opinions et des alertes ;

— l'extraction des concepts.

Plus en détails, il s'agissait de :

— mener une étude des corpus, afin de définir les catégories et différents éléments à détecter ;

— annoter les corpus, en indiquant les rôle des interlocuteurs et les autres éléments d'analyse linguistique ;

— adapter l'outil de catégorisation utilisé par l'équipe SÉMANTIQUE à l'analyse des conversations téléphoniques ;

— évaluer le système, en comparant les sorties avec le corpus de référence annoté manuellement.

Compte tenu de la crise sanitaire en France s'accentuant dès le début de notre stage en mars 2020, et impliquant un confinement généralisé, l'entreprise n'a pas pu obtenir les autorisations de ses clients, en particulier les assurances, d'utiliser leurs conversations téléphoniques. Après deux mois d'attente des enregistrements, durant lesquels nous avons travaillé sur d'autres tâches décrites dans la partie **4**, il a été décidé de ré-orienter le sujet du stage vers une autre problématique, afin que le temps restant permette de réaliser un travail de qualité dans un autre domaine d'intérêt pour la Sémantique. Ayant participé au travail d'analyse des erreurs du système de détection des *feedbacks*, et ayant identifié les problèmes du système et les facteurs qui pourraient améliorer les sorties, nous avons proposé de ré-orienter le sujet de stage vers le ré-étiquetage des *feedbacks*. Cette proposition a été acceptée par les responsables de l'équipe, ainsi que par nos encadrants industriel et universitaire.

3.2 Travail sur les *feedbacks*

3.2.1 Esquisse de la notion de *feedback*

La notion de *feedbacks clients* est le résultat d'une reconceptualisation du système d'analyse des opinions utilisé par Eloquant avant 2019, qui se reposait, d'une part, sur le travail théorique de Melnikova (2017), et d'autre part, sur un moteur d'analyse purement expert développé par la *startup* Holmes Semantic Solutions [7]. Ce système d'analyse extrayait du texte des fragments, appelés *snippets*, porteurs d'une opinion positive ou négative ou d'une alerte [8].

L'aspect purement expert du système étant perçu comme problématique, il a été décidé d'en développer un système successeur qui, d'une part, reposerait sur de l'apprentissage automatique et, d'autre part, utiliserait également des règles symboliques. Les experts-linguistes en ont profité pour proposer une conceptualisation plus riche et plus harmonique des retours clients. Les détails de cette conceptualisations sont donnés dans la section **2.1** de l'analyse de l'existant.

Suite à l'annotation manuelle d'un corpus de plusieurs milliers de *verbatims* avec cette nouvelle modélisation, un premier système minimal, fondé purement sur un apprentissage automatique (`CRF`), a été développé. Ce système s'est révélé prédisposé à certains types d'erreurs, et notre travail a consisté à trouver des façons de corriger ses sorties et à les enrichir.

3.2.2 Ré-étiquetage des *feedbacks*

Il s'agissait de répertorier les erreurs produites par le `CRF` et les corriger avec une autre approche. Plusieurs erreurs sont possibles, mais les plus gênantes sont celles où le *snippet* extrait porte une mauvaise étiquette. Le but a été de reprendre les *snippets* extraits par le `CRF` et de les ré-étiqueter. Cela part de deux observations :

— l'implémentation du `CRF` utilisée par la Sémantique n'est pas capable de tenir compte d'annotations sémantiques habituellement utilisées par les outils de la Sémantique, notamment

7. Le *startup* est devenue l'équipe Sémantique suite à son acquisition par Eloquant en 2016.
8. Classe d'opinions négatives, du type *Colère* requérant l'attention immédiate du client de la Sémantique.

de lexiques *gazetteers*[9], que l'on suppose utiles pour la tâche ;

— bien que l'étiquetage attribué aux *snippets* par le `CRF` laisse parfois à désirer, les *snippets* eux-mêmes sont souvent syntaxiquement corrects.

Le nouveau système devait donc proposer un nouvel étiquetage aux *snippets* extraits par le `CRF`, en intégrant les différentes ressources sémantiques que nous estimons utiles. Pour cela, il fallait établir une méthodologie d'évaluation efficace du système, de comparer les résultats avec ceux du système à améliorer et de s'assurer que le nouveau projet corrige plus d'erreurs qu'il n'en introduit.

3.2.3 Identification de l'intensité des *feedbacks*

La modélisation théorique des *feedbacks* inclut un attribut d'intensité binaire (*faible* ou *forte*), mais la détection de l'intensité n'était pas faite par le `CRF`. De plus, les corpus d'entraînement n'ont pas été annotés de façon systématique ou cohérente avec l'intensité. Il s'agissait donc de :

— faire une étude théorique de la notion d'intensité dans le contexte des *feedbacks* ;

— ré-annoter le corpus, en s'appuyant sur cette étude ;

— développer un système d'identification de l'intensité des *feedbacks* ;

— intégrer ce système dans le nouveau projet de détection des *feedbacks*.

Il s'agit d'un aspect important pour les *feedbacks* car certains d'entre eux deviennent des alertes lorsque leur intensité est évaluée comme *forte*.

3.3 Sujets hors du cahier des charges

Bien que, chronologiquement, la plupart des tâches décrites dans cette section aient été réalisées durant l'attente des audios pour le sujet initialement prévu, nous avons choisi de les décrire dans la dernière section du mémoire, puisqu'elles ont été assez disparates et de moindre envergure que le travail sur les *feedbacks*.

En plus d'une analyse des conversations téléphoniques d'un client, nous avons participé à l'enrichissement sémantique du projet pour une Maison de Luxe, à l'étude de faisabilité d'intégration des noms de logiciels mentionnés dans un référentiel et à la post-édition des *verbatims* traduits du français vers le russe.

4 Organisation du mémoire

Ce mémoire comporte quatre parties principales. Dans la première partie, nous exposerons l' état de l'art portant sur les différentes méthodes de classification utilisées actuellement, sur la détection de l'intensité et sur l'analyse des conversations téléphoniques. Nous allons également analyser les outils utilisés par l'équipe Sémantique.

La partie **2** portera sur le sujet principal de notre stage : le ré-étiquetage des *feedbacks*.

Dans la partie **3**, nous parlerons de la détection de l'intensité.

9. Un *gazetteer* est une liste de termes (pouvant être des expressions régulières) annotés par des étiquettes sémantiques.

Finalement, la quatrième partie portera sur les travaux transverses effectués tout au long du stage.

Nous finirons le mémoire par un bilan du stage, ses perspectives et un bilan personnel.

Partie 1

-

État de l'art et analyse de l'existant

Chapitre 1. État de l'art

1.1 Introduction

Étant donné la nature professionnelle de notre stage, nous ne pouvons pas rédiger un état de l'art au sens propre du terme pour chaque projet effectué au sein de l'entreprise. En effet, les deux travaux nécessitant une recherche scientifique étaient la détection de l'intensité et l'analyse des conversations téléphoniques. Pour ces deux sujets, il fallait créer des projets à partir de zéro et, par conséquent, faire une étude des ouvrages proposés par d'autres chercheurs. Quant à la tâche de ré-étiquetage des *feedbacks*, nous avons eu des consignes précises car les choix principaux avaient déjà été faits. Afin de faciliter la lecture et la compréhension de notre travail d'enrichissement automatique d'analyse d'opinions, nous avons fait un bref panorama des méthodes de classification automatique utilisées actuellement en TAL.

1.2 Méthodes de classification automatique utilisées actuellement

À présent, il existe trois grands types d'approches applicables à l'analyse automatique de corpus oraux ou écrits, utilisés en TAL. Il s'agit des approches symbolique, statistique et hybride.

1.2.1 Approche symbolique

Cette méthode se base particulièrement sur des règles définies par des spécialistes en langage naturel, généralement des linguistes, et exprimées soit dans un ou plusieurs formalismes dédiés (ex. *TokensRegex* (Chang & Manning, 2014)), soit codées directement dans un langage de programmation généraliste. Il s'agit de l'approche la plus ancienne du point de vue historique du TAL. Des exemples de règles expertes peuvent être trouvées dans la section **2.2.2.2** de l'analyse de l'existant.

Il peut s'agir d'un travail très laborieux car les experts étudient souvent le corpus cas par cas. En partant de l'élaboration des règles les plus génériques, ils sont souvent amenés à définir des règles très spécifiques afin de couvrir au maximum la variation langagière, ou *vice versa*. Il s'agit alors d'un long processus de lecture, de relecture, d'analyse et de réflexion. Cette méthode convient surtout aux données présentant relativement peu de variations : un petit corpus ou un domaine très particulier. Ces conditions permettent de réduire la variabilité au maximum et, ainsi, de faciliter le traitement.

Bien évidemment, lorsque les experts élaborent les règles, en étudiant minutieusement le corpus, ils aboutissent à un système ayant une très bonne précision, tout en essayant de maximiser le rappel. D'après Ramond (2016), en vertu du caractère déclaratif et lisible des règles, les résultats issus d'un système basé sur la méthode symbolique sont relativement facilement interprétables par un humain, le système est donc, en toute logique, plus facile à reprendre et améliorer par d'autres spécialistes. Pourtant, passée une certaine quantité de règles, il devient difficile de comprendre comment fonctionne le système, ce qui le rend plus difficile à modifier. Il devient également de plus en

plus compliqué d'apporter des modifications au système sans le dégrader par ailleurs. Le plus grand inconvénient de cette approche est que ce système sera difficilement applicable à d'autres types de corpus et de données, puisque ses performances seront vraisemblablement fortement dégradées en raison de la dissimilitude des données.

Pour résumer, les systèmes à base de règles ont pour avantage une facilité d'interprétation et d'enrichissement, et pour inconvénients un coût de développement généralement élevé, des limitations liées à l'expressivité du ou des formalismes utilisés, et une *tractabilité* décroissante avec l'augmentation du nombre de règles.

Les expressions régulières et les grammaires hors-contexte sont parmi les formalismes de règles les plus utilisés. Bien que pour certaines tâches l'on adopte de plus en plus des approches fondées sur l'apprentissage automatique, il existe toujours en TAL des systèmes basés uniquement sur des règles, par exemple le système de traduction automatique Apertium (Forcada et al., 2011).

1.2.2 Approche statistique

Quant à la méthode statistique, il s'agit de « la représentation du texte analysé selon des caractéristiques numériques déterminées » (Ramond, 2016, p.10). En effet, il s'agit d'un apprentissage automatique supervisé ou non, lors duquel l'ordinateur crée des statistiques à partir des traits prédéfinis par le développeur, en repérant des co-occurrences ou autres régularités statistiques, lui permettant de représenter le corpus et de pouvoir, par la suite, faire des prédictions avec de nouvelles données. Lorsqu'il s'agit de l'apprentissage supervisé, le système doit apprendre un modèle qui permet d'*inférer* les informations à partir de données annotées par l'humain. Des exemples d'algorithmes d'apprentissage supervisés très connus sont les `SVM`s [1] (Cortes & Vapnik, 1995), les `CRF`s [2] (Lafferty et al., 2001), les algorithmes `MaxEnt` [3] (Nigam et al., 1999) ou encore les réseaux de neurones (Haylin, 2009). Contrairement à ce type d'apprentissage, lors de la modélisation non-supervisée, le système apprend un modèle qui *découvre* les informations à partir des données non-annotées. Des exemples connus de telles approches sont le `clustering` [4] (Jain, 2010) et `word2vec` (Mikolov et al., 2013).

Lors de l'application de la méthode statistique, le travail principal consiste, tout d'abord, à collecter et à définir les données (travailler sur les *verbatims* de plus de 60 caractères, par exemple), à les annoter (souvent, en les normalisant au préalable). Puis, il faut choisir l'algorithme qui convient le mieux au vu des données existantes, ainsi qu'à définir et ajuster les paramètres. Une fois le modèle entraîné, il faut tester ses performances et, peut-être, réajuster les paramètres pour réentraîner le nouveau modèle qui serait plus adapté. Le `clustering` nécessite tout de même un pré-traitement. En effet, les algorithmes d'apprentissage requièrent souvent un pré-traitement des données : par exemple, une tokénisation et/ou une lemmatisation, l'éventuelle suppression des mots considérés comme *vides*, une normalisation orthographique, etc.

Les avantages des approches statistiques sont une plus grande automaticité dans la construction

1. *Support vector machine.*
2. *Conditional random fields.*
3. *Maximum Entropy.*
4. La classification automatique non supervisée.

et l'enrichissement subséquent d'un système de classification par rapport à l'approche experte, et une capacité d'inférer des relations statistiques non-triviales [5] entre les éléments des textes. Quant aux contraintes de l'approche en question, les systèmes qui l'utilisent, s'ils aspirent à être généralistes, ont généralement besoin d'énormes corpus pour apprendre un modèle. Ces corpus doivent de plus être annotés pour les méthodes supervisées. La constitution de tels corpus peut s'avérer difficile lorsqu'un domaine est très particulier, ou lorsqu'il y a des contraintes sur l'usage des données, qui peuvent tomber sous les contraintes du RGPD [6]. De plus, pour de nombreux algorithmes (hormis l'exception notable des arbres de décision), l'interprétabilité du modèle et de ses sorties est très limitée, et les modèles peuvent être très sensibles au bruit (comme nous verrons plus bas dans la section **2.3.1**). Finalement, même si l'apprentissage automatique non supervisé donne de bons résultats dans le `clustering`, il est possible de les améliorer, en combinant l'approche statistique avec des méthodes symboliques.

1.2.3 Approche hybride

L'on constate en pratique que la combinaison des approches citées ci-dessus donne les meilleurs résultats, et beaucoup de systèmes actuels se basent sur une approche hybride, c'est-à-dire celle qui combine l'expertise linguistique avec l'apprentissage automatique. C'est pourquoi, les connaissances linguistiques sont souvent indispensables en TAL.

Pour la classification thématique par exemple, l'approche d'Eloquant pour un nouveau jeu de données consiste en un `topic modelling` non supervisé, qui fait émerger des ensembles de mots correspondant à une hypothétique thématique. Cette étape est suivie d'une lecture humaine, qui abstrait de ces nuages de mots des catégories, en rejette d'autres, assigne les catégories aux *verbatims*, entraîne un classifieur sur ces *verbatims*, puis développe un enrichissement sémantique qui améliore les résultats. Bien évidemment, il existe plusieurs façons d'apprendre et d'enrichir un modèle.

Pour le développement d'un système générique de catégorisation RC [7], Ramond (2016) annote manuellement ses corpus *Train* et *Test*, tandis que le corpus *Devel* reste brut et lui sert à créer la base de règles. Ces annotations sémantiques comportent une classification à deux niveaux : des catégories-*mères*, ainsi que les catégories-*filles*. Elle crée également des *TokensRegex* [8], afin de traiter des cas particuliers, des expressions propres au domaine de la relation client. Cette méthode lui permet de créer un système qui donne de bons résultats dans la classification des *verbatims*. Il faut mentionner que cette tâche est plus compliquée qu'une simple classification de textes écrits car les *verbatims* ont une forte variation linguistique, puisqu'il s'agit du contenu qui est à la frontière des langages écrit et oral. Finalement, la méthode utilisée par Ramond (2016) est coûteuse car l'élaboration d'un nombre de règles assez important demande du temps et beaucoup de travail de linguiste.

5. Des exclusions mutuelles ou des implications, par exemple.
6. Réglementation générale de protection des données.
7. Relation client.
8. Nous parlerons de cette approche dans la section **2.2.2.2**.

1.3 Analyse de l'intensité

1.3.1 Introduction

Malgré les travaux faits au sein de l'entreprise sur la détection d'opinions et de l'intensité, au début de notre stage, l'attribution de ce dernier trait sémantique n'était pas encore appliquée dans le système hybride du projet `feedback-detection`. Afin de choisir la meilleure façon d'intégrer cette analyse des *feedbacks*, il convient de définir ce que c'est l'intensité, ainsi que de préciser sa graduation. Quels sont donc les indicateurs linguistiques permettant de caractériser l'intensité d'un *feedback* ?

Dans cette analyse, nous allons d'abord présenter différentes définitions du phénomène linguistique en question, puis diverses graduations proposées par plusieurs auteurs. Finalement, nous synthétiserons leurs propos dans la conclusion.

1.3.2 Partie 1 : Qu'est-ce que l'intensité ?

L'intensité sert à amplifier et mettre en valeur nos émotions ou à attirer l'attention de notre interlocuteur sur ce que l'on juge important. La notion en question est assez subjective, alors différents locuteurs peuvent la percevoir différemment. Cela complique la tâche d'attribution de l'intensité pour un mot ou une expression donnée.

Pour Kleiber (2013), l'intensité peut avoir une « détermination quantitative de propriétés » ou être une propriété elle-même. Ainsi, l'auteur oppose deux notions : *quantité* et *qualité*. Les exemples ci-dessous permettent de mieux comprendre cette distinction :

quantité : *beaucoup d*'attention pour le client
qualité : La conseillère a été *d'une grande douceur*

Dans le premier extrait de *verbatim*, l'intensifieur *beaucoup* exprime une quantité d'attention accordée au client sans pour autant modifier qualitativement l'entité sur laquelle il porte. Quant au deuxième extrait, l'attribut du sujet *d'une grande douceur* qualifie la conseillère. De plus, l'auteur précise que la quantité peut porter sur les objets et les événements mais aussi sur les propriétés et les états. Ces deux différents types de quantité ne s'expriment pas de la même façon :

évènement : il m'a aidé *beaucoup*
état : j'apprécie *beaucoup*

Gaatone (2007) oppose l'intensité et la quantité, en indiquant que ces deux notions sont « les deux faces » de ce qu'il appelle le *degré*. Pour l'auteur, « l'intensité caractériserait alors les états et les propriétés, et la quantité, les événements et les objets »(p. 93). En réalité, il s'agit de la même opposition que celle de Kleiber (2013) mais Gaatone (2007) ne réunit pas les deux caractéristiques sous le même nom de l'*intensité*. En ce qui concerne la classification des marqueurs de *degré* dans l'article, elle porte essentiellement sur les adverbes. En effet, Gaatone (2007) délimite trois sous-classes des marqueurs :

1. Les adverbes désignant exclusivement la quantité :

— *(bon) nombre, quantité, force.*

2. Les adverbes intensifs :
   — *aussi, si, très.*

Toujours *aussi* satisfait de cette agence
c'est dommage d'etre *si* mal considéré

3. Les adverbes qui expriment et la quantité et l'intensité.

Il est important de noter que la dernière sous-classe est considérée comme la plus grande et regroupe tous les autres adverbes que l'on pourrait percevoir comme *intenses*, y compris les adverbes en *-ment*.

Le site bug *énormément*

Dans son mémoire, Melnikova (2017) étudie l'expression des opinions ainsi que leur intensité. L'auteur distingue l'intensité provenant de l'affect [9] et celle du modifieur. De plus, en regardant ses exemples [10] dans le tableau ci-dessous, nous constatons que les modifieurs transmettent leur intensité à l'affect.

Intensite d'origine de l'affect	**Intensite d'origine de modifieur**	**Intensite obtenue de l'affect**
1 (content)	1 (*peu*)	1 (*peu content*)
1 (deçu)	2 (*très*)	2 (*très déçu*)
2 (colere)	1 (*un peu*)	1 (*un peu en colère*)
2 (choque)	2 (*extrêmement*)	2 (*extrêmement choqué*)

Tableau 7 : L'annotation de l'intensité

FIGURE 1.1 – Tableau d'annotation de l'intensité issu du mémoire de Melnikova (2017)

Si Kleiber (2013) et Gaatone (2007) ne mentionnent que les adverbes ou les locutions verbales [11], Melnikova (2017), illustre par ses exemples que l'intensité peut également être transmise par les adjectifs et les noms. Cette double transmission est décrite dans l'article de Zhang et Ferrari (2014). Les auteurs affirment que l'intensité est graduée de manière *intrinsèque* ou *extrinsèque*. Dans le premier cas, il s'agit du même noyau sémantique pour plusieurs lexèmes [12], tandis que le deuxième cas illustre les modifieurs [13].

intrinsèque : *Bon* site *vs Excellent* site
extrinsèque : Bon site *vs Très* bon site

9. Affect : « toutes sortes de ressentis humains : les sentiments, les émotions, les états dâme, les attitudes émotionnels » (Melnikova, 2017).
10. L'étiquette 1 correspond à une faible intensité, tandis que l'étiquette 2 - une forte.
11. cf. Kleiber (2013).
12. Le cas des noms et des adjectifs.
13. Le cas des adverbes.

1.3.3 Partie 2 : Graduation de l'intensité selon les auteurs

Comme indiqué ci-dessus, l'intensité peut être graduée. Pourtant, il existe différentes analyses des niveaux d'intensité, en allant d'un traitement binaire [14] jusqu'aux analyses de plus en plus fines.

Dans la partie de l'analyse d'intensité de son mémoire, Melnikova (2017) s'adresse aux travaux de Plutchik (1982). Le psychologue représente les émotions de base, tout en indiquant trois niveaux d'intensité pour chaque émotion : nul, faible et fort. La roue de Plutchik (1982) est une représentation d'une graduation *intrinsèque* selon Zang et Ferrari (2014) à trois degrés. Voici un exemple de cette graduation :

sérénité (nul) ⟶ joie (faible) ⟶ extase (fort)

Pourtant, pour son système, Melnikova (2017) choisit une graduation binaire, ne gardant que l'intensité faible ou forte. En effet, elle a décidé de réunir les intensités nulle et faible de Plutchik (1982) sous la même étiquette.

ennuyant 1 énérvant 2

Quant au Zhang et Ferrari (2014), les auteurs proposent un système à cinq niveaux de graduation de l'intensité : faible, modéré, standard, fort et extrême.

un peu bon ⟶ moyennement bon ⟶ bon ⟶ très bon ⟶ extrêmement bon/excellent

1.3.4 Conclusion

En conclusion, l'intensité peut porter soit les objets et les évènements, soit sur les propriétés et les états. En fonction de ce que les locuteurs intensifient, elle ne sera pas exprimée de la même façon.

L'intensité est transmise principalement par les adverbes mais peut également être exprimée dans les adjectifs et les noms. Cela définit la manière dont l'intensité est graduée : de façon intrinsèque ou extrinsèque.

Finalement, la graduation de l'intensité peut varier en fonction des besoins. Il est possible de faire un système binaire ou effectuer une analyse plus fine.

14. Intense ou non intense.

1.4 Analyse des conversations téléphoniques

1.4.1 Introduction

Aujourd'hui, la classification automatique de textes rédigés[15] est plutôt bien maîtrisée, ce qui n'est pas encore le cas des conversations téléphoniques. Ces dernières sont particulièrement difficiles à analyser car le langage oral est beaucoup moins structuré et normalisé par rapport au langage écrit. L'oral est spontané tandis que l'écrit est asynchrone, cela signifie que lors du processus de l'écriture, la personne peut reformuler la phrase à tout moment, choisir les meilleurs mots pour transmettre sa pensée. La variation inter, voire intra-locutrice se manifeste alors plus souvent à l'oral, ce qui impacte inévitablement l'analyse automatique. Quelle est donc la solution la plus adaptée pour la classification automatique des transcriptions de conversations téléphoniques ?

