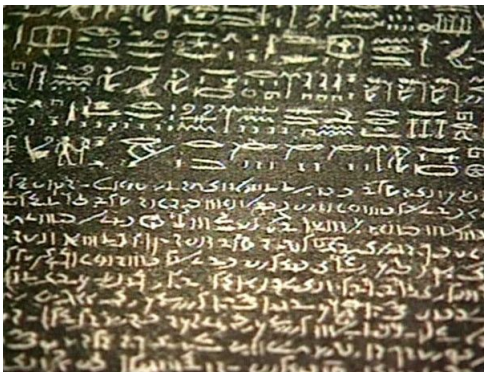
TextMine'20

Pascal Cuxac, Vincent Lemaire

TextMine '20

Atelier sur la Fouille de Textes



Organisateurs :

Pascal Cuxac (INIST - CNRS),
Vincent Lemaire (Orange Labs),

Organisé conjointement à la conférence EGC
(Extraction et Gestion des Connaissances)
le 28 janvier 2020 à Bruxelles

Editeurs :

Pascal Cuxac - INIST - CNRS
2 rue Jean Zay, CS 10310, 54519 Vandoeuvre les Nancy Cedex
Email : pascal.cuxac@inist.fr

Vincent Lemaire - Orange Labs
2 avenue Pierre Marzin, 22300 Lannion
Email : vincent.lemaire@orange.com

Publisher:

Vincent Lemaire, Pascal Cuxac
2 avenue Pierre Marzin
22300 Lannion

Lannion, France, 2020

PRÉFACE

C'est une évidence que de dire que nous sommes entrés dans une ère où la donnée textuelle sous toute ses formes submerge chacun de nous que ce soit dans son environnement personnel ou professionnel : l'augmentation croissante de documents nécessaires aux entreprises ou aux administrations, la profusion de données textuelles disponibles via Internet, le développement des données en libre accès (OpenData), les bibliothèques et archives en lignes, les medias sociaux ne sont que quelques exemples illustrant l'évolution de la notion de texte, sa diversité et sa prolifération.

Face à cela les méthodes automatiques de fouille de données (data mining), et plus spécifiquement celles de fouille de textes (text mining) sont devenues incontournables. Récemment, les méthodes de deep learning ont créées de nouvelles possibilités de recherche pour traiter des données massives et de grandes dimensions. Cependant, de nombreuses questions restent en suspens, par exemple en ce qui concerne la gestion de gros corpus textuels multi-thématiques. Pouvoir disposer d'outils d'analyse textuelle efficaces, capables de s'adapter à de gros volumes de données, souvent de nature hétérogène, rarement structurés, dans des langues variées, des domaines très spécialisés ou au contraire de l'ordre du langage naturel reste un challenge.

La fouille de textes couvre de multiples domaines comme, le traitement automatique des langues, l'intelligence artificielle, la linguistique, les statistiques, l'informatique et les applications sont très diversifiées, que ce soit la recherche d'information, le filtrage de spam, le marketing, la veille scientifique ou économique, la lutte antiterroriste...

Le but de cet atelier est de réunir des chercheurs sur la thématique large de la fouille de textes. Cet atelier vise à offrir une occasion de rencontres pour les universitaires et les industriels, appartenant aux différentes communautés de l'intelligence artificielle, l'apprentissage automatique, le traitement automatique des langues, pour discuter des méthodes de fouille de texte au sens large et de leurs applications.

P. Cuxac — INIST-CNRS
V. Lemaire — Orange Labs

Membres du comité de lecture

Le Comité de Lecture est constitué de:

Guillaume Cabanac (IRIT, Toulouse)

Mariane Clausel (Université de Lorraine, Nancy)

Vincent Claveau (IRISA, Rennes)

Natalia Grabar (STL - Lille3, Lille)

Sonia Le Meitour (Orange Labs, Lannion)

Denis Maurel (Université F. Rabelais, Tours)

TABLE DES MATIÈRES

Exposé Invité

Mise en place d'une chaîne d'évaluation de compréhension du langage naturel : retour d'expérience sur l'assistant vocal Djingo d'Orange
Ghislain Putois 1

Session Exposés

Génération de résumés abstractifs de commentaires sportifs
Aurélien Bossard, David-Stéphane Belemkoabga, Abdallah Essa, Valentin Nyzam, Christophe Rodrigues, Kevin Sylla 5

Extraction du contenu principal de pages web
Guillaume Bruneval, Mohamed Lacarne, Mohamed Kone, François-Xavier Bois, Stanislas Morbieu 17

Classification de phrases courtes : des approches non-supervisées aux approches faiblement supervisées
Kaoutar Ghazi, Sébastien Marchal, Andon Tchechmedjiev, Pierre-Antoine Jean, Nicolas Sutton-Charani, Sébastien Harispe 23

Extraction automatique de noms d'entreprises à partir de titres de presse : un exemple d?application chez ReportLinker
Marilyne Latour, Jocelyn Bernard, Corentin Regal 31

ARES : un extracteur d'exigences pour la modélisation de systèmes
Aurélien Lamercerie 37

Index des auteurs **41**

Mise en place d'une chaîne d'évaluation de compréhension du langage naturel : retour d'expérience sur l'assistant vocal Djingo d'Orange

Ghislain PUTOIS*

*Orange Labs, 2, rue Pierre Marzin, 22300 Lannion, France
ghislain.putois@orange.com

Résumé. Créer un assistant vocal est un projet qui bouscule les cadres et pratiques traditionnelles de l'architecture logicielle, par l'importance que revêtent les interactions avec l'utilisateur dans la qualité de service perçue.

Nous présentons