Dans cette étude, nous présenterons tout d'abord l'application des approches de classification automatique citées dans la section **1.2** à l'analyse du langage oral. Par la suite, nous parlerons des problèmes que l'on peut rencontrer lors du traitement de l'oral et nous présenterons quelque solutions. Nous finirons par synthétiser le contenu de cette section (**1.4**) dans la conclusion.

1.4.2 Approches de classification automatique utilisées pour l'analyse du langage oral

Lavalley (2012) préfère l'approche statistique car, selon lui, les méthodes symboliques « se prêtent moins facilement à une utilisation sur différents domaines ou sur des langues plus ou moins fortement dégradées » (p.16). Ces dernières se réfèrent au langage oral qui est difficilement prévisible. L'auteur affirme également que la méthode n'utilisant pas de structures linguistiques s'adapte plus facilement aux nouveaux corpus, y compris aux corpus recueillis dans d'autres langues. Dans son travail, Lavalley (2012) extrait automatiquement des collocations qui peuvent être non seulement des unigrammes, des bigrammes ou des trigrammes. Ces éléments extraits ont autant de n-grammes qu'il faut pour être définis comme pertinents pour la classification et servent à enrichir et améliorer le système.

Deux autres méthodes purement statistiques sont proposées par Bost et El-Bèze (2015) : `word2vec` et `densité thématique`. La première approche consiste en calcul des vecteurs des mots pour détecter les termes similaires et, ainsi, faire la classification. En ce qui concerne la `densité thématique`, il s'agit d'une attribution d'un thème à une conversation téléphonique « s'il est de densité dominante en une position quelconque du dialogue et si la somme de ses densités dans les positions où il est dominant excède un seuil fixé empiriquement » (p. 5). Pour leurs systèmes, les auteurs utilisent des modèles `HMM`[16]. En présentant les résultats de leur travail, ils démontrent que ces deux approches sont plus performantes qu'une approche `SVM`.

Lailler et al. (2015) proposent une méthode hybride d'enrichissement du modèle différente de celle de Ramond (2016) décrite dans la section **1.2.3**. En ce qui concerne l'annotation, seuls les thèmes sont mentionnés dans les transcriptions des conversations téléphoniques. De plus, il n'y a

15. L'attribution automatique d'une ou plusieurs classes à un texte.
16. *Hidden Markov model*. Les modèles utilisés ont 230000 distributions.

qu'un seul thème majoritaire attribué à chaque conversation. Le corpus *Train* sert à produire une liste de n-grammes pertinents pour chaque thème qui seront, après une analyse de la liste par des experts-linguistes, associés à un sous-thème. En tenant compte de cette liste, le système montre de meilleurs résultats. Cette méthode aurait pu être moins coûteuse que celle de Ramond (2016) car, tout d'abord, l'ontologie ne comporte que deux niveaux : thèmes et sous-thèmes. De plus, le travail de linguiste se réduit à une simple correction de la liste des n-grammes produite par le système, puis l'association des n-grammes les plus pertinents à des sous-thèmes. Pourtant, la transcription manuelle de tout le corpus d'entraînement demande énormément de temps et de travail, mais, en même temps, permet de travailler avec des données de meilleure qualité.

1.4.3 Problèmes actuels et solutions existantes

Vu la grande variation linguistique, les corpus oraux posent beaucoup de problèmes au traitement automatique. Il s'agit de la langue spontanée qui varie beaucoup d'un locuteur à l'autre. La parole du même locuteur peut également avoir une grande variation linguistique, ce qui rend très difficile l'élaboration des règles génériques. Lors d'un dialogue, les interventions des locuteurs sont imprévisibles, plusieurs thèmes peuvent être abordés, ce qui complique aussi la classification automatique.
Aujourd'hui, il existe quelque solutions qui permettent de réduire les difficultés de traitement de la parole spontanée. Par exemple, Lavalley (2012) ne compte qu'une occurrence par segment si le mot est répété plusieurs fois. La disfluence verbale peut également être diminuée grâce à la suppression des éléments non lexicaux comme *euh*, *hein*, *hum*, etc. Il ne faut pas oublier de traiter les mots-outils, en faisant une comparaison du corpus spécifique au domaine étudié avec un corpus générique. Tout ceci permet d'accéder directement au lexique qui servira à l'extraction des termes spécifiques et à la classification automatique par thème. Finalement, il est important de privilégier l'apprentissage machine semi-supervisé pour le traitement de petits corpus et lorsque les corpus recueillis contiennent des transcription automatiques dont le WER [17] est assez élevé.

1.4.4 Conclusion

Tout compte fait, le traitement du langage oral nécessite au minimum l'apprentissage machine, et un pré-traitement du corpus est indispensable. L'enrichissement linguistique peut améliorer les résultats, donc les système hybrides s'avèrent être plus performants.

17. *Word Error Rate.*

Chapitre 2. Analyse de l'existant

2.1 La notion de *feedback*

Dans cette section, nous allons définir ce que sont les *feedbacks*, ainsi que les motivations de l'entreprise du choix de leur utilisation. Nous allons également présenter le premier système d'ELOQUANT élaboré pour la détection des opinions et de l'intensité basé uniquement sur des règles symboliques. Finalement, nous décrirons le fonctionnement du module d'extraction des *feedacks* utilisé au début de notre stage, ainsi que ses limites.

2.1.1 Le projet `OPINION-DETECTION`

Ici, nous allons parler du premier système d'analyse d'opinions et de l'intensité utilisé par ELOQUANT. Nous décrirons d'abord les composantes du projet (au sens informatique) qui nous seront utiles par la suite. Puis, nous démontrerons les limites de ce système et présenterons la solution de l'équipe leur permettant de traiter une plus grande variété de données.

2.1.1.1 Système d'avant : purement expert

En ce qui concerne la première version du projet d'analyse des opinions, le système était purement expert, c'est-à-dire basé uniquement sur les règles symboliques. Cette première version a été élaborée par Melikova (2017) et appelée `opinion-detection`. Le système permet d'extraire des fragments exprimant des opinions grâce aux règles linguistiques écrites à l'aide des *TokensRegex* et des *gazetteers*. Le projet permet également d'attribuer un degré d'intensité au *snippet* : faible ou fort.

Quant à sa structure (présentée dans l'annexe **B**), le projet `opinion-detection` contient deux dossiers principaux :`src/main/java` qui sert à faire tourner le programme, et `src/main/resources` qui contient toutes les ressources linguistiques utilisées par le système. C'est le deuxième *package* qui nous intéressera pour notre projet d'amélioration du système hybride utilisé actuellement.

En effet, presque toutes les ressources sémantiques utilisées pour la détection d'opinions et de l'intensité se trouvent dans le dossier `emotion-detection`. Il s'agit des *gazetteers*, des *TokensRegex* et des *semrules*[1] qui permettent de classer les affects définis par Melnikova (2017) dans son mémoire, ainsi que les fichiers décrivant des règles permettant d'extraire l'intensité. Quant au dossier *gazetteers*, il y a un fichier par catégorie : noms, verbes, adjectifs, adverbes. Ce qui est d'un intérêt particulier pour notre projet, ce sont les fichiers traitant l'intensité.

2.1.1.2 Les limites du système purement expert

Comme expliqué dans la section **1.2** qui porte sur les différentes méthodes de classification utilisées à l'époque actuelle, l'approche purement symbolique a ses limites. Il est difficile de développer

1. Règles sémantiques.

un ensemble de règles ayant à la fois une bonne précision et une bonne couverture, du moins pour notre tâche. En tout cas, cette affirmation est vraie pour le cas des données ayant une grande variabilité inter- et intra-locutrice. Étant donné que l'ambition de l'équipe SÉMANTIQUE est d'analyser une grande quantité de *verbatims* variés, l'approche purement symbolique ne permet pas d'avoir des résultats performants sur tous les corpus des clients. Pour cela, il aurait fallu créer un très grand nombre de règles linguistiques, afin de couvrir le maximum des cas. D'un côté, il s'agirait d'un travail très coûteux et, d'un autre côté, il est vraiment difficile d'élaborer une telle quantité de règles sans qu'elles soient contradictoires.

Après avoir fabriqué un premier système fondé sur de l'apprentissage automatique pur (`CRF`), et en avoir constaté les limitations, les membres de l'équipe SÉMANTIQUE ont décidé de faire un système hybride qui ferait *cohabiter* l'approche symbolique et le *machine learning* au sein du même projet. Nous expliquerons le fonctionnement de ce système et ses composantes dans la section **2.2**.

2.1.2 Les catégories des *feedbacks*

Les catégories des affects proposées dans le projet `opinion-detection` ne permettaient pas de répondre aux besoins des clients d'ELOQUANT. Par conséquent, l'équipe a défini de nouveaux types d'entités, appelés *feedbacks*. Comme leur nom indique, il s'agit des retours des utilisateurs, plus particulièrement des fragments porteurs d'opinions ou de desiderata, qui méritent l'attention des clients de l'entreprise.

La plateforme Brat [2] a servi à annoter les corpus des clients. Les *feedbacks* intenses sont marqués par un #. Notons au passage que certains *feedbacks* sont discontinus, d'autres se chevauchent. Cela était permis et encouragé par le guide d'annotation fourni par l'entreprise, mais actuellement les technologies utilisées par la SÉMANTIQUE ne permettent pas d'exploiter ces annotations de façon satisfaisante.

Il existe sept catégories de *feedbacks*, et chaque type de *feedback* peut avoir une ou deux polarités.

Feedback1 Ces *feedbacks* englobent la plupart des affects traités dans le projet `opinion-detection` par Melnikova (2017). Ce sont des opinions ou des jugements des utilisateurs. Elles peuvent être positives ou négatives.

Très bon contact, bravo. ⟶ Feedback1-POS

Toujours trop cher !!! ⟶ Feedback1-NEG

Feedback2 Il s'agit des messages où les utilisateurs font part de leur intention de faire une bonne ou une mauvaise publicité de l'entreprise dont ils avaient eu des services, la recommander ou non.

2. Plateforme d'annotation *open source* : https://brat.nlplab.org/ (consulté le 03/09/2020).

Naturellement, cette catégorie peut également avoir deux polarités : positive ou négative.

Je conseille vos produits à tous mes amis. ⟶ Feedback2-POS

Je ne recommanderais naturellement aps vos services. ⟶ Feedback2-NEG

Feedback3 Cette catégorie englobe les phrases où les utilisateurs disent qu'ils doivent/ont dû faire un effort, une démarche pour obtenir ce qu'ils souhaitaient. La polarité par défaut de cette catégorie est négative.

Accessibilité aux service client téléphonique difficile. ⟶ Feedback3-NEG

Feedback4 Lorsque les utilisateurs souhaitent partir d'une entreprise pour une quelconque raison, leurs demandes sont classées dans la catégorie Feedback4 dont la polarité est toujours négative.

je pense du coup me désabonner malgré que je suis un viel abboné cordialement bst. ⟶ Feedback4-NEG

Feedback5 Cette catégorie apparaît quand les utilisateurs disent qu'ils vont faire un procès ou s'adresser à un organisme tel que l'association des consommateurs, par exemple. La polarité de cette catégorie est, bien évidemment, négative.

Bref cela va aller au tribunal j'ai porter plainte contre défaut de conseil et d'information et mon avocat prépare le courrier. ⟶ Feedback5-NEG

Les deux *feedbacks* suivants sont de nature différente, puisqu'ils n'expriment pas une opinion mais incitent, avec plus ou moins de force illocutoire, à réaliser une action.

Feedback6 Ce *feedback* couvre les cas où les utilisateurs souhaitent que l'entreprise fasse une action pour eux, comme envoyer un contrat ou les rappeler. La polarité est neutre par défaut.

Merci de me confirmer la reconduction de la GARANTIE A VIE sur ma commande de ce jour validée en ligne ⟶ Feedback6-NEU

Feedback7 Cette étiquette est attribuée aux fragments des textes qui décrivent une amélioration ou un changement qu'ils souhaitent dans l'entreprise, ses services ou produits. La polarité est également neutre pour l'étiquette.

La doc du nouveau boîtier aurait été bien. ⟶ Feedback7-NEU

2.1.3 Les corpus annotés

Les corpus annotés utilisés pour l'entraînement du `CRF` du projet `feedback-detection` proviennent de sept clients d'Eloquant. Il s'agit des retours des clients des entreprises sur des thématiques différentes : appareils pour la voiture, énergie (gaz, électricité), livraison de colis, assurance ou encore des avis sur le passage dans une boutique de télécom. Le grand corpus recueilli compte 9986 *verbatims* de qualité différente : des commentaires bien rédigés, parfois en langage soutenu, mais aussi des commentaires courts, écrits en langage SMS ou un langage qui est à la frontière de l'oral.

La tâche de la première annotation des corpus a été confiée à deux étudiants du Master 1 Sciences du langage : Industries de la langue de l'Université Grenoble Alpes. Le travail personnel de 25h a été effectué par chacun des stagiaires. À la fin de cette première étape de l'annotation, 7710 *feedbacks* ont été étiquetés manuellement sur la plateforme Brat.

Puisque la totalité des corpus n'a pas été annotée, sept membres des équipes Sémantique et R&D ont effectué 2h d'annotation en plus. La méthodologie de cette étape a été différente de la première : cette fois-ci, les membres de l'équipe Sémantique ont d'abord fait l'étape de pré-annotation automatique. Les sorties ont été complétées et corrigées par les sept annotateurs. Le fruit de ce travail sont 1416 *feedbacks* détectés en plus.

Finalement, le grand corpus qui a servi à l'entraînement du `CRF` comptait au total 9126 *feedbacks*, dont 2369 considérés comme intenses. La répartition des étiquettes est suivante :

Étiquette	Nombre d'occurrences
Feedback1-POS	6074
Feedback1-NEG	2037
Feedback2-POS	202
Feedback2-NEG	204
Feedback3-NEG	200
Feedback4-NEG	123
Feedback5-NEG	58
Feedback6-NEU	118
Feedback7-NEU	110

Table 2.1 – Répartition des *feedbacks* collectés et utilisés pour l'entraînement du modèle de `CRF`.

Nous voyons que le corpus d'entraînement est déséquilibré, ce qui provoquerait les erreurs du *machine learning* qu'il faudrait corriger par une autre approche d'étiquetage. De plus, l'équipe Sémantique a estimé que environ 25% des *feedbacks* collectés sont discontinus. Pourtant, le `CRF` implémenté dans le projet `feedback-detection` ne sait pas gérer la discontinuité des *snippets*.

2.2 Le projet FEEDBACK-DETECTION : composantes et principes de fonctionnement

2.2.1 Le CRF

Le modèle a été entraîné avec les annotations décrites dans la section **2.1.3**. Le CRF classifie chaque *token* en fonction de son contexte. Pour l'entraînement du CRF, on utilise un fichier *traning* au format BIO [3].

```
1  Prix         N       B-Feedback1-POS
2  attirants    ADJ     I-Feedback1-POS
3  .            PUNC    O
4  Moins        ADV     O
5  Service      N       B-Feedback1-NEG
6  de           P       I-Feedback1-NEG
7  médiocre     ADJ     I-Feedback1-NEG
8  qualité      NC      I-Feedback1-NEG
9  et           CC      O
10 sav          NC      B-Feedback1-NEG
11 délocalisée  VPP     I-Feedback1-NEG
12 .            PUNC    O
```

FIGURE 2.1 – Exemple d'un fichier au format BIO

Chaque *token* a une annotation morpho-syntaxique proposée par Stanford dans la deuxième colonne. Dans la troisième colonne : B [4] désigne début du *snippet*, I [5] - intérieur du *snippet* et O [6] - extérieur du *snippet*. Le CRF apprend les frontières des *snippets* : la séquence du B et des I ayant la même étiquette de *feedback*, jusqu'au premier O $\longrightarrow$ extraction des *snippets* + récupération de l'étiquette.

Le CRF entraîné avec le schéma BIO ne permet pas d'avoir une représentation simple des *feedbacks* discontinus. Pour celles-ci seul le dernier fragment, souvent le plus porteur de sens, a été inclus dans le jeu de données d'entraînement.

2.2.2 Ressources sémantiques

Cette section décrit les ressources développées par l'équipe, qui sont habituellement utilisées dans d'autres tâches de la SÉMANTIQUE, et qui, pour des raisons techniques, ne peuvent pas être exploitées par l'implémentation CRF utilisée actuellement. Le projet `feedback-detection` est donc contraint à les utiliser de diverses façons complémentaires au CRF, et la combinaison de ces approches est appelée approche *hybride*.

3. Une description du format BIO et de ses variantes peut être trouvée ici (en anglais) : https://en.wikipedia.org/wiki/Inside-outside-beginning_(tagging) (consulté le 03/09/2020).
4. *Begin*.
5. *Inside*.
6. *Outside*.

L'équipe s'est rapidement aperçue des limitations de l'approche purement *machine learning*, et a soit réutilisé des règles existantes (celles pour les anciennes *alertes*, par exemple), soit écrit de nouvelles, notamment pour les entités qui n'étaient jamais recherchées avant, comme Feedback6 ou Feedback7. Les règles reprises l'ont souvent été telles quelles, mais dans le cas du Feedback2, qui était indifférencié du point de vue de la polarité, il a fallu les restructurer et en écrire quelques nouvelles.

2.2.2.1 *Gazetteers*

Les *gazetteers* sont des listes de lexiques annotés. Le fichier de *gazetteers* contient deux colonnes : la première est une liste de lexies, souvent exprimées en expressions régulières, et la deuxième - l'étiquette sémantique correspondante.

Ces fichiers servent à faire des annotations sémantiques qui peuvent modifier les scores du *machine learning*.

```
1  compassion  POS
2  incirrect   NEG
3  att?rape-nigauts?   NEG
4  promotion   POS
5  forcing    NEG
```

FIGURE 2.2 – Exemple d'un fichier des *gazetteers* de polarité.

2.2.2.2 Regles d'annotation : *TokensRegex* et *Label-Transform*

L'annotation des simples *tokens* isolés n'est pas suffisante pour l'analyse des données langagières car bien souvent il faut étiqueter toute une expression particulière. C'est pourquoi, l'équipe SÉMANTIQUE utilise les *TokensRegex*. Ce formalisme permet d'annoter avec une étiquette sémantique voulue une suite de *tokens* décrite via une expression régulière pouvant porter non seulement sur les formes des *tokens*, mais également sur leurs annotations syntaxiques. Chang & Manning (2014) affirment : « with TOKENSREGEX, we can easily combine the robustness of statistical methods with the control of a rule based system » [7]. Ce formalisme étant utilisé par ailleurs dans les systèmes d'analyse de la SÉMANTIQUE, il a facilement été intégré dans le système hybride utilisé actuellement.

Dans les projets de l'équipe, il existe quatre façons d'utiliser les expressions régulières, dont les trois premières emploient les *TokensRegex* : les règles d'annotation sémantique (SA [8]), les règles de BOOST, les règles de classification et les règles de réannotation des chaînes. Précisons au préalable que les *TokensRegex* opèrent sur des séquences de *tokens* préalablement annotés avec des informations morpho-syntaxiques (voire sémantiques), et ces règles peuvent faire référence aux dites annotations, par exemple consulter l'étiquette morpho-syntaxique ou le lemme d'un *token*.

7. « Avec les *TokensRegex*, nous pouvons facilement combiner la robustesse des méthodes statistiques avec le contrôle d'un système basé sur les règles » (traduction personnelle).

8. *Semantic Annotation*.

SA Les règles d'annotation sémantique servent à créer une couche d'annotation supplémentaire sur les *tokens*. Cette annotation, qui porte sur l'ensemble des *tokens* captés par l'expression de la règle, peut par la suite diriger l'algorithme de classification vers la catégorie voulue. Dans l'exemple ci-dessous, la règle crée le vecteur entre le *pattern* donné et l'annotation FEEDBACK4. À force de rencontrer ce vecteur dans le contexte de la catégorie Feedback4, l'algorithme du *machine learning* augmentera les scores de cette catégorie pour les expressions qui matchent avec ce *pattern*. L'action demandée est exprimée par `sa`.

```
#test : perdu un client
{ pattern: ( [{lemma:/perdre/}] [{lemma:/.+/}]{0,3} [{lemma:/client/}] ),
  action: ( Annotate($0,sa,"FEEDBACK4") ) }
```

FIGURE 2.3 – Exemple d'une règle d'annotation sémantique.

BOOST Les règles de ce type servent à repondérer les scores attribués par l'apprentissage automatique, typiquement en leur ajoutant 1, d'où leur nom de règles de *boost*. Dans la catégorisation par `label-ranking`, cela sert à éviter que les catégories ayant des scores trop bas soient éliminées puisqu'ils n'atteignent pas un certain seuil minimum. Cela permet également, pour les catégories déjà bien proéminentes, de les mettre encore plus en valeur, lorsque l'on cherche à ne retenir que la catégorie la plus saillante. Par conséquent, le *snippet* qui correspond au *pattern* aura le score le plus élevé, et la catégorie choisie sera celle que l'on veut. Dans le cas suivant, le *pattern* recherché est toute expression ayant une annotation sémantique ACCUEIL/ATTITUDE.

```
#Accueil
{ pattern: (([{sa:/ACCUEIL|ATTITUDE/}])),
  result: ( HolmesGroup($1,"command","action:BOOST;target_cat:2.ACCUEIL_ATTITUDE") ),
  name: "acc2" }
```

FIGURE 2.4 – Exemple d'une règle de BOOST.

Règles de classification Ce type de règles est utilisé dans le système hybride du projet `feedback-detection`. En effet, elles interviennent après l'étiquetage des textes fait par le `CRF`. Leur but est de remplacer l'étiquette donnée par le *machine learning* par une nouvelle étiquette.

Règles de ré-annotation de chaînes Ce dernier type de règles se distingue des trois précédents, puisqu'il s'agit de simples expressions régulières opérant sur des chaînes, auxquelles sont associées des commandes de ré-annotation ou de suppression d'annotations existantes. Par exemple, il est totalement inutile d'analyser les formules de politesse telles que *Bonne soirée* ou *Cordialement*. Voici une règle qui permet de l'indiquer :

```
#test: J ai resilie le 3 juin dernier
#test:  je suis resilier depuis le 17 mai de mon abonnement
{ pattern: ( [{lemma:/avoir|être/} | {word:suis}] [{lemma:/r[ée]sili(er|ee?|ée?)/}]  ),
  result: ( HolmesGroup($0,"FEEDBACK","FEEDBACK4") )
}
```

FIGURE 2.5 – Exemple d'une règle de classification.

```
rex:.*[Bb]onn?e (journ[eé]e?|soir[ée]e?|continuation|souscription|chance|(fin de journ[ée]e?)).*=-
```

FIGURE 2.6 – Exemple d'une règle d'extraction de *snippets*.

Nous appelons ce dernier type de règles des commandes de *label-transform*, dont l'application est généralement la dernière ou l'avant-dernière étape de toute la chaîne de traitements.

2.2.3 L'intégration des règles symboliques dans le projet FEEDBACK-DETECTION

Dans cette section, nous allons décrire la structure du système hybride du projet `feedback-detection` et expliciter comment les règles symboliques y sont intégrées.

2.2.3.1 Structure générale du projet

Comme tous les projets Java de l'équipe SÉMANTIQUE, le projet `feedback-detection` a deux dossiers principaux : `src/main/java` et `src/main/resources` (la structure du projet est présentée dans l'annexe **C**). Il existe également deux dossiers qui contiennent les fichiers permettant de tester le système, comme des tests de régression, par exemple : `src/test/java` et `src/test/resources`. Il s'agit d'une structuration générale des projets Maven [9].

Le dossier `src/main/java` contient l'ensemble de codes propres au système, des outils de développement ou des scripts d'évaluation. Quant au dossier `src/main/resources`, il englobe tous les fichiers de règles symboliques et les fichiers `.properties` qui permettent de configurer la *pipeline* (notamment l'enchaînement de ses différentes étapes) via un fichier de paramétrage XML.

2.2.3.2 L'ordre d'application des règles

En ce qui concerne l'application des règles symboliques dans le système hybride, elles interviennent après le `CRF`. Une stratégie assez simple permet de réconcilier les cas de chevauchement entre les *snippets* extraits par le `CRF` et ceux extraits par les règles : on ne garde que ceux des règles. Quant à leur intégration, le projet `feedback-detection` prend en compte seulement les règles de classification et les règles d'extraction des *snippets*. Les autres ressources sémantiques, même si elles ont été élaborées et sont présentes dans le projet, n'ont jamais pu être intégrées dans l'implémentation du `CRF`, ce qui constitue l'une des motivations du travail de ré-étiquetage.

9. http://maven.apache.org/ (consulté le 03/09/2020).

2.3 Les limites du système hybride

Même si le projet `feedback-detection` combine l'apprentissage automatique avec les règles symboliques préalablement éprouvées, ce qui devrait donner de très bons résultats, le système fait encore beaucoup d'erreurs qui « sautent aux yeux » des clients. Nous pouvons classer les défauts du projet en quatre types : les erreurs d'extraction des *snippets*, les erreurs d'étiquetage, l'impossibilité d'intégrer les annotations sémantiques et l'absence de détection de l'intensité.

2.3.1 Erreurs d'extraction

Tout d'abord, le `CRF` fait beaucoup d'erreurs d'extraction des *snippets*. Dans le fichier de suivi des erreurs du *machine learning*, dont nous parlerons dans la section **2.3.5**, nous rencontrons beaucoup de *snippets* qui sont inutiles pour l'analyse sémantique effectuée dans le cadre du projet de détection des *feedbacks*. Nous pouvons les classer ainsi :

- **Les omissions** Certains *snippets* qui doivent être analysés par le système, ne sont pas extraits par le `CRF`.
- **Le bruit** Il s'agit des *snippets* extraits que le système ne doit pas analyser.
  - *La ponctuation (surtout les points d'exclamation).*

!!!

  - *Les formules de politesse.*

Bonne soirée
Merci !
Bonne fin de journée

  - *Les connecteurs logiques.*

Au contraire
Tout d'abord
En conséquence

  - *Les adresses e-mail, les URL*

fournisseur@ANONYME.com...

  - *Les interjections.*

Aie !

- **L'extraction partielle des *snippets*** Dans ce cas, il manque une partie du *snippets* qui permettrait la bonne analyse.

joignable
Cette facture
service client

Il est important de mentionner que les erreurs d'extraction causent, dans la plupart des cas, les

erreurs d'étiquetage par la suite.

2.3.2 Erreurs d'étiquetage

Lorsque le `CRF` attribue une mauvaise étiquette au *snippet*, nous parlons d'une erreur d'étiquetage. Il s'agit d'une erreur de polarité et/ou d'une erreur de label.

Erreurs de polarité Étant donné que certains *feedbacks* peuvent être soit négatifs, soit positifs, le `CRF` attribue une étiquette avec le bon label mais la polarité est incorrecte. Ainsi, nous retrouvons les cas suivants :

vous ne serez pas déçu ⟶ Feedback1-POS (Feedback1-NEG attendu)
Je recommande rien chez vous ⟶ Feedback2-POS (Feedback2-NEG attendu)

Erreurs de labels Dans ce cas, le `CRF` ne choisit pas le bon label. De plus, la polarité choisie peut également être incorrecte.

faire bien attention à eux ! ⟶ Feedback1-POS (Feedback2-NEG attendu)
aucun contact ⟶ Feedback1-POS (Feedback3-NEG attendu)

2.3.3 Intégration des annotations sémantiques

Comme évoqué dans la section **2.2.3.2**, seules les règles de classification et les règles d'extraction des *snippets* sont utilisables dans le projet `feedback-detection`. Toutes les autres ressources sémantiques, c'est-à-dire les *gazetteers*, les règles d'annotation sémantique, les règles de BOOST, ne sont pas prises en compte car la version du `CRF` utilisée ne le permet pas. Autrement dit, les règles symboliques utilisées dans ce système hybride sont complètement indépendantes du *machine learning*. Elles servent tout d'abord à remplacer les mauvaises étiquettes par les bonnes et à compléter l'extraction des *feedbacks* par des règles pour les entités que le `CRF` n'extrait pas (en raison du peu d'exemples d'entraînement).

L'inconvénient que cela engendre est que les règles de classification ne sont pas incorporées dans le modèle du *machine learning*, alors seuls les *snippets* qui matchent parfaitement avec le *pattern* sont corrigés. Au contraire, l'inclusion d'annotations sémantiques aurait permis au `CRF` de modifier les scores non seulement des *snippets* qui matchent avec la règle mais aussi ceux des *snippets* que l'algorithme *juge* ressemblants. Cette approche aurait permis de couvrir un plus grand nombre de cas et de commettre moins d'erreurs.

De même, l'impossibilité d'intégrer les *gazetteers* nous force à créer des *TokensRegex* qui ont une syntaxe plus complexe et volumineuse :

```
1 fuir    BAD_FEEDBACK2
2 fuiye[rz]   BAD_FEEDBACK2
```

vs.

```
#test: fuyez cette société
{ name: "feedback1_neg_11",
  pattern: ( [{word:/fuyez|FUYEZ|fuiye[rz]/}]),
  action: ( Annotate($0,applied, "applied") ),
  result: ( HolmesGroup($0,"FEEDBACK","BAD_FEEDBACK2") )
}
```

2.3.4 Intensité

Le projet `feedback-detection` n'identifie pas l'intensité des *feedbacks*. Il avait été choisi d'entraîner le `CRF` sans indication d'intensité, sinon cela aurait doublé le nombre de classes à apprendre, et de plus, l'intensité n'avait pas été annotée de façon systématique dans l'ensemble des données. Aucun autre développement n'étant venu combler cette lacune, tous les *snippets* ont la valeur 1 [10] pour l'option `intensity`. L'analyse approfondie de l'intensité fait partie des besoins actuels d'ELOQUANT.

```
<DOCUMENT analyzable="true" id="testErreursCRF_6">
    <METADATA corpus="testErreursCRF" language="fr"/>
    <TEXT>Fuyez !!!</TEXT>
    <ANALYSIS date="2020-07-10T11:42:57.390644300">
        <SENTENCES>
            <SENTENCE id="0" spanFrom="0" spanTo="8" tokenCount="2">Fuyez !!!</SENTENCE>
        </SENTENCES>
        <FEEDBACK-DETECTION>
            <FEEDBACKS>
                <OPINION type="feedback2" polarity="negative" intensity="1" label="Feedback2" spanFrom="0" spanTo="5"
                score="0.9943" algorithm="ALGO">
                    <SNIPPET>Fuyez</SNIPPET>
                    <LEMMATIZED-SNIPPET>fuir </LEMMATIZED-SNIPPET>
                </OPINION>
            </FEEDBACKS>
        </FEEDBACK-DETECTION>
    </ANALYSIS>
</DOCUMENT>
```

FIGURE 2.7 – Analyse du *verbatim* « Fuyez !!! » avec une option `intensity`

2.3.5 Suivi des erreurs

Afin de pouvoir analyser et améliorer les sorties du projet `feedback-detection`, nous avons rempli, à plusieurs, une feuille Excel partagée de suivi des erreurs. Cette feuille contient plusieurs colonnes, voici les éléments principaux :

1. Énoncé problématique : il s'agit du *snippet* portant une étiquette estimée erronée.

10. Intensité faible d'après Melnikova (2017)

2. Phrase correspondante : le *verbatim* d'où vient le *snippet*, autrement dit le *snippet* dans son contexte. Disposer du *verbatim* complet permet de retester le système suite à une amélioration.
3. Label obtenu : l'étiquette attribuée par le système hybride.
4. Label attendu : l'étiquette qui, selon nous, aurait dû être attribuée.
5. Niveau de récurrence : l'erreur apparaît-elle souvent dans les sorties ? Le niveau varie de 1 (rare) à 3 (très souvent).
6. Source de l'erreur : l'erreur provient-elle du *machine learning* ou d'une règle symbolique ?
7. Catégorie de l'erreur : s'agit-il d'un mauvais étiquetage et/ou d'une mauvaise extraction ?
8. Solution proposée : *TokensRegex* ou *gazetteers* ? [11]

Pour construire ce corpus d'erreurs, nous avons parcouru plusieurs corpus des clients différents et thématiquement hétérogènes, en commençant par des opérateurs téléphoniques et en finissant par des assurances. Seules les erreurs par omission sont absentes de cette feuille de suivi, puisqu'en faire une liste exhaustive aurait requis beaucoup de travail manuel et de lecture de corpus.

Finalement, nous avons obtenu un fichier contenant 371 erreurs provenant soit du `CRF`, soit des règles symboliques. Le but principal était de recueillir un corpus varié mais petit car la prochaine étape était la correction de ces sorties à l'aide de *TokensRegex*. Comme indiqué dans la section **1.2.1**, un très grand nombre de règles symboliques pourrait dégrader davantage les résultats du projet `feedback-detection`.

Type de l'erreur	Nombre d'erreurs
CRF	368
Symbolique	3
Mauvaise extraction	177
Polarité	182
Labels	19

TABLE 2.2 – Répartition des erreurs de la feuille de suivi.

Nous supposons que les erreurs venant des règles symboliques peuvent être causées par le conflit entre plusieurs règles, vu qu'il est difficile de contrôler un grand nombre de *TokensRegex* faits pour plusieurs *feedbacks* différents. Par ailleurs, en analysant le fichier des erreurs, nous constatons que, dans le cas des erreurs de label, le `CRF` *préfère* attribuer l'étiquette Feedback1. Ceci peut être expliqué par le déséquilibre du corpus d'apprentissage utilisé pour entraîner le `CRF` : la quasi-totalité des exemples sont des Feedback1.

11. Même si nous ne pouvons pas intégrer les *gazetteers* dans le `CRF`.

Conclusion : étude du besoin

La définition des limites du système hybride utilisé au moment du début de notre travail sur ce sujet nous aide à spécifier les besoins de l'entreprise pour leur projet de détection des *feedbacks*. Tout d'abord, nous devons développer un mécanisme de ré-étiquetage des *feedbacks* qui corrigerait le maximum des erreurs d'extraction et d'étiquetage. Pour cela, un classifieur multi-classes avec le même algorithme d'apprentissage que celui utilisé dans les tâches de classification thématique de la SÉMANTIQUE nous a paru être une approche prometteuse. Dans l'idéal, il devra permettre d'intégrer les différentes annotations sémantiques, à savoir les *TokensRegex* et les *gazetteers* qui créeront des vecteurs entre le *token*/*pattern* et l'étiquette souhaitée, tout en aidant l'algorithme de l'apprentissage automatique. Par la suite, il nous faut inclure l'identification de l'intensité dans l'analyse des *feedbacks*, en élaborant un système avec un corpus d'entraînement le plus adapté. Il est également nécessaire de définir les degrés de l'intensité en fonction des besoins des clients. Finalement, il est indispensable de ne pas négliger les contraintes opérationnelles, à savoir l'empreinte mémoire et le temps d'exécution.

Partie 2

-

Feedbacks : Ré-étiquetage des *feedbacks*

Chapitre 3. Introduction

Dans ce chapitre, nous allons présenter le projet `feedback-classifier` que nous avons conçu et implémenté et qui a servi à améliorer les résultats de l'analyse des *feedbacks* utilisée par l'équipe SÉMANTIQUE. Il s'agit d'un système hybride qui utilise une cascade de classifieurs multi-classes et des règles expertes pour le ré-étiquetage des *feedbacks*. Nous allons, tout d'abord, décrire les composantes principales du projet, puis la méthodologie générale du développement. Nous finirons par évaluer le nouveau système, en le confrontant à plusieurs corpus de référence.

3.1 Les composantes logicielles du projet FEEDBACK-CLASSIFIER

Le projet `feedback-classifier` reprend la structure générale `Maven` des projets de l'équipe SÉMANTIQUE (annexe **D**). À la différence de la description des autres projets évoqués auparavant, dans cette section, nous allons détailler toutes les composantes et expliquer à quoi elles servent au sein du projet. Nous tenons à mentionner que, au tout début, il s'agissait d'un programme de test : il a été créé pour entraîner des modèles et observer leurs performances. C'est pourquoi, le nom officiel du projet est `feedback-classifier-training`, mais nous l'appelons dans le mémoire simplement `feedback-classifier`.

Les dossiers-*parents* sont les mêmes que pour tous les autres projets : `src/main/java` et `src/main/resources`. En plus de cela, nous avons créé le dossier `data` qui contient les ressources pour l'entraînement des modèles.

Le dossier `src/main/java` englobe plusieurs *packages* qui incluent les différents codes Java.

Les éléments dont nous parlerons dans cette partie sont ceux qui ont dans leur nom le mot *relabelling*[1]. Le *package* `classification` contient le script de la *pipeline* principale qui fait toute l'analyse des entrées et produit en sortie des fichiers XML annotés. Dans `devtools`[2], se trouvent les scripts CLI[3] permettant de tester les nouvelles règles symboliques sur des phrases entrées dans la console. Quant au *package* `enricher`, nous trouvons dedans l'*enricher*[4] qui déplace le résultat de l'analyse des *feedback* dans la balise dédiée du fichier XML et non dans la balise de catégorisation[5]. Finalement, `training` contient le code Java qui sert à entraîner les modèles.

Les ressources sémantiques nécessaires pour le ré-étiquetage sont dans le dossier `src/main/resources/relabelling`. Nous y trouvons le fichier de configuration de la *pipeline* principale du projet, ainsi que le dossier `fr` contenant les *gazetteers*, les *TokensRegex*, ainsi que les taxonomies et les fichiers `.xml` et `.properties` qui sont propres à chaque classifieur[6] utilisé dans le

1. Ré-étiquetage des labels (notre traduction).
2. Outils de développement (notre traduction).
3. *Command line interface*, ou interface en ligne de commandes, qui accompagne les projets de la SÉMANTIQUE incluant une pipeline.
4. Fichier servant à enrichir l'analyse principale des *verbatims*.
5. Puisque nous avons choisi d'implémenter la tâche de ré-étiquetage comme une tâche de catégorisation (de *snippets*), dont les résultats sont habituellement stockés dans une balise dédiée dans notre structure de données.
6. Le projet `feedback-classifier` utilise trois classifieurs en plus du `CRF`.

projet.

Quant au dossier `data/relabelling` du projet, il contient les modèles obtenus et les corpus d'entraînement dans `training`.

Nous donnerons plus de détails sur les *packages* et dossiers `intensity` dans la partie **3** du mémoire. Le contenu du *package* `convert` sera abordé dans la section **3.2.2**.

3.2 Méthodologie générale du développement

Cette section est consacrée à la méthodologie de notre travail qui nous a permis de définir le nombre de classifieurs à utiliser, ainsi que de trouver les meilleures configurations pour chacun d'eux. Nous commencerons par la présentation de l'outil d'entraînement des modèles utilisé par l'équipe SÉMANTIQUE, puis nous expliquerons l'étape de la préparation des données. Pour finir, nous parlerons des essais de différentes configurations dont les résultats nous ont poussée à faire le choix d'utiliser un nombre précis de classifieurs dans le projet `feedback-classifier`.

3.2.1 Prise en main de l'outil d'entraînement des modèles

Afin de choisir les meilleures configurations, il nous a fallu apprendre à utiliser le script d'entraînement des modèles qui calcule et sérialise un modèle (au format `.ser` de STANFORD), et génère une matrice de confusion [7] avec plusieurs mesures permettant de décrire les performances du modèle : la précision, le rappel, l'*accuracy* [8] et la F-mesure.

L'exemple du fichier d'entraînement se trouve dans l'annexe **E**. Ce programme prend plusieurs arguments :

1. `"-taxonomie"` : il s'agit du fichier de taxonomie du modèle que l'on veut entraîner. Celui-ci doit être au format `.csv`.

```
1 id  concept concept parent  id parent   description
2 1.FEEDBACK1  1.FEEDBACK1  1. Feedback1
3 2.AUTRE 2.AUTRE 2. Autre
```

FIGURE 3.1 – Taxonomie des Feedback1 issue du projet `feedback-classifier`.

7. Il s'agit d'une façon de visualiser, sous la forme d'un tableau à deux dimensions, les distributions des prédictions d'un classifieur par rapport à une classification de référence.

8. Parfois appelée *exactitude* ou *justesse* en français.

2. `"-trainingCsv"` : le fichier d'entraînement annoté manuellement au format `.csv`.

```
FEEDBACK2    A fuir absolument
FEEDBACK3   il demande 10 fois les meme documents !
FEEDBACK2    a éviter carrément
FEEDBACK6    ils ne reponde pas au mails ! ! !
```

FIGURE 3.2 – Extrait du fichier d'entraînement `OtherLabels`.

3. `"-parametrizedConfig"` : la configuration de la *pipeline* au format `.xml`.
4. `"-modelId"` : le nom que l'on veut attribuer au modèle sérialisé.
5. `"-sizeQuotaTraining"` : la proportion du corpus donné dédiée à l'entraînement du modèle. Il s'agit d'un nombre décimal compris entre 0 et 1. Par exemple, si l'on met `0.8`, alors 80% du corpus serviront pour la phase d'entraînement, et 20% restants seront utilisés pour la phase de test. À partir du corpus de test, la matrice de confusion sera établie automatiquement.
6. `"-reportRelevantTerms"` : le rapport détaillé contenant un comptage de *features* pertinents par classe, leur récurrence exprimée en nombre absolu d'occurrences et en pour-milles par rapport au nombre total de *features*.

3.2.2 Préparation des données

Pour faire les différents tests de macro- et micro-configurations, nous avons préparé cinq types de corpus : `Polarity`, `HybridLabels`, `OtherLabels`, `Feedback1` et `HybridTotal`. L'objectif était de séparer la polarité des labels, de tester la mise à part des feedback1, afin d'équilibrer au maximum les corpus donnés. Finalement, il fallait comparer les résultats avec ceux du modèle entraîné avec tous les labels et la polarité ensemble.

Polarity Étant donné que le système `feedback-detection` comptait beaucoup d'erreurs de polarité sur les *feedbacks* polarisés, nous avons décidé de commencer par la création d'un classifieur binaire qui pourrait résoudre ce problème. Le corpus à notre disposition contenait 7510 *snippets* dont 72% d'énoncés positifs et 28% de négatifs. Cette répartition étant très déséquilibrée, nous avons supposé que cela pourrait biaiser le modèle, qui produirait beaucoup de résultats faux positifs avec l'étiquette POS. C'est pourquoi, nous avons constitué un deuxième corpus, équilibré, c'est-à-dire dont la répartition serait 50/50. Pour le faire, nous avons gardé tous les énoncés négatifs et nous avons récupéré automatiquement le même nombre de *snippets* positifs de façon aléatoire. L'inconvénient de cette approche était que nous perdions beaucoup d'exemples d'énoncés : le corpus ainsi constitué ne contenait que 4256 *snippets*. Nous avons donc préparé un troisième corpus, également équilibré, mais avec d'autres exemples positifs choisis aussi aléatoirement. Après tous les tests, il fallait garder le modèle qui donnerait les meilleurs résultats.

HybridLabels Puisque les corpus ci-dessus ne contiennent que les annotations de polarité, il nous en fallait un avec les labels des *feedbacks*. Dans un deuxième temps, nous avons préparé le corpus de *training* contenant tous les exemples de tous les labels sans la polarité. Ce dernier contenait 7709 *snippets*, et la répartition était suivante :

- Feedback1 - 89.5% ;
- Feedback2 - 4% ;
- Feedback3 - 1.8% ;
- Feedback4 - 1.4% ;
- Feedback6 - 1.3% ;
- Feedback7 - 1.3% ;
- Feedback5 - 0.6%.

OtherLabels Comme le corpus de tous les labels était également très déséquilibré (en faveur des feedback1), nous avons pris la décision d'isoler tous les feedback1 dans un corpus à part. Finalement, le corpus appelé `OtherLabels` [9] était composé de 1014 énoncés. Voici la répartition des étiquettes de ce fichier de *training* :

- Feedback2 - 40% ;
- Feedback3 - 19.6% ;
- Feedback4 - 12.1% ;
- Feedback6 - 11.6% ;
- Feedback7 - 10.8% ;
- Feedback5 - 5.7%.

Feedback1 Le but du modèle entraîné avec ce corpus était de prédire si le *snippet* donné est un Feedback1 ou non. Il attribue donc soit l'étiquette Feedback1, soit Autre. Malheureusement, nous n'avons pas pu obtenir un corpus plus ou moins équilibré car nous aurions perdu énormément d'exemples. Nous avons donc décidé de garder le jeu de données tel quel, avec 90% d'énoncés pour les Feedback1 et 10% - pour les autres. Le corpus final contenait 7723 *snippets*.

HybridTotal Ce fichier de *training* nous a été nécessaire pour pouvoir comparer les résultats de tous les autres modèles avec le modèle entraîné sur le corpus de base qui comprend le jeu d'étiquettes décrit dans la section **2.1.2**. Autrement dit, et les labels, et les polarités sont pris en compte. Le corpus d'entraînement est composé de 11403 *snippets*, et la répartition des étiquettes est suivante :

- Feedback1-POS - 70% ;
- Feedback1-NEG - 20.5% ;
- Feedback2-POS - 2% ;
- Feedback2-NEG - 2% ;
- Feedback3-NEG - 1.8% ;
- Feedback4-NEG - 1.1% ;

9. Autres labels (notre traduction).

— Feedback6-NEU - 1.1% ;
— Feedback7-NEU - 1% ;
— Feedback5-NEG - 0.5%.

3.2.3 Essais de différentes configurations

Cette section est consacrée à la méthodologie de conduite et de suivi des expériences réalisées avec différentes configurations des classifieurs.

3.2.3.1 Mise en place du suivi des expériences

Afin d'avoir un panorama de tous les tests effectués, pouvoir comparer les résultats et choisir la meilleure configuration pour le projet `feedback-classifier`, nous avons mis en place un classeur de suivi au format Excel. Ce document rend les expériences reproductibles et permet de voir quels sont les éléments qui influencent le plus le contenu des fichiers de sortie.

Le classeur contient cinq feuilles :

— **Résultats Classifieurs binaires :** elle réunit les scores des classifieurs `Polarity`, `Feedback1` et `Intensity`.
— **Résultats HybridLabels :** cette feuille contient les scores des classifieurs `HybridLabels` et `OtherLabels`.
— **Résultats HybridTotal :** nous avons décidé de mettre ces résultats à part en tant que référence *baseline*.
— **Tests SA :** ce sont les résultats des tests de différentes configurations des annotations sémantiques.
— **Erreurs :** il s'agit du suivi des erreurs des meilleurs modèles.

Les classifieurs binaires sont séparés des classifieurs multi-catégoriels car les mesures choisies et la façon de les interpréter ne sont pas toujours les mêmes.

Mesures utilisées Pour tous les classifieurs, nous avons décidé de calculer les mesures de base : la précision, le rappel, l'*accuracy* et la F-mesure. En ce qui concerne les classifieurs binaires, nous avons décidé de calculer le MCC [10], le coefficient qui, d'après Delgado et Tibau (2019) et Chicco et Jurman (2020), semble être plus informatif que la F-mesure en cas des corpus déséquilibrés. Ce coefficient, qui est une mesure de corrélation, varie de -1 à 1 et montre si le système prédit les étiquettes parfaitement (1), s'il répond au hasard (0) ou s'il donne les résultats inverses à ceux attendus (-1).

Finalement, nous avons mis de côté cette mesure car nous nous sommes rendu compte que le MCC corrélait presque parfaitement avec la F-mesure, et n'apportait donc aucune nouvelle information sur le classement ordinal des classifieurs.

10. *Matthews correlation coefficient.*

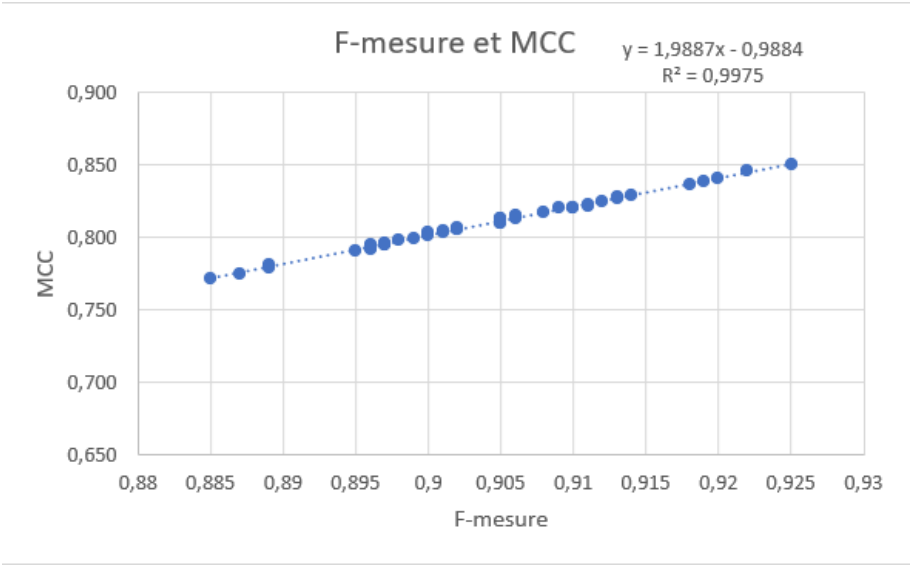


FIGURE 3.3 – La F-mesure *vs.* le MCC. La linéarité de la dépendance est flagrante. La droite de régression linéaire suggère la formule suivante : MCC = 2*F - 1.

Puisque pour toutes les mesures nous avons calculé la moyenne de toutes les classes, nous avons décidé de calculer également la médiane et l'écart-type pour la F-mesure et le MCC, afin d'avoir plus de critères de référence pour choisir le système et la configuration qui produisent les meilleures moyennes et une dispersion de scores minimale.

Validation croisée Étant donné que tous les calculs ont été faits sur seulement 20% du corpus d'entraînement choisis aléatoirement, chaque évaluation était influencée par des hasards de l'échantillonnage aléatoire. C'est pourquoi, nous avons choisi de refaire la même manipulation pour les configurations les plus prommeteuses cinq fois. Nous l'aurions fait pour toutes les configurations, si le processus d'entraînement du classifieur était plus rapide.

Macro- et micro-configurations Nous décrivons dans les sections suivantes ce que l'on appelle les macro- et micro-configurations. Par macro-configuration nous entendons le choix, le nombre et l'enchaînement des classifieurs, et par micro-configuration les paramètres individuels de chaque classifieur.

3.2.3.2 Macro-configurations

Une fois l'ensemble des tests effectué, il nous a fallu faire le choix de la structure du projet `feedback-classifier`. Le tableau ci-dessous présente les meilleurs scores obtenus par chaque modèle. Nous avons décidé de ne donner que la précision, le rappel et la F-mesure car ce sont les mesures principales sur lesquelles nous nous sommes basée pour choisir la meilleure marco-configuration.

L'analyse des mesures nous a démontré que les meilleurs scores étaient obtenus lorsque nous séparions le traitement en plusieurs étapes. Cela s'explique par le fait que la grande majorité des corpus

sont très déséquilibrés, donc nous obtenons un biais vers les catégories qui sont le plus représentées.

En enlevant la classe sur-représentée, nous aidons le système à mieux catégoriser les autres classes : pour le Feedback2, nous gagnons 0.199 de F-mesure, et pour le Feedback3 - 0.404. Bien évidemment, nous ne pouvons pas nous fier aux scores de la classe Feedback5 car elle est représentée par très peu d'exemples, donc les fluctuations des scores sont extrêmes.

Compte tenu des tests effectués, nous avons décidé de partir sur les trois classifieurs : `Feedback1`, `OtherLabels` et `Polarity`. Nous avons également choisi de garder cet ordre car il nous a paru logique pour faire les premiers essais, et finalement, il nous a donné de très bons résultats.

3.2.3.3 Micro-configurations

L'outil de classification utilisé par l'équipe SÉMANTIQUE propose de générer différents *features*[11] que l'on peut utiliser dans l'analyse des textes. Le choix de ces traits se fait dans les fichiers `.xml` (un extrait du fichier que nous utilisons pour le classifieur `Feedback1` se trouve dans l'annexe **F**) et `.properties`. Le dernier fichier dépend du premier et permet de choisir les annotations sémantiques que l'on souhaite appliquer.

Tout au long de l'expérience, pour chaque classifieur, nous avons fait varier ces paramètres :

- `"grammar"` : le fichier `.properties` qui définit les *features* sémantiques qui seront utilisés.
- `"useNPs"` : utiliser ou non les n-grammes de groupes nominaux.
- `"NPsMaxLen"` : la longueur maximale des n-grammes. Si nous mettons `3`, alors le système va utiliser les unigrammes, les bigrammes et les trigrammes.
- `"useRels"` : utiliser les relations syntaxiques ou non.
- `"useSemFeat"` et `"useSemWord"` : utiliser les *gazetteers* et les *TokensRegex* ou non. Ces deux attributs doivent avoir la même valeur : soit `true`, soit `false`.
- `"usePos"` : utiliser les *part of speech*[12] ou non.
- `"useLemma"` : utiliser les lemmes ou non.

Nous pouvons également choisir d'appliquer ou non les fichiers de *StopWords* et de synonymes élaborés par les linguistes de l'équipe SÉMANTIQUE. Si nous ne voulons pas les utiliser, alors nous pointons vers un fichier vide.

Le choix des *features* vraiment utiles pour l'analyse sémantique n'était pas évident car une surabondance de traits pouvait dégrader les résultats. C'est pourquoi, nous avons fait un grand nombre d'essais et avons choisi seulement ceux qui étaient vraiment nécessaires pour l'analyse de chaque classifieur. Toutes les micro-configurations que nous avons testées pour chaque classifieur sont présentées dans l'annexe **G**.

Dans le tableau ci-dessous nous allons présenter les meilleures micro-configurations, en les comparant avec les résultats de l'analyse sans les *features*.

11. Traits.
12. Catégories grammaticales.

Classifier	useNPs	gazetteers	TRs	useRels	useSemFeat	useSemWord	usePOS	useLemma	
Appreciation	-	-	-	-	-	-	-	-	
	3	+	+	+	+	+	+	no Stopwords	
OtherLabels	-	-	-	-	-	-	-	-	
	3	+	+	+	+	+	-	-	
Polarity	-	-	-	-	-	-	-	-	
	3	+	+	+	+	+	+	no Stopwords	

FIGURE 3.4 – Les meilleures micro-configurations choisies pour les tois classifieurs.

Finalement, pour les classifieurs binaires, nous avons choisi d'utiliser les trigrammes et tous les *features* sauf le fichier des *Stop Words*. Quant au classifieur `OtherLabels`, nous n'utilisons ni les *part of speach*, ni les lemmes.

3.3 Annonce du plan du volet

Dans la suite de la partie, nous présenterons le travail principal effectué pendant notre stage : le ré-étiquetage des *feedbacks*. Nous parlerons plus en détails des travaux linguistiques et logiciels qui ont été faits, et nous finirons par évaluer le système obtenu.

Chapitre 4. Travail linguistique

4.1 Ressources sémantiques

Afin de corriger les erreurs les plus importantes et les plus récurrentes de la première version du projet `feedback-classifier`, nous avons inclus les différentes annotations sémantiques : les *gazetteers* et les *TokensRegex*. Dans ce chapitre, nous parlerons de leur intégration, création, et, bien entendu, de leurs impacts sur les performances du système.

4.1.1 *Gazetteers*

Tout d'abord, nous avons récupéré le fichier *gazetteers* déjà existant dans le projet `feedback-detection` mais qui n'a jamais été pris en compte par le `CRF`. Ce fichier, qui s'appelle `polarite_all_pos.txt`, contient une grande liste de *tokens*, exprimés à l'aide des expressions régulières, de toutes les catégories grammaticales.

```
1  ab[îi]mée?s?    NEG
2  abb?[ée]rr?[ae]nt   NEG
3  impolie?s?^^INEG
4  d[ée]sesp[ée]r[ée]ment  NEG
5  efficacement    POS
6  [ée]goïstement  NEG
7  crise   NEG
8  cruaut[ée]  NEG
9  danger  NEG
10 demeuré NEG
```

FIGURE 4.1 – Extrait du fichier `polarite_all_pos.txt`

Nous remarquons que ces *gazetteers* utilisent des expressions régulières pour couvrir les variations morphologiques des mots. Nous avions vu que, dans notre projet, nous pouvions travailler avec des lemmes, à condition d'utiliser un autre type de fichiers *gazetteers*. Tous les *gazetteers* qui ont été créés par nous-même par la suite se trouvent dans ces nouveaux fichiers. En effet, il est plus facile de travailler avec les lemmes car l'écriture énumérant toutes les formes possibles du mot peut-être remplacée par sa forme canonique :

impolie ?s ? $\longrightarrow$ impoli

Nous avons également choisi de séparer toutes les parties du discours pour les *gazetteers* de polarité, afin de simplifier la navigation. Ainsi, nous avons dissocié les adjectifs, les noms, les verbes et les adverbes. Dans le fichier des propriétés, nous avons bien précisé toutes les catégories grammaticales

données par TALISMANE[1] (Urieli, 2013) que nous pouvons rencontrer dans chacun des fichiers.

```
fr_feedback_neg_verbs_lemma.file=relabelling/fr/gazetteers/fr_feedback_neg_verbs_lemma.txt
fr_feedback_neg_verbs_lemma.sourceAnnotation=edu.stanford.nlp.ling.CoreAnnotations$LemmaAnnotation
fr_feedback_neg_verbs_lemma.sourcePOS=^(V|VINF|VPP|VPR|VS|VIMP|NPP)
fr_feedback_neg_verbs_lemma.targetAnnotation=fr.ho2s.holmes.annotations.
                                            SemanticTypeAnnotations$SemanticFeatureAnnotation
fr_feedback_neg_verbs_lemma.ignoreCase=true
```

FIGURE 4.2 – Configuration du fichier `fr_feedback_neg_verbs_lemma.txt` dans le fichier des propriétés.

Finalement, nous avons établi un fichier des *gazetteers* avec des labels, et nous l'avons intégré dans le projet `feedback-classifier`. Ce fichier a été principalement inspiré par les *TokensRegex* du projet `feedback-detection`. Certains *gazetteers* ont été ajoutés pour corriger de nouvelles erreurs.

```
UFC FEEDBACK5
fuir    FEEDBACK2
difficile   FEEDBACK3
difficult[ée]   FEEDBACK3
juge    FEEDBACK5
```

FIGURE 4.3 – Extrait du fichier `feedback_labels.txt`.

Quant à l'ordre d'application de ces fichiers, ils sont employés avant les *TokensRegex* dans cet ordre-là :

```
gazetteer.annotators.ids=feedback_labels,fr_feedback_neg_adjs_lemma,fr_feedback_neg_nouns_lemma,
                         fr_feedback_neg_verbs_lemma,fr_feedback_neg_adverbs_lemma,polarite_all_pos
```

FIGURE 4.4 – L'ordre d'application des *gazetteers* dans les classifieurs `Feedback1` et `OtherLabels`

```
gazetteer.annotators.ids=fr_feedback_neg_adjs_lemma,fr_feedback_neg_nouns_lemma,fr_feedback_neg_verbs_lemma,
                         fr_feedback_neg_adverbs_lemma,polarite_all_pos
```

FIGURE 4.5 – L'ordre d'application des *gazetteers* dans le classifieur `Polarity`

4.1.2 *TokensRegex*

Le travail d'intégration des *TokensRegex* a demandé plus de réflexion car il a fallu tout d'abord copier les règles utilisées dans le projet `feedback-detection`, en modifiant leur type : une règle

1. Outil d'étiquetage morpho-syntaxique utilisé par les classifieurs de la SÉMANTIQUE.

de classification devait devenir une règle d'annotation sémantique. Par la suite, nous avons trouvé que certains *TokensRegex* étaient trop génériques, il était donc nécessaire de les spécifier un peu plus. Enfin, nous nous sommes rendu compte que quelques règles provoquaient des conflits entre les catégories, et certains *snippets* ont été mal classés.

4.1.2.1 Copie de la plupart des règles

Comme dans le projet `feedback-detection` les étiquettes données par le système comprennent et le label, et la polarité, il existe donc deux fichiers séparés pour les Feedback2 : `feedback_good_feedback2.txt` et `feedback_bad_feedback2.txt`. Dans notre projet, nous avons réuni toutes ces règles, en les modifiant légèrement et en changeant leur type, afin de n'obtenir qu'un seul fichier pour la catégorie en question. Ainsi, deux règles séparées dans le projet `feedback-detection` sont devenues une seule dans le projet `feedback-classifier`.

```
#test: je vais vous faire une mauvaise publicité
{ name: "feedback2_neg_1",
  pattern: ( ([{lemma:/mauvais/}] [{lemma:/pub(licité)?/}])  ),
  action: ( Annotate($0,applied, "applied") ),
  result: ( HolmesGroup($1,"FEEDBACK","BAD_FEEDBACK2") )
}
```

FIGURE 4.6 – Exemple d'une règle issu du fichier `feedback_bad_feedback2.txt`.

```
#test: je vais vous faire une bonne publicité
{ name: "feedback2_pos_6",
  pattern: ( ([{lemma:/bon/} &!{applied:applied}] [{lemma:/pub(licité)?/} &!{applied:applied}])  ),
  action: ( Annotate($1,applied, "applied") ),
  result: ( HolmesGroup($0,"FEEDBACK","GOOD_FEEDBACK2") )
}
```

FIGURE 4.7 – Exemple d'une règle issu du fichier `feedback_good_feedback2.txt`.

```
#test : je vais vous faire une mauvaise publicité
#test : je vais vous faire une bonne publicité
{ pattern: ( ([{lemma:/mauvais|bon/}] [{lemma:/pub(licité)?/}]) ),
  action: ( Annotate($0,sa,"FEEDBACK2") ) }
```

FIGURE 4.8 – Exemple d'une règle issu du fichier `feedback_feedback2.txt`.

Pour les autres *feedbacks*, nous avons simplement modifié le type des *TokensRegex*, en laissant les expressions régulières telles quelles étaient.

4.1.2.2 Spécification des règles trop génériques

Certains *TokensRegex* n'étaient pas assez spécifiques, ce qui pouvait causer des erreurs car la couverture de l'expression régulière était trop large. Nous avons donc caractérisé encore plus précisément les structures grammaticales et syntaxiques qui nous intéressaient. Nous allons illustrer ce travail par l'exemple du traitement de l'énoncé *j'ai beau donner toutes les infos*. Voici la règle de catégorisation :

```
#test: j'ai beau donner toutes les infos
{ name: "FEEDBACK3_avoir_beau",
  pattern: ( [{fst==1} && {pos:CLS}] [{lemma:avoir}] /.+/{0,1} [{word:beau} &!{applied:applied}] /.+/{0,5}),
  action: ( Annotate($0,applied,"applied") ),
  result: ( HolmesGroup($0,"FEEDBACK","FEEDBACK3") )
}
```

FIGURE 4.9 – *TokensRegex* pour traiter l'expression *avoir beau* dans le projet `feedback-detection`.

D'après le TLFi [2], l'expression *avoir beau* est toujours suivie d'un verbe à l'infinitif. La présence d'un pronom entre le mot *beau* et le verbe est également possible. Nous avons donc modifié la règle ainsi :

```
#test : j'ai beau donner toutes les infos
{ pattern: ( [{fst==1} && {pos:CLS}] [{lemma:avoir}] /.+/{0,1} [{word:/beau/} &!{applied:applied}] /.+/{0,1}
[{pos:/VINF/}] /.+/{0,3}),
  action: ( Annotate($0,sa,"FEEDBACK3") ) }
```

FIGURE 4.10 – *TokensRegex* pour traiter l'expression `avoir beau + VINF` dans le nouveau projet.

Ce changement nous a permis, d'un côté, de préciser que l'expression en question est toujours suivie d'un infinitif, et, d'un autre côté, de réduire le nombre de *tokens* possibles à la fin du *pattern* de cinq à trois.

4.1.2.3 Suppression de certaines règles

La frontière entre les *feedbacks* Feedback6 et Feedback7 n'est pas toujours nette. Certains membres de l'équipe proposent même de réunir ces deux types de *feedback* en une seule catégorie Directives [3] ou Incitatives.

Dans le projet `feedback-detection`, l'énoncé *Revoyez la sélection de votre personnel* est considéré comme Feedback7, et la phrase *Revoyez-moi la sélection de votre personnel* est classée comme Feedback6. De plus, il y a une règle qui reconvertit certains Feedback7 en Feedback6.

2. Trésor de la langue Française informatisé.
3. Terme issu de la théorie des actes de langage de Searle (1969).

```
#test : Revoyez la sélection de votre personnel
{ name: "feedback7_imperatif_2eme_pers_bis",
  pattern: ( ([{plur==1} && {scd==1} && {sa:/actions_suggerees/} && {firsttok==1}
  &!{applied:clitique_sujet_2eme_pers}] /.+/{0,15}) | ([{scd==1} && {imp==1} && {sa:/actions_suggerees/}]
  /.+/{0,15}) ),
  action: ( Annotate($1,applied,"applied") ),
 result: ( HolmesGroup($0,"FEEDBACK","FEEDBACK7") )
}
```

FIGURE 4.11 – Exemple d'une règle Feedback7 issu du projet `feedback-detection`.

```
#test : Revoyez-moi la sélection de votre personnel
{ name: "feedback6_imperatif",
  pattern: ( ([{plur==1} && {scd==1} && {sa:/actions_suggerees/} && {firsttok==1}] [{lemma:/-/}]
  ([{pos:/CLO|PRO/}] | [{lemma:/vous|moi|me/}]) /.+/{0,15}) | ([{scd==1} && {imp==1} &&
  {sa:/actions_suggerees/}] /.+/{0,15}) ),
  action: ( Annotate($1,applied,"applied") ),
 result: ( HolmesGroup($0,"FEEDBACK","FEEDBACK6") )
}
```

FIGURE 4.12 – Exemple d'une règle Feedback6 issu du projet `feedback-detection`.

```
#Cette règle permet de convertir des cas précis de FEEDBACK7 en FEEDBACK6 : Changez mon contrat. Rappelez moi.
merci de me recontacter etc."
{ name: "feedback7_devient_feedback6",
  pattern: ([{groupString:/.*FEEDBACK7.*/}]* ([{groupString:/.*FEEDBACK7.*/} && {pos:/CLO|PRO/} && {fst==1}]
  | [{groupString:/.*FEEDBACK7.*/} && {lemma:/ma|mon|mes/}]) [{groupString:/.*FEEDBACK7.*/}]*),
  action: ( ClearGroups($0)),
  result: ( HolmesGroup($0,"FEEDBACK7","FEEDBACK6") )
```

FIGURE 4.13 – Règle de conversion de certains Feedback7 en Feedback6 issue du projet `feedback-detection`.

Dans notre projet, nous avons décidé de bien séparer les deux types de *feedback* en question, et considérer tous les impératifs comme un Feedback6, et toutes les expressions au conditionnel comme un Feedback7. Cela nous a permis de supprimer les règles susceptibles de créer des conflits entre ces deux catégories.

```
#test : Revoyez-moi la sélection de votre personnel
{ pattern: ( ([{plur==1} && {scd==1} && {firsttok==1} && ({lemma:/actualiser|r?app?eler|afficher|améliorer|
changer|continuer|arrêter|demander|proposer|donner|revoir|aider|réduire|diminuer|r?ajouter|interdire|insister|
dire|écouter|r?allonger|expliquer|pouvoir|motiver|tenir/}|{word:/ayez|AYEZ|Ayez/})] ([{lemma:/-/}])?
([{pos:/CLO|PRO/}] | [{lemma:/vous|moi|me/}]) /.+/{0,15}) | ([{scd==1} && ({lemma:/actualiser|afficher|
améliorer|changer|continuer|arrêter|demander|proposer|donner|revoir|aider|réduire|diminuer|r?ajouter|interdire|
insister|dire|écouter|r?allonger|expliquer|pouvoir|motiver|veuillez/}|{word:/ayez|AYEZ|Ayez/})] /.+/{0,15}) ),
  action: ( Annotate($0,sa,"FEEDBACK6") ) }
```

FIGURE 4.14 – *TokensRegex* de Feedback6 issu du projet `feedback-classifier`.

```
#test : il faudrait un tableau des prix.
#test : il faudrait changer votre methode
#test : il faudrait nous donner les bonnes infos. il faudrait plus de services
#test : il faudrait plus de services
#test : il faudrait un truc mieux. il faudrait une chose plus fun.
{ pattern: ([{lemma:/il/}] [{word:/faudrai[ts]/} & !{applied:/clitique_sujet_1ere_pers/}]  /.+/{0,15}),
  action: ( Annotate($0,sa,"FEEDBACK7") ) }
```

FIGURE 4.15 – Exemple d'un Feedback7 issu du projet `feedback-classifier`.

4.1.2.4 Création de nouvelles *TokensRegex*

Deux nouveaux fichiers ont été créés : `feedback_polarity.txt` et `feedback_no-feedback1.txt`. Certaines expressions devaient être annotées avec la polarité : les cas de négation, de certains *snippets* compliqués pour l'analyse, comme *vous en avez du boulot à présenter vos excuses à tous les clients.*

```
#test : je ne regrette absolument pas
{ pattern: ( (([{lemma:/ne/}])? [{sa:/NEG/}] /.+/{0,1} [{lemma:/pas|jamais/}] ) ),
  action: ( Annotate($0,sa,"POS") ) }
```

FIGURE 4.16 – Changement de la polarité avec la négation.

En ce qui concerne le deuxième fichier, il a été créé pour corriger certaines erreurs du premier classifieur (`Feedback1`). Nous voulions que les *snippets* ayant une annotation sémantique d'un autre label, passent directement dans la catégorie Autre. C'est pourquoi, nous avons créé cette règle de BOOST :

```
#test : je vais vous faire une bonne publicité
{ pattern: ( [{sa:/FEEDBACK2|FEEDBACK6|FEEDBACK3|FEEDBACK4|FEEDBACK5|FEEDBACK7/}]),
  result: ( HolmesGroup($0,"command","action:BOOST;target_cat:2.AUTRE") ),
  name: "AUTRE_1" }
```

FIGURE 4.17 – Seule règle de BOOST utilisée par le projet `feedback-classifier`.

4.2 Choix des annotations sémantiques pour chaque classifieur

Pour le classifieur `OtherLabels`, la différence des scores est flagrante : nous gagnons 0.057 de F-mesure, lorsque nous utilisons toutes les annotations sémantiques présentes dans le projet. Pour le classifieur `Feedback1`, nous avons gardé la même configuration car l'écart n'est pas aussi important, et les annotations des labels servent pour la règle de BOOST. Pour `Polarity`, nous avons choisi de ne garder que les annotations sémantiques de polarité.

4.3 Impact des annotations sémantiques sur les performances des classifieurs

Afin d'évaluer l'impact des annotations sémantiques sur les performances du système, nous avons comparé les scores des modèles ayant les mêmes micro-configurations hormis l'intégration des ressources linguistiques au sein de chaque classifieur.

En analysant le tableau ci-dessous, nous constatons que les classifieurs binaires ont de base des scores très élevés, et alors les annotations sémantiques n'ont pas d'influence significative. C'est pourquoi, pour le classifieur `Polarity`, nous avons gardé le minimum de ressources complémentaires, à savoir les annotations de polarité. Cependant, même si quantitativement l'effet des ressources sémantiques n'est pas très visible pour les scores des classifieurs binaires (tels que `Polarity` et `Feedback1`), ces ressources permettent tout de même de corriger les erreurs les plus récurrentes qui pourraient *sauter aux yeux* des clients.

Concernant le classifieur `OtherLabels`, nous notons que les annotations sémantiques jouent un rôle important dans la classification multi-classes et font monter la F-mesure de 0.091. Cette différence n'est pas négligeable.

4.4 Conclusion

Nous aurions pu évaluer les résultats du *machine learning* pure, analyser les erreurs et créer des annotations sémantiques qui servent uniquement à corriger les erreurs de notre système. Faute de temps alloué à l'amélioration de la détection des *feedacks*, nous avons choisi d'intégrer les ressources sémantiques déjà existantes indépendemment de leur utilité. Finalement, nous avons vu que l'impact des ressources intégrées, adaptées et créées à nouveau est significatif est très positif pour le projet `feedback-classifier`.

Chapitre 5. Travail logiciel

5.1 Description de l'algorithme *machine learning* choisi

Pour notre système, nous avons utilisé le même algorithme qui est employé par tous les autres projets de catégorisation chez ELOQUANT.

Le classifieur multi-classes reprend le *snippet* extrait par le `CRF` utilisé dans le projet `feedback-detection`, en ignorant l'étiquette de *feedback* donnée par le *machine learning*. Contrairement au `CRF`, ce classifieur re-classifie le *snippet* entier (et non chaque *token*) et ne tient pas compte du contexte plus large des *snippets*.

5.2 Approche par cascade de classifieurs

5.2.1 Description de la cascade

La cascade comporte l'enchaînement de quatre classifieurs : `CRF`, `Feedback1`, `OtherLabels` et `Polarity`. Le `CRF` est utilisé uniquement pour l'extraction des *snippets*, et les autres classifieurs servent au ré-étiquetage des énoncés obtenus.

5.2.2 Raisons du choix

La raison principale du choix de cette approche était la grande disproportion du premier corpus d'apprentissage. Par conséquent, il y a eu un biais dans l'apprentissage avec les catégories les plus représentées (les Feedback1-POS), et un sous-apprentissage des autres classes. Pour aider la reconnaissance des autres labels, nous avons mis les Feedback1 dans un classifieur à part. Nous avons également préféré de traiter la polarité séparément car elle varie seulement pour les Feedback1 et les Feedback2.

5.2.3 Description des classifieurs

Dans cette section, nous décrirons le fonctionnement logiciel des trois classifieurs utilisés pour l'étiquetage des *snippets*.

5.2.3.1 Feedback1

`Feedback1` prend en entrée les *snippets* extraits par le `CRF`. Le classifieur ignore l'étiquette donnée par le `CRF`, et attribue soit l'étiquette Feedback1, soit Autre à chaque énoncé.

5.2.3.2 OtherLabels

Le classifieur prend en entrée seulement les *snippets* étiquetés comme Autre, et les re-classe.

5.2.3.3 Polarity

Le classifieur n'a que deux polarités : positive et négative car il s'applique seulement sur les Feedback1 et les Feedback2.

5.2.3.4 L'enricher

Afin de mettre en place l'enchaînement des classifieurs et placer les sorties dans la balise `<feedback>` des fichiers XML enrichis, l'équipe a élaboré l'*enricher*[1] spécifique à notre projet, dont un extrait est présenté dans l'annexe **H**.

Chaque snippet extrait par le `CRF` passe dans la boucle `for` et rentre d'abord dans le classifieur `Feedback1`. Si l'étiquette attribuée est Feedback1, alors `newFeedback` prend une valeur qui n'est pas `null`. Dans le cas de l'étiquette Autre, `newFeedback` reste `null`, et le *snippet* rentre dans le classifieur `OtherLabels` où une autre étiquette lui est attribuée. Finalement, le troisième classifieur `Polarity` prend en entrée seulement le *snippet* annoté comme Feedback1 ou Feedback2, et lui attribue la polarité positive ou négative.

5.3 Scripts-outils

Pour faciliter l'étape d'évaluation du projet `feedback-classifier` décrite dans le prochain chapitre, nous avons écrit un script qui permet de récupérer certaines informations des fichiers XML de sortie du classifieur. Ce script utilise la classe `WriteFile` (annexe **I**) conçue pendant les cours de Java à l'université. La classe permet de créer un nouveau fichier, d'écrire dedans et de le fermer.

Nous avons écrit la classe `ConvertXml2TxtTraining` qui contient trois méthodes : `extractFeedback`, `compareFilesIdem` et `compareFilesDifferent`.

La première méthode permet d'extraire les labels, la polarité, les catégories, les scores et toute autre information nécessaire (annexe **J**).

Il y a également la méthode `compareFilesIdem` (annexe **K**) qui compare deux fichiers de sortie (sorties des projet `feedback-detection` et `feedback-classifier`, par exemple), et écrit le nouveau fichier qui contiendra tous les *snippets* qui ont été annotés identiquement dans les deux fichiers comparés.

La méthode `compareFilesDifferent` (annexe **L**), comparé à la méthode cité ci-dessus, sort le fichier contenant tous les *snippets* qui ont au moins un paramètre qui ne matche pas dans les deux fichiers comparés. Dans la première colonne, il y a l'étiquette donnée par le premier classifieur, dans la deuxième - le deuxième classifieur, et la troisième colonne contient le *snippet* concerné.

5.4 Conclusion : contributions logicielles et prise de recul

En ce qui concerne nos contributions logicielles au projet `feedback-classifier`, nous avons trouvé les meilleures micro-configurations pour chaque classifieur utilisé par le système. Nous avons

1. Une brique d'enrichissement intégrable dans les chaînes de traitement de la SÉMANTIQUE.

également définit le nombre des classifieurs qui donne les meilleurs résultats, tout en définissant l'ordre de leur application. Tout ceci est décrit dans le chapitre **3**. Nous avons également donné l'idée de l'*enricher* conçu par l'équipe SÉMANTIQUE, en expliquant le fonctionnement de ce dernier que nous voulions avoir. Finalement, nous avons écrit le script permettant de traiter les fichiers de sortie de la *pipeline* principale, en sortant des informations précises. En outre, ce code Java permet de comparer deux fichiers de sortie et d'extraire les *snippets* annotés pareillement ou différemment.

En conclusion, l'approche par cascade à trois classifieurs a montré de bons résultats de ré-étiquetage de *feedbacks*, surtout pour le traitement des corpus déséquilibrés. Nous nous demandons tout de même si nous aurions dû essayer un plus grand nombre de classifieurs, et mettre les Feedback2 à part car ils comptent 40% du corpus `OtherLabels`. Quelles seraient les contraintes d'une telle approche ? Cela pourrait occuper plus de mémoire et le temps de traitement augmenterait. De plus, vu la petite taille du corpus ne contenant pas de Feedback1 et de Feedback2, les résultats n'augmenteraient probablement pas.

Chapitre 6. Évaluation du nouveau système

6.1 Méthodologie d'évaluation

L'évaluation a été faite en trois étapes : une évaluation avec un corpus GOLD, une évaluation avec le corpus de suivi des erreurs du `CRF` et une évaluation avec un corpus d'un nouveau client d'ELOQUANT.

La première étape servait à comparer les performances du projet `feedback-classifier` avec celles du `feedback-detection`.

La deuxième évaluation nous a été nécessaire pour observer le nombre d'erreurs du `CRF` corrigées par le nouveau système de classification.

L'étape finale nous a permis de vérifier si le système `feedback-classifier` induisait de nouvelles erreurs et analyser leur impact sur les sorties.

6.2 Évaluation avec le corpus GOLD

6.2.1 Méthodologie

Le travail consistait en application du script d'évaluation élaboré pour évaluer les résultats du `CRF` seul et du projet `feedback-detection` (système hybride) par la SÉMANTIQUE. L'objectif était de comparer les scores du nouveau système avec ceux de l'ancien, en nous basant sur le même corpus de référence.

Le principal avantage du corpus GOLD est qu'il a été annoté manuellement, donc nous sommes pratiquement sûre des étiquettes attendues. Il s'agit du fichier de 500 *snippets* au format `BIO`.

L'inconvénient de ce corpus est qu'il contient principalement les Feedback1, nous ne pouvons

donc pas nous fier à cette évaluation pour les autres catégories.

6.2.2 Résultats

Comme évoqué auparavant, nous ne pouvons pas interpréter les scores pour toutes les classes, sauf les Feedback1 car, sur 500 énoncés, nous trouvons moins de cinq exemples pour chacune de ces catégories.

En ce qui concerne les Feedback1, les résultats du nouveau système sont très proches de ceux du `CRF` utilisé seul. Nous observons que l'application des règles du système hybride `feedback-detection` fait baisser la F-mesure des Feedback1-NEG de 0.109. En effet, certaines règles rentrent en conflit avec le `CRF` et, puisque l'équipe a accordé la priorité aux règles symboliques, ce sont ces règles qui dégradent les résultats. La décision de garder le système hybride moins performant en détection des Feedback1-NEG est expliquée par la capacité du projet `feedback-detection` de traiter les labels moins représentés. Il sera démontré par la suite que le nouveau système `feedback-classifier` analyse les Feedback1 aussi bien que les autres types de *feedback*.

6.2.3 Conclusion

Comme il n'y a pas de dégradation de résultats pour les Feedback1, nous pouvons garder le nouveau système pour l'analyse des *feedbacks*, tout en améliorant les performances d'analyse des autres labels.

6.3 Évaluation avec le corpus des erreurs du CRF

6.3.1 Méthodologie

L'évaluation sur le corpus de suivi des erreurs s'est faite en deux étapes. Tout d'abord, nous avons analysé les énoncés avec le système `feedback-classifier` utilisant uniquement les annotations sémantiques issues du projet `feedback-detection`. En analysant les erreurs du nouveau système, nous avons créé de nouvelles règles symboliques, afin de corriger les cas les plus récurrents. Nous avons refait l'évaluation pour observer l'impact des nouvelles règles.

6.3.2 Résultats

La première étape de l'évaluation nous a démontré que notre système corrigeait 46% des erreurs du `CRF`, soit 166 *snippets* sur 358. Dans la plupart des cas, il s'agissait des erreurs de polarité.

Après la création de nouvelles règles d'annotation sémantique, nous avons réussi à corriger au total 77% des erreurs présentes dans le corpus de suivi, soit 274 énoncés sur 358.

6.3.3 Conclusion

Cette évaluation nous a permis de confirmer que le système `feedback-classifier` apportait de nombreuses corrections des erreurs du `CRF`, grâce à ses particularités logicielles et les nouvelles

ressources sémantiques.

6.4 Évaluation avec le corpus d'un nouveau client

6.4.1 Méthodologie

Nous avons démontré que le nouveau système avait corrigé 77% des erreurs de l'ancien, mais cela ne signifie pas qu'il n'en crée pas d'autres. C'est pourquoi, nous avons comparé les sorties du système `feedback-detection` avec celles de `feedback-classifier` sur les mêmes données en entrée. Il s'agissait du client à qui ELOQUANT allait proposer l'analyse des *feedback*.

Le corpus contenait 903 *snippets*. Puisque nous ne pouvions pas vérifier chaque énoncé un par an dans les deux fichiers de sortie, nous nous sommes intéressée aux cas du désaccords des deux systèmes. En effet, lorsque les deux programmes sont d'accord, peu importe s'il y a des erreurs ou non car cela ne pourra pas dégrader l'analyse encore plus. Par contre, s'ils ne sont pas d'accord, il faut décider quel est le système qui donne les bonnes réponses le plus de fois. Les cas de désaccord, nous les avons vérifiés manuellement.

6.4.2 Résultats

Finalement, dans le cas des 653 *snippets*, les systèmes `feedback-detection` et `feedback-classifier` ont attribué les mêmes étiquettes.

Pour les 250 *snippets* qui restaient, dans 57% des cas, le nouveau système a attribué la bonne étiquette, dans 29% - les deux systèmes se sont trompés et seulement dans 14% des énoncés, le système `feedback-detection` a fait le bon étiquetage.

6.4.3 Conclusion

Dans les 72% des cas, les deux systèmes sont d'accord, nous pouvons donc dire que le projet `feedback-classifier` n'y introduit pas de nouvelles erreurs.

En ce qui concerne les 28% restants, nous faisons plus confiance au nouveau système qu'à l'ancien car il se trompe beaucoup moins souvent.

En conclusion, nous pouvons affirmer que le système `feedback-classifier` est plus performant et donne de meilleurs résultats que `feedback-detection`.

Conclusion

Les résultats du projet `feedback-classifier` ont démontré l'importance des annotations sémantiques au sein d'un système de classification multi-classes. Dans le cas des classifieurs binaires, il est préférable de se contenter de quelque règles qui corrigeraient les erreurs les plus récurrentes.

La deuxième étape d'évaluation décrite dans la section **6.2** révèle que le *machine learning* utilisant des vecteurs sémantiques est plus performant que le système qui sépare le système hybride en deux

parties : l'apprentissage automatique d'un côté, et les règles expertes d'un autre. En effet, la priorité donnée aux règles symboliques risque de dégrader les résultats du système en cas des conflits.

Finalement, l'approche par cascade de classifieurs est très utile, lorsque les corpus d'apprentissage sont très déséquilibrés. Elle permet d'harmoniser les corpus utilisés pour l'entraînement de chaque classifieur, ce qui améliore considérablement les performances du système conçu.

Partie 3

-

Feedbacks : Identification de l'intensité

Chapitre 7. Introduction

Dans cette partie, nous allons rentrer dans les détails du package `intensity`. Nous parlerons des ressources sémantiques implémentées et de leur impact sur le système. Par la suite, nous parlerons de l'approche logicielle appliquée et des deux algorithmes testés. Nous finirons par expliquer quelque perspectives du projet.

Chapitre 8. Approche linguistique

8.1 Choix de la graduation de l'intensité

Afin de concevoir la taxonomie pour ce projet, nous devions définir la graduation de l'intensité à partir des besoins des clients d'Eloquant. Compte tenu de la situation sanitaire dans le pays au moment où nous entamions cette partie du stage, nous n'avons pas pu avoir de retours clients à temps. Nous avons donc décidé de fixer la graduation de l'intensité à deux niveaux : le *snippet* serait annoté comme INTENSE ou NON-INTENSE, correspondant à leur définition initiale dans le schéma des *feedbacks*. Les étiquettes correspondent respectivement à l'intensité forte ou faible d'après Melnikova (2017). Cette relative simplicité nous arrange, puisque les classifieurs binaires montrent généralement de meilleurs résultats face aux classifieurs multi-classes.

8.2 Étude et préparation des corpus

En analysant le grand corpus annoté pour l'entraînement du `CRF` du projet `feedback-detection`, nous avons observé de nombreux désaccords entre annotateurs. En outre, le même annotateur n'attribuait pas la même étiquette au même *snippet* tout au long de l'annotation du corpus. Ce phénomène peut être expliqué, en premier lieu, par la subjectivité de perception de l'intensité de chacun. Finalement, le même individu peut également être influencé par un ou plusieurs facteurs extérieurs, comme l'humeur, le niveau de fatigue, etc. L'annotation de l'intensité se faisait via le cochage d'une case *Intense* relativement discrète dans l'interface de l'annotation, ce qui a sans doute engendré des omissions. D'autres facteurs, comme le contexte englobant du *snippet*, ont probablement pu influer sur la décision.

8.2.1 Étude des cas de désaccord

Compte tenu de la subjectivité de perception de l'intensité, nous avons fait une étude sur un corpus de 200 *snippets*, afin de comprendre comment gérer les cas de désaccord. Pour ce corpus, nous avons pris 50 énoncés qui étaient, selon nous, à 100% INTENSE, 50 énoncés à 100% NON-INTENSE, et 100 énoncés où il y avait un désaccord entre les annotateurs.

Nous avons demandé à sept membres d'ELOQUANT de ré-annoter le corpus de 200 *snippets*, individuellement et indépendamment les uns des autres. Une des personnes (consultante CX) avait une consigne particulière : il fallait annoter comme INTENSE seulement les *snippets* qui intéresseraient potentiellement les clients comme *alertes*.

Enfin, nous avons fait un vote général entre ces sept ensembles d'annotations pour faire le choix définitif d'étiquette pour tous les cas problématiques. Par ailleurs, nous avons remarqué que la consultante CX, ayant la consigne particulière, était le plus d'accord avec le vote général. En effet, nous avons calculé le coefficient de corrélation de Pearson pour chaque participant de l'expérience. Le corpus de référence (les étiquettes attribuées par premiers annotateurs aux mêmes *snippets*) corrélait relativement peu avec le vote général, donc nous ne pouvions pas nous fier à ces annotations.

Corpus	Coefficient de Pearson
Corpus de référence	0.35
Corpus annoté par la consultante CX	0.87

TABLE 8.1 – Corrélations entre les annotations d'intensité de référence et les annotations de la consultante CX avec le vote de sept personnes sur 200 *snippets*.

8.2.2 Préparation des corpus

Pour le classifieur de l'intensité, nous avons préparé deux corpus d'entraînement, les deux équilibrés [1].

Le premier corpus contient 620 *snippets*. Pour le constituer, nous avons parcouru le corpus annoté pour l'entraînement du `CRF`, et nous avons extrait les exemples des *snippets* intenses et non intenses les plus récurrents. En effet, le grand corpus contient beaucoup de doublons.
Étant donné son petit volume, nous avons pu le vérifier manuellement, et surtout le normaliser : tous les cas de désaccord sont corrigés. Par exemple, le *token Parfait* a été soit INTENSE, soit NON-INTENSE, selon les annotateurs. Dans le corpus d'entraînement conçu, toutes les occurrences de ce *token* sont considérées comme intenses après le vote général.

Puisqu'un corpus de petite taille n'est pas le mieux adapté à l'apprentissage automatique, nous avons également constitué un corpus plus grand, comptant 4422 *snippets*. Chaque occurrence de ce corpus n'a pas été vérifiée manuellement, nous sommes donc consciente de la présence des doublons mais aussi des désaccords entre les annotateurs. Ces incohérences au sein du corpus de *training* vont perturber et biaiser l'apprentissage machine.

Comme il est difficile de vérifier rapidement un corpus de telle taille manuellement, nous avons eu l'idée de nous laisser aider par la machine pour cette tâche. Nous voulions appliquer le modèle entraîné sur 620 *snippets* (données fiables) sur les 4422 énoncés non contrôlés, et récupérer uniquement les *snippets* ayant les étiquettes attribuées avec le score de probabilité au-dessus de 0.95. Avec ces *snippets*, que nous aurions tout de même révisé manuellement, nous comptions enrichir le corpus de 620 énoncés. Puis, nous aurions entraîné le nouveau modèle avec le corpus enrichi et l'aurions

1. 50% des étiquettes INTENSE et 50% NON-INTENSE.

appliqué au reste du grand corpus. Finalement, il aurait fallu refaire la même manipulation jusqu'à ce qu'il n'y ait plus de *snippets* non vérifiés, avec l'idée qu'avec chaque nouvel ajout de données validées au modèle d'entraînement, le nombre de corrections à faire sur les meilleurs candidats suivants diminuerait.

8.3 Ressources sémantiques

Il est important de mentionner que nous n'avions pas travaillé sur l'élaboration de nouvelles ressources sémantiques pour le projet de détection de l'intensité car leur impact aurait été minimal sur les performances du système. En effet, les scores de la F-mesure étaient très élevés, en utilisant le *machine learning* seul.

Nous avons tout de même intégré les *gazetteers* créés par Melnikova (2017) pour son projet `opinion-detection`. Afin d'inclure ces ressources dans notre projet, nous avons simplement remplacé les étiquettes 1 et 2 de l'ancien projet par NON-INTENSE et INTENSE respectivement.

```
1 désespérément   NON-INTENSE
2 misérablement   NON-INTENSE
3 ardemment   INTENSE
4 énergiquement   INTENSE
5 éperdument  INTENSE
```

FIGURE 8.1 – Extrait du fichier `fr_gaz_adj_intensity.txt`

Il en va de même pour les *TokensRegex*, aucune règle n'a été élaborée pour le projet de détection de l'intensité.

8.4 Impact des annotations sémantiques sur les performances du système

Les annotations sémantiques intégrées dans le projet n'ont eu aucun impact sur les performances du système. Cela peut être expliqué par l'absence de l'étape d'analyse des erreurs du *machine learning* de notre projet et de l'élaboration des règles corrigeant ces erreurs d'étiquetage. Les annotations sémantiques de Melnikova (2017) se sont révélées sans utilité, nous soulignons, *concrètement dans notre projet*, probablement parce que le lexique permettant de bien séparer les deux classes était déjà présent dans les *snippets*, et que les autres termes de Melnikova (2017) ne s'y trouvaient que trop peu souvent pour avoir un impact visible. En outre, les scores de notre système étant élevés de base, nous avons remarqué dans la partie **2** que l'impact d'enrichissement sémantique est moins important dans ce cas.

Chapitre 9. Approche *machine learning*

9.1 Algorithme *machine learning* choisi

L'algorithme utilisé pour le classifieur `Intensity` est le même que celui utilisé pour l'entraînement des modèles des classifieurs `Feedback1`, `OtherLabels` et `Polarity`.

9.1.1 Tests de différentes configurations de la *pipeline*

9.1.1.1 Méthodologie

Afin de choisir la meilleure configuration pour les classifieurs de l'intensité, nous avons effectué les mêmes tests que pour les classifieurs des *feedbacks*. La méthodologie est décrite dans la section **3.2.3.3**.

9.1.1.2 Résultats

Pour le petit corpus de 620 *snippets*, la meilleure configuration est celle qui utilise les *gazetteers*, mais les scores ne différent pas beaucoup avec ceux de la configuration de base.

La meilleure configuration pour le corpus de 4422 *snippets* est celle qui n'utilise pas de *features* supplémentaires, donc la configuration de base.

9.1.1.3 Conclusion

En conclusion, l'inclusion de différentes *features* dans notre classifieur d'intensité n'apporte pas d'amélioration des performances du système. Au contraire, dans le cas du corpus de 4422 *snippets*, les *features* supplémentaires dégradent les résultats. Nous gardons donc la configuration de base pour les deux modèles.

9.2 Autre classifieur testé : Mallet

Outre que la classe principale utilisée pa l'équipe SÉMANTIQUE, nous avons testé une autre librairie de classifieurs : `Mallet` [1] (McCallum, 2002). À la différence du premier classifieur, `Mallet` peut prendre en compte tous les n-grammes de *tokens* et non seulement de groupes nominaux. En ce qui concerne l'algorithme, nous avons choisi le `MaxEnt` qui est souvent utilisé dans la classification de textes.

1. La documentation sur `Mallet` se trouve sur le site suivant : http://mallet.cs.umass.edu/ (consulté le 03/09/2020), et la dernière vesion en date ici : https://github.com/mimno/Mallet (consulté le 03/09/2020).

9.2.1 Méthodologie

9.2.1.1 Préparation des corpus

Le format de corpus d'entraînement utilisé pour `Mallet` est différent : le corpus doit être au format `.txt` et contenir trois colonnes :

1. Id du *snippet* ;
2. Étiquette ;
3. *Snippet*.

Les colonnes doivent être séparées par des tabulations. En outre, nous avons dû enlever le caractère « - » de l'étiquette NON-INTENSE. Voici un extrait du corpus d'entraînement utilisé pour `Mallet` :

```
304 INTENSE une expérience assez exceptionnelle
305 INTENSE Je le recommande fortement
306 NONINTENSE  Le vendeur n'est pas très professionnel
307 NONINTENSE  Pas très à l'écoute du client.
```

FIGURE 9.1 – Extrait du corpus d'entraînement pour `Mallet`.

Comme le classifieur principal d'ELOQUANT enlève les doublons dans le corpus de *training*, nous l'avons également fait pour `Mallet`, afin d'avoir les mêmes données en entrée et de pouvoir comparer les résultats.

9.2.1.2 Nombre d'itérations

En ce qui concerne le nombre d'itération d'entraînement des modèles, nous en avons gardé cinq, comme pour tous les autres classifieurs.

9.2.2 Résultats

Nous constatons que le classifieur principal traite légèrement mieux les corpus de petite taille que `Mallet` : sa F-mesure est supérieure à celle de `Mallet` de 0,035 . En revanche, les deux classifieurs ont la même performance sur un grand corpus.

Chapitre 10. Conclusion

Faute de temps, nous n'avons pas pu aller jusqu'au bout du projet. C'est pourquoi, dans cette section nous parlerons de ce que nous aurions aimé faire.

Tout d'abord, il aurait fallu constituer un grand corpus de *snippets* fiablement annotés par leur intensité. La méthodologie de l'approche est décrite dans la section **8.2.2**. Un tel corpus nous aurait

probablement permis d'avoir de meilleurs résultats, en évitant les conflits d'étiquettes pour un même *snippet* qui se trouverait plusieurs fois dans le corpus.

Une fois le premier modèle entraîné, il aurait été nécessaire d'analyser les erreurs du système et de créer de nouvelles ressources sémantiques qui seraient adaptées à ce projet en particulier.

L'étape suivante serait de refaire tous les tests de configurations détaillés dans la section **3.2.3.3**, afin de choisir la meilleure.

Finalement, il faudrait concevoir l'*enricher* qui intégrerait toutes les analyses dans la *pipeline* principale du projet `feedback-classifier`.

Nous parlerons d'autres perspectives de ce projet dans la section **1.2** du bilan de stage.

Partie 4

-

Travaux transverses et introductifs

Chapitre 11. Analyse des conversations téléphoniques

11.1 Introduction

En attendant le corpus qui devait servir à élaborer une classification générique d'appels téléphoniques, il nous a été proposé d'analyser les données du premier client pour lequel ELOQUANT propose une analyse sémantique des conversations. Il s'agit d'une entreprise française spécialisée en confort thermique dans des résidences individuelles ou collectives. La particularité de ce client est que l'analyse en question est faite sur mesure et ne peut être applicable qu'à ses données.

Le travail qui nous a été donné comportait trois parties : la familiarisation avec la technologie utilisée pour l'entreprise en question en particulier, l'étude des données audio et la mise en place de la classification multi-labels pour ce client, afin de voir si les résultats de la classification s'améliorent.

11.2 Familiarisation avec les outils

Dans cette partie, nous allons présenter l'outil qui permet de passer des transcriptions automatiques des conversations téléphoniques à des fichiers analysés sémantiquement. Le projet dont nous parlons comporte la partie `java` contenant les fichiers permettant de tourner le programme ainsi que la partie `resources` englobant les fichiers décrivant et spécifiant l'analyse.

La *pipeline* principale `.java` permet de définir le dossier `input`, le dossier `output` ainsi que la méthode qui sera utilisée pour le traitement du premier dossier. Il est possible de faire une analyse de tout le répertoire ou bien d'une seule phrase (cette dernière doit être écrite en dur dans le code). Lors de l'application du traitement sur tous les fichiers du dossier `input`, il existe la possibilité d'effectuer une impression de toutes les étapes de la *pipeline* dans la console, si besoin. Ces étapes d'analyse sont décrites dans le fichier `.properties` dans `resources`, elles peuvent être facilement modifiées. En effet, la *pipeline* en question permet de (1) segmenter le texte en phrases, (2) *tokéniser* les phrases, (3) attribuer les étiquettes morpho-syntaxiques, (4) construire un arbre de dépendances syntaxiques, (5) effectuer les différentes analyses sémantiques via les *gazetteers*, les règles sémantiques génériques ou spécifiques au domaine du client, la classification automatique, l'extraction automatique de concepts (selon Sclano & Velardi, 2007). Les détails sur ces différentes briques, ainsi qu'un aperçu de la restitution de ces analyses peuvent être trouvés dans l'article de Dusserre et al. (2020).

Le projet comportent également des répertoires qui entraînent des modèles qui seront utilisés par le système. Les fichiers d'entraînement se trouvent dans des dossiers incluant le mot `training` dans leurs noms. Quant au format des modèles, il s'agit des fichiers `.ser`.

11.3 Étude du corpus

L'étude du corpus des conversations téléphoniques spécifique devait nous amener à répondre à ces quatre questions :

1. La classification actuelle est-elle pertinente ?
2. Pouvons-nous séparer automatiquement des parties thématiques pour chaque conversation ?
3. Lorsque des opinions sont exprimées, pouvons-nous observer leur éventuelle évolution au fil de la conversation ?
4. Quel fournisseur (parmi deux) de transcription automatique propose de meilleurs résultats ?

11.3.1 Classification actuelle pour ce client

La taxonomie du projet en question comprend deux niveaux : les catégories-*mères* et les catégories-*filles*. De plus, le projet contient deux classifications distinctes : une pour le service avant-vente (AVV) et une pour le service après-vente (SAV). L'objectif de notre première étude du corpus était de valider ou non la pertinence des taxonomies proposées. Dans le cas où la classification actuelle ne couvre pas toutes les thématiques ayant un intérêt sémantique particulier pour ce domaine, il aurait fallu l'élargir.

Après avoir vérifié 13 fichiers AVV et 15 fichiers SAV analysés sémantiquement, nous avons pu constater que la classification était, dans l'ensemble, pertinente et couvrait la totalité de chaque conversation. Pourtant, une grande partie de tours de parole n'avaient pas été annotés à cause des transcriptions automatiques de mauvaise qualité. En effet, dans plusieurs fichiers, les locuteurs *Client* et *Agent* ont été inversés, la diarisation [1] des locuteurs était également mauvaise.

Conseiller *Entreprise* : voilà j'ai recherché.
Conseiller *Entreprise* : sèche-serviette donc je suis allée sur votre site.
Conseiller *Entreprise* : y'proposer un test.
Conseiller *Entreprise* : avec bon ce qu'on souhaiter quoi la surface de la pièce et cetera et cetera.
Conseiller *Entreprise* : donc à la sortie en me préconise le modèle *NomDuModele* pivotantes.
Client : oui.
Conseiller *Entreprise* : et quand je vais sur la fiche produits.
Conseiller *Entreprise* : ce produit il a ce produit n'est plus commercialisé alors je voulais savoir par quel type de l'appareil il était remplacé le monde alors qui le remplace.
Client : on n'a pas de modèle qui remplace maintenant on a pu *NomDuModele* pivotantes par contre on a *NomDuModele* étroit donc c'est la même chose.

De plus, étant donné que le système de transcription automatique n'est pas entraîné avec les données d'une entreprise spécialisée en chauffe-eaux, beaucoup de mots spécifiques au domaine ne sont

1. Identification du nombre des locuteurs et attribution des tours de parole aux bons locuteurs.

pas reconnus.

papi votants ⟶ pas pivotant

et trois ⟶ étroit

anglais ⟶ onglet

Cela pose un problème dans l'analyse sémantique car les mots en question ne sont pas vides de sens et devraient donc nous amener vers la bonne catégorie. C'est pourquoi, certains tours de parole ne sont pas catégorisés, l'analyse sémantique, par conséquence, ne peut pas être complète. Alors, nous avons décidé de revoir les transcriptions et de faire une comparaison de deux fournisseurs de transcription automatique afin de voir lequel des deux nous offre les transcriptions de bonne qualité et les mieux adaptées au domaine du projet. Les résultats de cette comparaison seront décrits dans la section **11.3.4**.

11.3.2 Analyse générique de la structure des conversations du client

Après avoir analysé les corpus AVV et SAV, nous avons pu définir une structure qui est plutôt bien respectée dans la plupart des conversations. Celle-ci est très générique, elle peut facilement être appliquée aux deux corpus. Il est possible qu'elle soit également applicable aux corpus des conversations téléphoniques dans le domaine de la relation client. Voici la structure :

1. Exposition du problème ou demande de renseignement par le client de l'entreprise.
2. Demande d'informations sur le client pour la création du dossier par le conseiller.
3. Examen du problème ou de la demande en détails et proposition de la solution ou une promesse d'action (rappeler par téléphone, envoyer un mail, etc.).
4. Fin de la conversation.

Nous nous sommes demandé si les parties des conversations pourraient être délimitées par un nombre défini de tours de parole, ce qui nous aurait permis d'extraire facilement uniquement les parties intéressantes pour l'analyse sémantique. Ce n'est malheureusement pas possible car la longueur des conversations varie, l'ordre d'apparition de chaque bloc sémantique n'est toujours pas suivi. Nous pouvons tout de même essayer de nous baser sur le vocabulaire spécifique à chaque partie afin d'effectuer l'extraction.

11.3.3 Évolution des opinions au fil de la conversation

En ce qui concerne les opinions, elles sont exprimées très rarement dans les conversations AVV et SAV. Dans tous les cas rencontrés, nous n'avons pas pu suivre l'évolution des opinions : si le client qui appelle a pu obtenir toutes les réponses à ces questions, il aura une opinion positive de l'entreprise.

Lorsque le client exprime une opinion négative, c'est souvent lié à l'impossibilité du conseiller de résoudre le problème.

11.3.4 Évaluation de la qualité des transcriptions et de la diarisation automatique des fournisseurs X et Y

Comme il a été évoqué dans la section **11.3.1**, les transcriptions actuelles étaient de mauvaise qualité, ce qui dégradait la qualité de l'analyse sémantique des conversations téléphoniques [2]. Nous avons donc décidé de comparer les transcriptions automatiques de deux entreprises (X et Y) spécialisées dans ce domaine, afin de choisir la meilleure.

La première étape du travail était d'analyser une transcription automatique pour chaque entreprise faite avec une conversation *stylisée*, artificielle [3] évoquant le domaine des assurances maladie. L'audio était de très bonne qualité, les locuteurs étaient bien séparés (pas de chevauchement), ils articulaient bien. Pour un humain, un tel audio ne pose aucun problème pour la transcription.

Tout d'abord, nous avons vérifié la bonne séparation des locuteurs.
En ce qui concerne la transcription X, il n'y a eu qu'une seule erreur de séparation sur 18 tours de parole prévus de base. Quant à l'entreprise Y, leur transcription automatique contenait 16 erreurs de séparation de locuteurs sur 18 tours de parole.
Nous avons, par la suite, regardé le nombre d'erreurs lexicales dans les transcriptions. Étant donné que ce travail a été fait manuellement, nous n'avons pas calculé la précision et le rappel mais avons regardé globalement si une transcription automatique est meilleure que l'autre. Finalement, la transcription X contenait deux fois moins d'erreurs que celle de l'entreprise Y.
Par conséquent, nous avons trouvé que l'entreprise X nous permettait d'avoir de meilleurs résultats lorsque nous travaillons avec des conversations artificielles.

Naturellement, la prochaine étape était de tester les transcriptions de ces deux entreprises avec des « vraies » données du domaine du client spécialisé en chauffage des immeubles. Cette fois-ci, nous avons sélectionné 45 enregistrements SAV. Il y a eu plusieurs facteurs liés au signal acoustique qui pouvaient dégrader la qualité des transcriptions automatiques :

- mauvaise qualité du signal : qualité téléphonique [4], en mono (ce qui requiert une phase de séparation automatique des locuteurs) ;
- le client parle avec un accent (prononciation non standard) ;
- les locuteurs utilisent des acronymes spécifiques au domaine de chauffage ;
- beaucoup de chevauchements dans les discours de l'agent et du client, ce qui gène la séparation automatique des locuteurs ;
- les deux locuteurs ont des voix très proches [5], ce qui gène la séparation automatique des locuteurs ;

2. Une liste des difficultés de l'analyse sémantique appliquée aux transcriptions de centres d'appels peut être trouvée dans l'article de Kalitvianski et al. (2020).
3. La conversation est enregistrée par des collègues du MARKETING.
4. 8 Khz, 64 Kbit/seconde, encodé en G.711.
5. Leurs fréquences fondamentales sont proches.

— la conversation peut être coupée par la musique lorsque le conseiller met le client en attente.
Bien entendu, ces facteurs ne sont pas tous présents dans tous les enregistrements recueillis. Pourtant, l'apparition d'un d'eux peut impacter la qualité des transcriptions.

Dans notre étude, nous avons évalué deux paramètres : la séparation des locuteurs (est-ce que les tours de parole sont bien identifiés et attribués aux bons locuteurs ?) et la fidélité lexicale (est-ce que le contenu lexical est suffisant pour la compréhension du sens de la phrase et sa bonne classification ?). Les deux paramètres de chaque transcription ont été notés sur **5**.
Afin d'évaluer les deux systèmes de transcription automatique, nous avons calculé la moyenne et la médiane des notes pour chaque paramètre. Voici les résultats :

	X		**Y**	
	Séparation Loc	Fidélité lexicale	Séparation Loc	Fidélité lexicale
Moyenne	4.5	3.9	1.8	3.5
Médiane	5	4	1	4

TABLE 11.1 – Les notes de la diarisation et de fidélité lexicale.

Comme nous pouvons le voir, le système de transcription automatique X est significativement meilleur en séparation des locuteurs que le système Y. Quant à la fidélité lexicale, les deux transcripteurs ont des notes similaires. Pourtant, la transcription automatique ayant une mauvaise séparation des locuteurs devient simplement illisible pour un humain, il est donc plus difficile de trouver des erreurs du système d'analyse sémantique automatique et de les corriger manuellement. De plus, la bonne attribution des tours de parole à chaque locuteur devient indispensable lorsque l'entreprise-client veut obtenir les informations concernant seulement leurs clients ou seulement leurs conseillers téléphoniques.

Nous sommes tout de même allée plus loin qu'une simple notation des paramètres évoqués auparavant. Pour une analyse plus objective, nous avons définit les plus grands défauts de chaque système de transcription automatique, afin de trouver des solutions que nous pourririons apporter.
En ce qui concerne le système X, certains mots, surtout à la fin de phrase, ne sont pas transcrits, tandis que le système Y transcrit tout, même s'il se trompe. Le rappel des transcriptions X est donc plus faible. De plus, le système X réussit à *deviner* moins de mots spécifiques que le transcripteur Y. Cela pourrait être expliqué par le fait que le corpus d'entraînement du système X a été plus générique, tandis que le système Y a été entraîné avec un corpus issu d'un domaine technique. Finalement, comme le transcripteur X ne connaît pas le mot *Gmail*, il le remplace souvent par *j'aime*, ce qui peut fortement impacter l'analyse d'opinions (et empêcher de correctement anonymiser l'adresse email du client). Afin de résoudre ces problèmes, nous essayerons d'élargir la liste des mots spécifiques que nous pouvons intégrer dans le système, et nous verrons, par la suite, si les notes de la fidélité lexicale augmenteront.
Quant au système Y, son plus grand défaut est la mauvaise séparation des locuteurs qui dégrade l'analyse sémantique. La seule solution est de solliciter l'entreprise afin qu'ils résolvent le problème en question. Pour le moment, leurs transcriptions sont difficilement exploitables par notre système

d'analyse sémantique.

Finalement, cette étude comparative montre que le système X convient plus à notre projet car donne une meilleure lisibilité des transcriptions.

Chapitre 12. Enrichissement sémantique du projet pour une Maison de Luxe

Le projet pour une Maison de Luxe nous a été très important car il nous a permis de prendre en main les outils d'enrichissement sémantique, en particulier l'analyse des *feedbacks*, ainsi que la CLI [1]. Cette dernière est nécessaire pour tester les nouvelles règles sur des énoncés écrits dans la console.

L'analyse des *feedbacks* devait être appliquée aux données du client en question. La particularité de ce client était que le nom d'un de ses produits risquait d'être interprété comme un *feedback*. Il nous a donc été confié de faire des *TokensRegex* qui permettraient de ne pas considérer ce nom comme un *feedback*.

Nous avons écrit cinq règles qui traitent de différents contextes où l'appellation du produit apparaît. En raison de la confidentialité du client, nous ne pouvons pas donner les exemples de ces règles, mais nous énumérons tous les contextes traités dans la liste ci-dessous :

- traitement de tous les produits dérivés de la marque portant ce nom ;
- différentes versions du nom du produit qui incluent la suite de *tokens* correspondant au *feedback* ;
- dans le cas du nom de parfum, le contexte du verbe *porter* ;
- présence d'un déterminent ou d'une préposition avant le nom du produit ;
- traitement du contexte où le nom du produit est suivi de *a ma préférence*.

Nous insistons sur l'importance de ce projet d'enrichissement sémantique pour la Maison de Luxe car, tout d'abord, nous avons compris le fonctionnement des *TokensRegex*, en mettant la main à la pâte. En outre, travailler pour un client en particulier et savoir que nos règles seront appliquées tout de suite dans l'analyse de l'équipe Sémantique nous a vraiment motivée pour la suite du stage.

Chapitre 13. Reconnaissance de noms de logiciels dans des *verbatims* courts

Un des clients d'Eloquant réalise des enquêtes auprès de ses employés concernant les logiciels utilisés quotidiennement au travail. Les employés peuvent choisir un logiciel dans un long référentiel déroulant, ou bien écrire son nom dans un champ de texte libre lorsqu'ils ne le trouvent pas dans la liste. Il nous a été demandé de faire une étude d'un ensemble de ces réponses-*verbatims*, afin de voir combien d'entre eux contiennent un nom de logiciel présent dans le référentiel fourni. Le référentiel

1. *Command line interface.*

contenait 573 noms d'applications, ainsi que leurs synonymes (par exemple, Navigateurs (Internet Explorer, Firefox, ...)). Le corpus à étudier comptait 1242 *verbatims* qui devraient contenir les noms mentionnés dans le référentiel mais parfois orthographiés différemment.

Le client souhaitait donc reconnaître un maximum de noms de logiciels dans ses *verbatims*, y compris lorsque ceux-ci étaient mal orthographiés [1]. Il nous a été accordé une journée pour faire une analyse de ces deux fichiers et de dire dans combien de *verbatims* il était possible de les reconnaître automatiquement (on appelle cette opération *reclassification*). Comme le travail d'analyse a dû être manuel, nous avons choisi d'étudier 10% du corpus, à savoir 125 *verbatims* choisis aléatoirement à l'aide d'un script. Pour chaque énoncé, nous avons manuellement cherché le nom correspondant dans le référentiel. Un autre obstacle nous empêchant de donner un pourcentage précis de *verbatims* reclassables était l'absence d'une liste de synonymes plus complète et de certaines précisions de la part du client.

Nous avons tout de même pu fournir une estimation des *verbatims* reclassables. Finalement, 59 énoncés, soit 47.2% du total de 125 *verbatims*, ont été considérés comme non reclassables car absents du référentiel. 14 *verbatims*, soit 11.2 % du corpus, sont présents tels qu'ils sont dans le référentiel et ne nécessitent donc pas un reclassement. Quant aux énoncés reclassables, ils comptent 25.6% du total, à savoir 32 *verbatims*. Encore 18 *verbatims*, soit 14.4%, seraient potentiellement reclassables, si le client donne plus de précisions. Finalement, 2 énoncés, soit 1.6% du total, sont partiellement reclassables (par exemple, dans l'énoncé *Internet Explorer et Acrobate* le premier nom est référencé, mais pas le deuxième). Certains *verbatims* que nous ne pouvions pas classer ne précisent pas les noms de logiciels, comme dans l'exemple ci-dessous.

Tous les applis concernant mon poste de travail

Les chiffres issus de cette étude convenaient au client, et il était enthousiaste de mettre en place le système très rapidement, ce qui a été fait par nos collègues. Le service a été mis en production, et le référentiel a été continuellement enrichi depuis.

Chapitre 14. Post-édition des *verbatims* traduits du français vers le russe

Actuellement, l'équipe Sémantique travaille sur le projet d'analyse de *verbatims* en cinq langues étrangères : l'anglais, le russe, l'italien, l'allemand et le portugais. Il s'agit de l'extraction des concepts, de la classification multi-labels et de l'analyse d'opinions. L'équipe étudie la possibilité d'utiliser la traduction automatique pour analyser les *verbatims* écrits dans une langue autre que le français : les *verbatims* seraient traduits vers le français pour, par la suite, être analysés par la *pipeline* de la Sémantique. En revanche, cette étude requiert une estimation de la perte d'informations utiles pour

1. Par rapport à leur forme canonique donnée dans le référentiel.

ces analyses induite par un passage par la traduction automatique (Kalitvianski & Padro, 2020) [1].

Comme ELOQUANT n'avait pas de données en langues étrangères, notre tuteur de stage a traduit automatiquement 500 *verbatims* rédigés par des francophones vers les cinq langues mentionnées ci-dessus. Pour la traduction automatique, ont été utilisés deux systèmes de traduction automatique. Chacun des systèmes a traduit au total 250 *verbatims*.

L'étape suivante de l'expérience était la post-édition des *verbaims* traduits par des étudiants bilingues de GRENOBLE ÉCOLE DE MANAGEMENT [2]. Il s'agissait des locuteurs natifs de chaque langue cible. Quatre post-éditeurs pour l'anglais, trois pour le russe, trois pour l'italien, deux pour l'allemand et deux pour le portugais ont été retenus pour faire le travail de post-édition des traductions automatiques. L'objectif principal de ce travail a été de corriger les erreurs de la traduction automatique, en cherchant à conserver autant que possible le sens, mais aussi la forme du *verbatim*, à savoir les particularités et les imperfections de l'écriture. Cela dans l'optique d'obtenir des textes équivalents aux textes en français originel, et qui seraient ensuite rétrotraduits automatiquement vers le français, afin de comparer les analyses des textes originels et des traductions automatiques des textes dans ces autres langues.

Pour s'assurer du respect de la consigne de la part des post-éditeurs du russe, il nous a été confié de vérifier leur travail. Étant donné que notre langue première est le russe, et que nous avons une bonne maîtrise de l'orthographe et de la grammaire, notre profil convenait parfaitement à cette tâche.

L'image illustrant la structure du document à analyser se trouve en annexe **M**. Les deux colonnes que nous devions comparer étaient la **Source FR** et la **Post-édition RU**.

Un des post-éditeurs des *verbatims* russes s'est désisté au début de l'expérience, donc n'y a pas participé. Finalement, seulement deux post-éditeurs ont travaillé sur ces énoncés. En analysant leurs documents, nous nous sommes rendu compte que la consigne n'a pas été respectée. Le premier post-éditeur a, dans la plupart des cas, fait du simple copier-coller des traductions automatiques. De plus, la ponctuation n'a pas été respectée. Nous avons donc eu beaucoup de travail de correction et de post-édition. L'image ci-dessous illustre le nombre de corrections [3] en moyenne que nous avons dû apporter à chaque *verbatim*. Cet exemple prouve la présence du copier-coller car la phrase *être chez Orange devient pénible* est traduite comme *être dans une orange fait très mal*. Les commentaires sont inutiles...

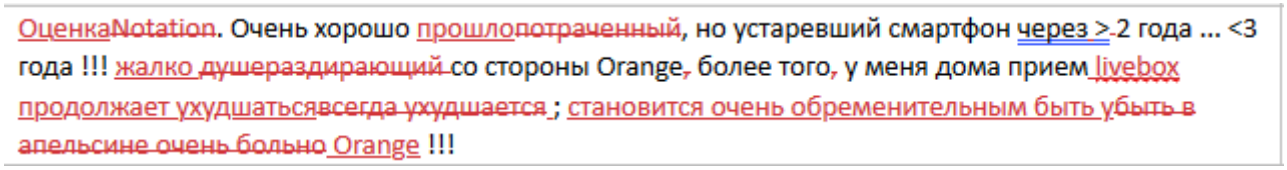


FIGURE 14.1 – Extrait du document de post-édition des *verbatims* russes du premier post-éditeur.

1. L'article a récemment été soumis à un atelier à COLING 2020.
2. https://www.grenoble-em.com/ (consulté le 03/09/2020).
3. Marquées en rouge.

En ce qui concerne le deuxième post-éditeur, il a fait un travail honnête mais n'a pas pu non plus respecté la consigne. En effet, il n'a pas gardé ni le style, ni la ponctuation des *verbatims* français. Finalement, les traductions obtenues appartenaient plutôt à un registre soutenu.

Conclusion générale et perspectives

1 Bilan du stage

Dans cette section, nous allons, tout d'abord, expliquer ce que nous avons pu apporter à l'entreprise pendant notre stage. Dans un second temps, nous parlerons des améliorations possibles de notre travail et de ses perspectives. Nous conclurons par un bilan personnel du stage, en explicitant ce que cette expérience de six mois nous a apporté.

1.1 Contributions à la Sémantique

Tout d'abord, nous avons fourni à la Sémantique une nouvelle version du projet d'analyse des *feedbacks*. Cette version apporte de nombreuses corrections des erreurs de l'ancien système : le nouveau projet gère mieux la polarité, permet d'intégrer facilement de nouvelles ressources sémantiques, ce qui améliore la performance du *machine learning*. Au total, nous avons réussi à corriger 77% des erreurs du `CRF` sans pour autant induire de nouvelles erreurs importantes, du moins à ce stade. Bien évidemment, le temps et les nouvelles données des clients permettront de juger plus objectivement les performances du nouveau système. Les résultats de nos évaluations sont tout de même prometteuses. Le projet `feedback-classifier` servira en tant que base pour la version 2 du projet d'analyse des *feedbacks*, ce que nous considérons comme une grande réussite de notre stage.

Nous avons également élaboré un projet qui sera une base solide pour la détection de l'intensité. Notre étude auprès de l'équipe a démontré la perception très subjective de cette notion linguistique, mais elle nous a également permis de faire des choix dans les cas difficiles, grâce au vote général. Finalement, nous avons constaté que le choix des marqueurs de l'intensité, ainsi que de sa graduation, doit être fait en fonction des besoins des clients et non sur un simple appui sur une analyse des études linguistiques à ce sujet. Le dernier doit être complémentaire au premier car l'objectif principal d'une entreprise est la satisfaction des clients. Le produit doit être non seulement performant mais également utile.

En ce qui concerne le projet d'analyse des conversations, notre étude des ressources scientifiques et des corpus d'un client ont servi comme un point d'appui pour le projet qui se construit actuellement. Nous avons défini les difficultés liées à l'analyse des conversations téléphoniques, en proposant plusieurs solutions de leur traitement automatique. Nous avons également comparé les transcriptions automatiques proposées par deux entreprises externes, en identifiant leurs points forts et leurs défauts nécessitant des améliorations.

Enfin, nous avons participé à d'autres projets de l'équipe Sémantique, en leur apportant de l'aide et notre analyse linguistique. Nous avons élaboré des *TokensRegex* pour le traitement des données de la Maison de Luxe, avons fait une étude de faisabilité pour un autre client et, finalement, avons effectué la post-édition des *verbatims* traduits en russe. Ce dernier travail a été particulièrement important pour l'équipe car il leur a permis d'avoir des traductions de qualité et de mieux comprendre les performances d'analyse des *verbatims* rédigés dans cette langue si différente des autres langues romanes.

1.2 Perspectives des travaux effectués

Étant donné le temps alloué et la qualité des données fournies pour notre travail, nous n'avons pas pu effectuer d'autres tests, et nos projets ont de nombreuses perspectives. Elles concernent la quantité et la qualité des corpus, les tests de nouveaux *features*, le ré-entraînement des outils et la prise en compte des métadonnées.

Tout d'abord, il est nécessaire de continuer à annoter les corpus utilisés pour l'entraînement des modèles des systèmes d'analyse des *feedbacks* et de détection de l'intensité. Grâce aux données déjà étiquetées, il est possible de faire une première annotation automatique des corpus restants. Ainsi, le travail humain se réduirait en grande partie à la révision et à la correction des étiquettes attribuées par le système à base de l'apprentissage automatique. Il est également utile d'enrichir les corpus d'entraînement, de les rendre encore plus variés, afin de couvrir un plus grand nombre de cas de variation linguistique, ainsi que de pouvoir proposer l'analyse des *feedbacks* et de l'intensité à d'autres types de clients que les clients actuels de l'entreprise.

Durant notre stage, nous avons remarqué que TALISMANE, l'étiqueteur morpho-syntaxique utilisé par la SÉMANTIQUE, n'effectue pas toujours une bonne analyse des *verbatims*, en attribuant une étiquette incorrecte à certains *tokens*, ce qui empêche à certaines *TokensRegex* de s'appliquer. Dans le cas de nos projets, nous avons dû inclure dans plusieurs règles des étiquettes morpho-syntaxiques supplémentaires (par exemple, les mots inconnus au TALISMANE sont classés comme NC [4], même s'il peut s'agir d'un verbe ou une autre catégorie grammaticale), ce qui nous a permis de traiter les *snippets* que nous voulions analyser. Tout de même, cet ajout de catégories grammaticales supplémentaires peut également induire de nouvelles erreurs. Il serait donc intéressant de ré-entraîner TALISMANE avec de nouveaux corpus qui seraient plus spécifiques au domaine cible. Une autre piste serait de tester d'autres étiqueteurs morpho-syntaxiques, afin de trouver le plus adapté aux données spécifiques des clients d'ELOQUANT.

En ce qui concerne les solutions algorithmiques, il est possible d'explorer d'autres implémentations du `CRF`, qui sauraient intégrer les annotations sémantiques utilisées par l'équipe. Il faudrait tester d'autres algorithmes d'apprentissage automatique, tels que les réseaux de neurones, par exemple, qui pourraient aussi apporter de meilleurs résultats au traitement des *verbatims*.

Quant à la détection de l'intensité, il paraît très prometteur d'étudier l'impact des *features* graphiques à l'analyse. Par exemple, les majuscules peuvent jouer un rôle important et indiquer la présence de l'intensité dans le *snippet* à analyser. Nous notons également que le *verbatim* écrit en majuscules peut avoir une telle forme graphique simplement parce que l'auteur a oublié de désactiver le *Caps Lock*, alors l'intégration d'un tel *feature* peut induire du bruit à l'analyse. Il est alors nécessaire de faire une étude statistique sur un grand corpus, afin de définir la pertinence de l'ajout de ce trait. Par ailleurs, l'analyse des *emojis* ou des répétitions d'une lettre au sein d'un *token* peut s'avérer très utile à la détection de l'intensité, voire à l'analyse des *feedbacks*. Finalement, une étude pragmatique auprès des clients sur leurs besoins est incontournable avant le développement de ce projet.

Pour le projet de l'analyse des *feedbacks*, il est intéressant de tester d'autres *features* que ceux

4. Nom commun.

employés actuellement par le système. Par exemple, au lieu d'utiliser les n-grammes de *tokens*, tester les n-grammes de caractères qui capteraient mieux les racines invariantes des mots utiles. Il est également intéressant d'étudier si l'intensité peut être exploitée comme un *feature* sémantique supplémentaire pour l'étiquetage des *feedbacks*. Par exemple, les Feedback2 pourraient être plus souvent intenses que les autres *feedbacks*.

Les métadonnées peuvent également s'avérer utiles à l'analyse d'opinions. Par exemple, la note de satisfaction accompagnant certains *verbatims* pourrait être très informative pour la détection des *feedbacks* et de leur intensité.

Il faut voir si et comment l'analyse des *feedbacks* peut être appliquée au traitement des conversations téléphoniques. Nous notons tout de même que, d'après notre étude des corpus SAV et AVV, les opinions sont très rarement exprimées et n'évoluent quasiment jamais au fil du dialogue. De plus, les phrases ne sont pas tournées de la même façon que celles des *verbatims*, certaines règles symboliques pourraient alors être inutiles ou demanderaient des adaptations.

2 Bilan personnel

Le stage de six mois effectué au sein de l'entreprise ELOQUANT a été une expérience très enrichissante pour mon profil professionnel. J'ai acquis de nouvelles compétences techniques, professionnelles et personnelles qui me seront utiles lors de mon futur parcours.

Tout d'abord, le travail avec les outils d'ELOQUANT codés en Java m'a permis de mieux comprendre le fonctionnement des langages orientés objet. J'ai appris à faire les *gazetteers* et les *TokensRegex* qui peuvent être utiles à l'analyse automatique du langage naturel. J'ai également été conduite à faire des scripts permettant d'automatiser certaines tâches très utiles pour l'évaluation des grandes quantités de données. J'ai pu travailler avec des données de nature différente, opposer le traitement de l'écrit à celui de l'oral, me rendre compte des difficultés liées à ces types d'analyse. Finalement, je me suis familiarisée avec de nouveaux algorithmes et des approches d'évaluation que nous n'avions pas étudié profondément auparavant à l'université.

En ce qui concerne d'autres compétences, j'ai beaucoup pratiqué le travail en équipe, et c'est grâce à la coopération de toute l'équipe que nous avons pu atteindre de beaux résultats. J'ai également appris à être autonome et à m'adapter aux conditions de travail compliquées, vu la situation sanitaire dans le pays. L'organisation, le savoir demander de l'aide lorsque j'avais des questions et des doutes, vite repérer les points forts de chaque coéquipier, toutes ces compétences ont été acquises lors du stage et me seront indispensables dans mon futur professionnel.

En conclusion, le stage chez ELOQUANT m'a apporté de la confiance en moi, de nouvelles connaissances des domaines de l'informatique et de la vie de l'entreprise. Cette expérience a été une des plus riches de tout mon parcours universitaire et me servira en tant que tremplin pour mon avenir.

Bibliographie

Bost, X., Senay, G., El-Bèze, M. & Mori, R. D. (2015). Multiple topic identification in human/human conversations. *CoRR*, *11*(34), 18-42. Récupérée 3 septembre 2020, à partir de https://arxiv.org/pdf/1812.07207.pdf

Chang, A. X. & Manning, C. D. (2014). TokensRegex : Defining cascaded regular expressions over tokens. *Stanford University Computer Science Technical Reports. CSTR*, *2*. Récupérée 3 septembre 2020, à partir de https://nlp.stanford.edu/pubs/tokensregex-tr-2014.pdf

Chicco, D. & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*(1), 6. Récupérée 3 septembre 2020, à partir de https://link.springer.com/content/pdf/10.1186/s12864-019-6413-7.pdf

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297. Récupérée 3 septembre 2020, à partir de https://link.springer.com/content/pdf/10.1007%252FBF00994018.pdf

Delgado, R. & Tibau, X.-A. (2019). Why Cohens Kappa should be avoided as performance measure in classification. *PloS ONE*, *14*(9), e0222916. Récupérée 3 septembre 2020, à partir de https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0222916&type=printable

Dusserre, E., Kalitvianski, R., Ruhlmann, M. & Padró, M. (2020). Analyse sémantique de transcriptions automatiques d'appels téléphoniques en français. *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 31e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 4 : Démonstrations et résumés d'articles internationaux*, 14-17. Récupérée 3 septembre 2020, à partir de https://hal.archives-ouvertes.fr/hal-02768497v3/document

Forcada, M. L., Ginest-Rosell, M., Nordfalk, J., ORegan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martnez, F., Ramrez-Sánchez, G. & Tyers, F. M. (2011). Apertium : a free/open-source platform for rule-based machine translation. *Machine translation*, *25*(2), 127-144.

Gaatone, D. (2007). Les marqueurs d'intensité et les locutions verbales : quelques réflexions. *Travaux de linguistique*, *2*(55), 93-105. Récupérée 3 septembre 2020, à partir de http://www.cairn.info/revue-travaux-de-linguistique-2007-2-page-93.htm

Haykin, S. (2009). *Neural networks and learning machines.* New York : Prentice Hall, Récupérée 3 septembre 2020, à partir de http://repository.fue.edu.eg/xmlui/bitstream/handle/123456789/3421/5618.pdf?sequence=1

Jain, A. K. (2010). Data clustering : 50 years beyond K-means. *Pattern recognition letters*, *31*(8), 651-666. Récupérée 3 septembre 2020, à partir de https://www.cs.odu.edu/~sampath/courses/w17/cs599/papers/reading7/45.pdf

Kalitvianski, R., Dusserre, E. & Padró, M. (2020). Promises and Disappointments of Semantic Analysis of Speech-To-Text Applied to Call Center Conversations in an Industrial Setting. *A paraitre dans Proceedings of the Industry Track of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*.

Kalitvianski, R. & Padró, M. (2020). Multilingual Semantic Analysis via Machine Translation. *Article soumis et en cours de revue à un atelier à Coling 2020*.

Kleiber, G. (2013). À la recherche de l'intensité. *Langue française*, *1*(177), 63-76. Récupérée 3 septembre 2020, à partir de http://www.cairn.info/revue-langue-francaise-2013-1-page-63.htm

Lafferty, J., McCallum, A. & Pereira, F. C. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. Récupérée 3 septembre 2020, à partir de https://repository.upenn.edu/cgi/viewcontent.cgi?article=1162&context=cis_papers

Lailler, C., Estève, Y., De Mori, R., Bouallègue, M. & Morchid, M. (2015). Utilisation d'annotations sémantiques pour la validation automatique d'hypothèses dans des conversations téléphoniques. *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles. Articles courts*, 185-191. Récupérée 3 septembre 2020, à partir de https://www.aclweb.org/anthology/2015.jeptalnrecital-court.28.pdf

Lavalley, R. (2012). *Extraction automatique de segments textuels, détection de rôles, de sujets et de polarités* (thèse de doct.). Récupérée 3 septembre 2020, à partir de http://www.theses.fr/2012AVIG0182/document

McCallum, A. K. (2002). Mallet : A machine learning for language toolkit. Récupérée 3 septembre 2020, à partir de http://mallet.cs.umass.edu

Melnikova, E. (2017). *Mise en place de grammaires pour la détection automatique des émotions dans le corpus de Relation Client* (mém. de mast.).

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*. Récupérée 3 septembre 2020, à partir de https://arxiv.org/pdf/1301.3781.pdf

Nigam, K., Lafferty, J. & McCallum, A. (1999). Using maximum entropy for text classification. *IJCAI-99 workshop on machine learning for information filtering*, *1*(1), 61-67. Récupérée 3 septembre 2020, à partir de http://www.kamalnigam.com/papers/maxent-ijcaiws99.pdf

Plutchik, R. (1982). A psychoevolutionary theory of emotions.

Ramond, A. (2016). *Intérêt de l'enrichissement sémantique pour une tâche de catégorisation de textes courts par méthode hybride avec peu de données d'entraînement* (mém. de mast.). Récupérée 3 septembre 2020, à partir de https://dumas.ccsd.cnrs.fr/dumas-01366802/document

Sclano, F. & Velardi, P. (2007). Termextractor : a web application to learn the shared terminology of emergent web communities. *Enterprise Interoperability II* (p. 287-290). Springer. Récupérée 3 septembre 2020, à partir de https://pdfs.semanticscholar.org/cb00/a8e57bb30763f8abb1de2f7c26a4afefb304.pdf

Searle, J. R. (1969). *Speech acts : An essay in the philosophy of language* (T. 626). Cambridge university press.

TLFi : Trésor de la langue Française informatisé [Beau]. (2020). ATILF - CNRS Université de Lorraine. Récupérée 3 septembre 2020, à partir de http://atilf.atilf.fr/

Urieli, A. (2013). *Robust French syntax analysis : reconciling statistical methods and linguistic knowledge in the Talismane toolkit* (thèse de doct.). Récupérée 3 septembre 2020, à partir de https://tel.archives-ouvertes.fr/file/index/docid/1058143/filename/Urieli_Assaf.pdf

Zhang, L. & Ferrari, S. (2014). Intensité et polarité : un modèle opératoire articulant plusieurs travaux linguistiques. *Langue française*, *4*(184), 35-54. Récupérée 3 septembre 2020, à partir de http://www.cairn.info/revue-langue-francaise-2014-4-page-35.htm

Table des figures

Liste des tableaux

Table des annexes

Annexe A. Plateforme de restitution

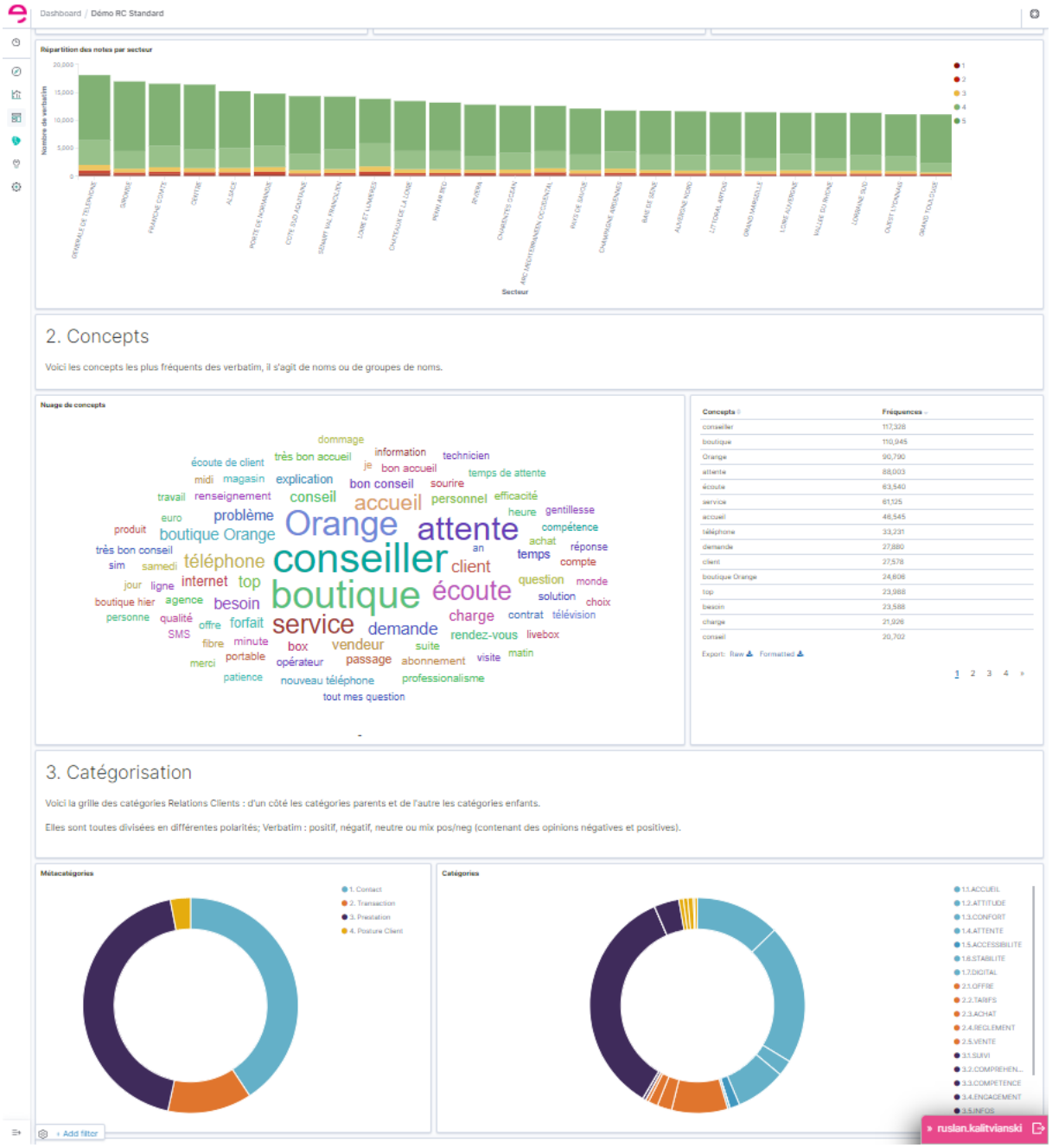
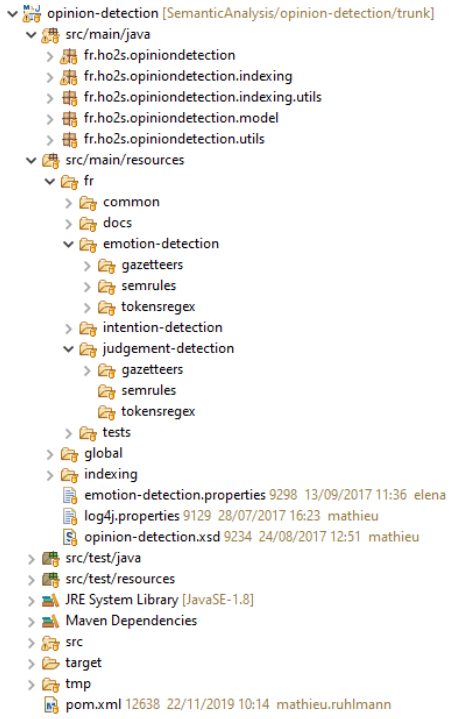
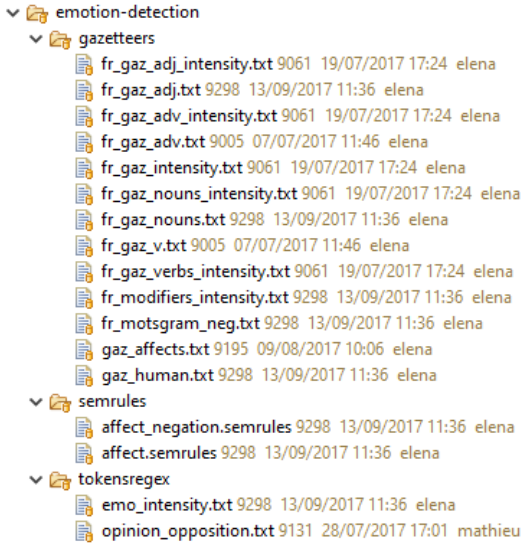


Illustration de quelque analyses de *verbatims* sur la plateforme de restitution.

Annexe B. Structure du projet `OPINION-DETECTION`

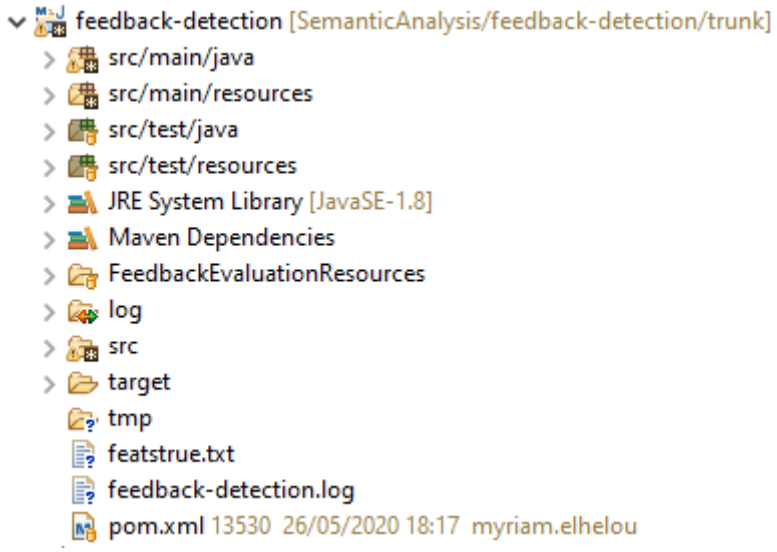


Structure du projet `opinion-detection` de Melnikova (2017).

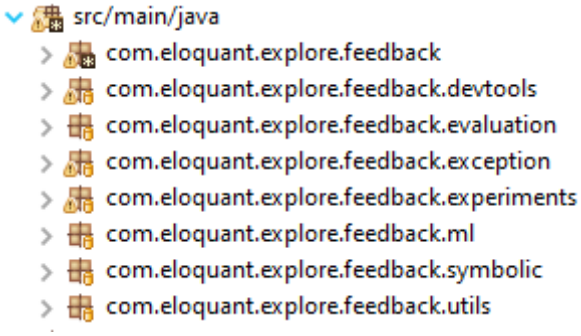


Contenu du dossier `emotion-detection` du projet `opinion-detection`.

Annexe C. Structure du projet FEEDBACK-DETECTION

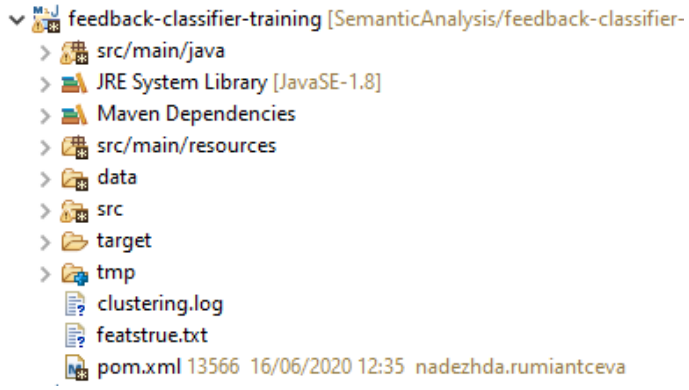


Structure du projet `feedback-detection`.

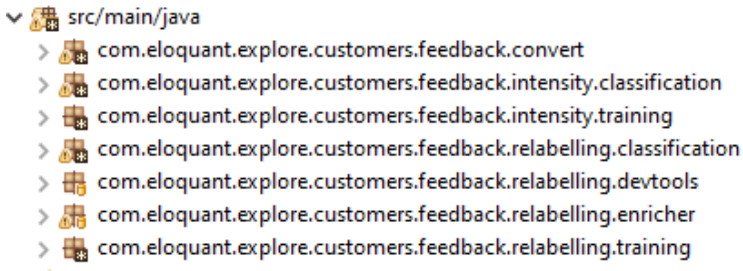


Contenu du dossier `src/main/java`

Annexe D. Structure du projet FEEDBACK-CLASSIFIER



Structure générale du projet `feedback-classifier`.



Différents *packages* du projet `feedback-classifier`.

Annexe E. Fichier d'entraînement de modèles

```
1  PACKAGE COM.ELOQUANT.EXPLORE.CUSTOMERS.FEEDBACK.RELABELLING.TRAINING;
2
3  IMPORT ORG.APACHE.LOGGING.LOG4J.LEVEL;
4  IMPORT ORG.APACHE.LOGGING.LOG4J.CORE.CONFIG.CONFIGURATOR;
5  IMPORT COM.ELOQUANT.EXPLORE.CRMCLASSIFIER.CRMDEFAULTS;
6  IMPORT FR.HO2S.CLASSIFICATION.TRAINING.MAINCLASSIFICATIONTRAINING;
7
8  PUBLIC CLASS FEEDBACKMAINCLASSIFICATIONTRAINING {
9      PUBLIC STATIC VOID main(String[] args) THROWS Exception {
10         Configurator.setRootLevel(Level.DEBUG);
11         IF (args == NULL || args.length <= 0) {
12             args = NEW String[] {
13                 "-taxonomy", "relabelling/fr/taxonomy/Feedback1ClassificationTaxonomy.csv",
14                 "-trainingCsv", "data/relabelling/training/Feedback1Training.csv",
15                 "-parametrizedConfig", "relabelling/fr/feedback_feedback1_classifier_config_fr.xml",
16                 "-resources", CRMDefaults.DEFAULT_LEXICAL_TUNNING,
17                 "-modelId", "Feedback1NoStopwords",
18                 "-sizeQuotaTraining", "0.8",
19                 "-text_normalizer", CRMDefaults.DEFAULT_SPELL_CORRECTIONS,
20                 "-reportRelevantTerms", "data/relabelling/ClassifierTraining/relevant_terms_report.log",
21             };
22         }
23         NEW MainClassificationTraining().doMain(args);
24     }
25 }
```

Exemple du fichier d'entraînement de modèles issu du projet `feedback-classifier`.

Annexe F. Pipeline XML

```
<!--  statistical classifier/trainer config -->
<ANNOTATOR name="holmes">
    <ATTRIBUTE name="grammar">relabelling/fr/feedback_feedback1_classifier_fr.properties</ATTRIBUTE>
    <ATTRIBUTE name="useNPs">true</ATTRIBUTE>
    <ATTRIBUTE name="NPsMaxLen">3</ATTRIBUTE>
    <ATTRIBUTE name="compounderFile">{RESOURCES}/crm_compounds.txt</ATTRIBUTE>
    <ATTRIBUTE name="useRels">true</ATTRIBUTE>
    <ATTRIBUTE name="relConfig">relations-fr.properties</ATTRIBUTE>
    <ATTRIBUTE name="useSemFeat">true</ATTRIBUTE>
    <ATTRIBUTE name="useSemWord">true</ATTRIBUTE>
    <ATTRIBUTE name="usePos">true</ATTRIBUTE>
</ANNOTATOR>
<CLASSIFICATION useLemma="true">
    <STOPWORDSFILE>relabelling/fr/vide.txt</STOPWORDSFILE>
    <IRRELEVANTWORDSFILE>{RESOURCES}/crm_irrelevant.txt</IRRELEVANTWORDSFILE>
    <SYNONYMSFILE>{RESOURCES}/crm_synonyms.txt</SYNONYMSFILE>
</CLASSIFICATION>
```

Extrait du fichier `.xml` utilisé pour le classifieur `Feedback1`.

Annexe G. Micro-configurations

useNPs	gazetteers	TRs	useRels	useSemFeat	useSemWord	usePOS	useLemma	Pr
-	-	-	-	-	-	-	-	
3	-	-	-	-	-	-	-	
3	+	-	-	+	+	-	-	
3	+	-	+	+	+	-	-	
3	+	-	+	+	+	+	-	
3	+	-	+	+	+	+	+	
2	+	-	+	+	+	+	+	
4	+	-	+	+	+	+	+	
3	+	-	+	+	+	+	no Stopwords	
3	+	-	+	+	+	+	no Synonyms	
3	+	-	+	+	+	+	no Stropwords no Synonyms	
3	-	-	+	-	-	+	no Stropwords no Synonyms	
3	+	-	+	+	+	-	no Stropwords no Synonyms	
-	+	-	+	+	+	+	no Stropwords no Synonyms	
3	-	-	+	-	-	+	+	
3	+	-	+	+	+	-	+	
-	+	-	+	+	+	+	+	
3	-	-	+	-	-	+	no Stopwords	

Toutes les micro-configurations testées pour chaque classifieur.

Annexe H. *Enricher*

```
PUBLIC List<Feedback> detect(String text) THROWS FeedbackDetectionException {
    List<Feedback> mlFeedbacks = crfDetector.detect(text);
    List<Feedback> multiMlFeedbacks = NEW ArrayList<>();
    Feedback newFeedback = NULL;
    FOR(Feedback feedback : mlFeedbacks) {
        ClassificationInstance aClassificationInstance = NULL;
        TRY {
            IF(feedback1Classifier!=NULL) {
                aClassificationInstance =    feedback1Classifier.runClassifier(feedback);
                IF(aClassificationInstance!=NULL)
                    newFeedback = completeFeedback(feedback, aClassificationInstance);
                ELSE
                    CONTINUE;
            }
            IF(othersClassifier!=NULL && newFeedback==NULL) {
                aClassificationInstance = othersClassifier.runClassifier(feedback);
                IF(aClassificationInstance!=NULL)
                    newFeedback = completeFeedback(feedback, aClassificationInstance);
                ELSE
                    CONTINUE;
            }
            IF(polarityClassifier!=NULL && newFeedback!=NULL && (newFeedback.getLabel()
            .equals(TAG_TYPE_OPINION_BUZZ) || newFeedback.getLabel().equals(TAG_TYPE_OPINION_FEEDBACK1))) {
                aClassificationInstance = polarityClassifier.runClassifier(feedback);
                IF(aClassificationInstance!=NULL)
                    newFeedback = completePolarityFeedback(newFeedback, aClassificationInstance);
                ELSE
                    CONTINUE;
            }
            IF(newFeedback!=NULL)
                multiMlFeedbacks.add(newFeedback);
        } CATCH (Exception e) {
            e.printStackTrace();
        }
    }
    RETURN multiMlFeedbacks;
}
```

Méthode gérant l'application des classifieurs.

Annexe I. WriteFile

```
1  PUBLIC CLASS WRITEFILE {
2      // classe pour écriture dans un fichier
3      PrintWriter fw;
4      // Constructeur
5      PUBLIC WriteFile(String nom) {
6          TRY {
7              // ouverture du fichier en écriture
8              fw = NEW PrintWriter(NEW BufferedWriter(NEW FileWriter(nom)));
9          } CATCH (IOException e) {
10             System.out.println("Ouverture du fichier impossible en écriture");
11         }
12     }
13     // Méthodes
14     PUBLIC BOOLEAN write(String chaine) {
15         TRY {
16             fw.println(chaine);
17             RETURN (TRUE);
18         } CATCH (Exception e) {
19             RETURN (FALSE);
20         }
21     }
22     PUBLIC VOID close() { // void ne renvoie rien
23         fw.close();
24     }
25 }
```

Classe `WriteFile`.

Annexe J. Méthode extractFeedback

```
PRIVATE STATIC String extractFeedback(DocumentList docs, String text) {
    FOR (Document doc : docs.getDocuments()) {
        FOR (Feedback feedback : doc.getAnalysis().getFeedbackDetection().getFeedbacks()) {
            text = text + feedback.getLabel() + "\t" + feedback.getSnippet() + "\t" + feedback.getScore()
            + "\n";
        }
    }
    RETURN text;
}
```

Méthode permettant d'extraire le label, le *snippet* et le score du *machine learning* de la sortie XML d'un classifieur.

```
<DOCUMENT-LIST id="verbatimToTest">
    <DOCUMENT analyzable="true" id="verbatimToTest_65">
        <METADATA corpus="verbatimToTest" language="fr"/>
        <TEXT>Pas de réponse à 2 mails.</TEXT>
        <ANALYSIS date="2020-07-08T15:11:05.545662400">
            <SENTENCES>
                <SENTENCE id="0" spanFrom="0" spanTo="24" tokenCount="7">Pas de réponse à 2 mails.</SENTENCE>
            </SENTENCES>
            <FEEDBACK-DETECTION>
                <FEEDBACKS>
                    <OPINION type="feedback3" polarity="negative" intensity="1" label="Feedback3"
                    spanFrom="0" spanTo="24" score="0.8313" algorithm="ALGO">
                        <SNIPPET>Pas de réponse à 2 mails</SNIPPET>
                        <LEMMATIZED-SNIPPET>pas de réponse à #NUM# mail </LEMMATIZED-SNIPPET>
                    </OPINION>
                </FEEDBACKS>
            </FEEDBACK-DETECTION>
        </ANALYSIS>
    </DOCUMENT>
</DOCUMENT-LIST>
```

Exemple d'entrée XML.

```
Feedback3   Pas de réponse à 2 mails    SCORE
```

Exemple de sortie avec la méthode `extractFeedback`.

Annexe K. Méthode compareFilesIdem

```
PRIVATE STATIC String compareFilesIdem(DocumentList docs, DocumentList docs2, String text) {
    FOR (INT i = 0; i < docs.getDocuments().size(); i++) {
        Document verbatim = docs.getDocuments().get(i);
        Document verbatim2 = docs2.getDocuments().get(i);
        FOR (Feedback feedback : verbatim.getAnalysis().getFeedbackDetection().getFeedbacks()) {
            FOR (Feedback feedback2 : verbatim2.getAnalysis().getFeedbackDetection().getFeedbacks()) {
                IF (feedback.getSnippet().equals(feedback2.getSnippet())) {
                    IF (feedback.getLabel().equals(feedback2.getLabel())
                        && feedback.getPolarity().equals(feedback2.getPolarity())) {
                        text = text + feedback.getLabel() + " " + feedback.getPolarity() + "\t"
                                + feedback.getSnippet() + "\n";
                        BREAK;
                    }
                }
            }
        }
    }
    RETURN text;
}
```

Méthode `compareFilesIdem`.

```
Feedback1 POSITIF   Mon problème a été résolu facilement
```

Exemple de sortie avec la méthode `compareFilesIdem`.

Annexe L. Méthode compareFilesDifferent

```
PRIVATE STATIC String compareFilesDifferent(DocumentList docs, DocumentList docs2, String text2) {
    FOR (INT i = 0; i < docs.getDocuments().size(); i++) {
        Document verbatim = docs.getDocuments().get(i);
        Document verbatim2 = docs2.getDocuments().get(i);
        FOR (Feedback feedback : verbatim.getAnalysis().getFeedbackDetection().getFeedbacks()) {
            FOR (Feedback feedback2 : verbatim2.getAnalysis().getFeedbackDetection().getFeedbacks()) {
                IF (feedback.getSnippet().equals(feedback2.getSnippet())) {
                    IF (!feedback.getLabel().equals(feedback2.getLabel())
                        || !feedback.getPolarity().equals(feedback2.getPolarity())) {
                        text2 = text2 + feedback.getLabel() + " " + feedback.getPolarity() + "\t"
                                + feedback2.getLabel() + " " + feedback2.getPolarity() + "\t"
                                + feedback.getSnippet() + "\n";
                    }
                }
            }
        }
    }
    RETURN text2;
}
```

Méthode `compareFilesDifferent`.

```
Feedback1 NEGATIF   Feedback1 POSITIF   Personne n'a répondu à ma question
```

Exemple de sortie avec la méthode `compareFilesDifferent`.

Annexe M. Document de post-édition des *verbatims* traduits en russe

ID1	Corpus	Système	Source FR	TA RU	Post-édition RU
1	A	E	Très satisfait. Je suis passé pour régler un problème d'option suite à un changement de titulaire de ma ligne mobile. Prise en charge rapide. Problème réglé. Je suis très satisfait.	Очень доволен Я решил решить проблему с опционами после смены владельца моей мобильной линии. Быстрая поддержка. Проблема решена. Я очень доволен.	Очень доволен. Я пришёл решить проблему с опциями после смены владельца моей мобильной линии. Быстрая поддержка. Проблема решена. Я очень доволен.
2	A	E	Une catastrophe sur cet exemple. Orange a suspendu ma ligne en croyant suspendre celle du voisin ! Plus d'internet alors que je travail en parti à domicile. Après appel, le conseillé m'a promis 50G, que je n'ai jamais eu ! alors que des mails m'informaient que tt était réglé !? Rien est réglé aujourd'hui.	Бедствие на этом примере. Апельсин приостановил мою линию, думая, что она приостанавливает линию моей соседки! Нет больше интернета, пока я работаю частично дома. После звонка консультант пообещал мне 50G, которого у меня никогда не было! когда электронные письма сообщили мне, что тт был улажен!? Сегодня ничего не решено.	Катастрофа на этом примере. Orange приостановил мою линию, думая, что они приостанавливают линию соседа! Нет больше интернета, а я работаю частично дома. После звонка, консультант пообещал мне 50G, которых у меня никогда не было! тогда как электронные письма сообщали мне, что все было улажено!? Сегодня ничего не улажено.
3	A	E	Bon ACCUEIL et service très compétent. Je suis passé hier chez Orange, J'ai été très bien accueilli, et ma demande a été rapidement prise en compte .	Хороший ДОБРО ПОЖАЛОВАТЬ и очень компетентный сервис. Я пошел в Orange вчера, меня очень хорошо приняли, и моя просьба была быстро учтена.	Хороший ПРИЕМ и очень компетентный сервис. Я вчера пошел в Orange , меня очень хорошо приняли, и моя просьба была быстро учтена.

Structure du document de post-édition des *verbatims* traduits du français vers le russe.

Table des matières

MOTS-CLÉS : classification, analyse d'opinions, analyse de l'intensité, enrichissement sémantique, apprentissage automatique, système hybride, règles symboliques, TAL, *verbatim*, conversations téléphoniques

RÉSUMÉ

Depuis 2019, ELOQUANT développe un nouvel outil d'extraction d'opinions, combinant apprentissage automatique et règles linguistiques. La première version ayant une performance inégale selon les entités à relever, et commettant fréquemment des erreurs dans l'identification des opinions, l'équipe SÉMANTIQUE a envisagé de faire appel à des stratégies d'étiquetage complémentaires.

Nous avons développé une deuxième version du projet d'analyse d'opinions, en utilisant une approche permettant de concilier l'apprentissage automatique et l'intégration des annotations sémantiques. Pour l'apprentissage automatique, nous avons reconstitué les corpus d'entraînement, afin de les rendre plus équilibrés. Par la suite, en analysant les erreurs de la version précédente du projet, nous avons adapté les règles symboliques et en avons créé de nouvelles pour apporter le plus d'améliorations possibles. Finalement, l'évaluation de la nouvelle version du projet d'analyse d'opinions nous a permis de démontrer la supériorité de ses performances par rapport à la première version.

Nous avons également travaillé sur la détection de l'intensité pour l'analyse d'opinions. L'étude des ouvrages scientifiques et des désaccords entre les annotateurs des corpus nous a démontré la subjectivité de la notion en question. Nous avons tout de même entraîné plusieurs modèles basés sur différents corpus qui serviront comme base pour le projet en développement.

Finalement, nous avons participé à plusieurs projets d'ELOQUANT, tels que l'analyse des conversations téléphoniques, l'enrichissement sémantique pour une Maison de Luxe, la reconnaissance de noms de logiciels dans des *verbatims* courts et la post-édition des *verbatims* traduits du français vers le russe. Ces projets nous ont aidés à nous familiariser avec les technologies utilisées au sein de l'entreprise.

KEYWORDS : classification, opinion analysis, intensity analysis, semantic enrichment, machine learning, hybrid system, symbolic rules, NLP, verbatim, phone conversations

ABSTRACT

Since 2019 ELOQUANT has been working on new opinion extraction tool, combining machine learning and linguistic rules. As the first version were inconsistent and opinions, often badly identified, the SÉMANTIQUE team considered using complementary labelling strategies.

We have developed a second version of the opinion analysis project, using an approach that reconciles machine learning and the integration of semantic annotations. First, we have reconstructed the training corpora, in order to make them more balanced. Then, the analysis of the errors of the previous version of the project, we adapted the symbolic rules and created new ones to make the tool as many effective as possible. Finally, the evaluation of the new version of the opinion analysis project allowed us to demonstrate its superiority compared to the first one.

We also worked on the intensity detection in opinion analysis. The study of scientific works and disagreements between corpora annotators showed us the subjectivity of this notion. Nevertheless, we have produced several models based on different corpora, which will serve as a basis for the project under development.

Finally, we participated in several ELOQUANT projects, such as the analysis of phone conversations, semantic enrichment for a Luxury House, the recognition of software names in short verbatims and the post-editing of verbatims translated from French into Russian. These projects helped us become familiar with the technologies used within the company.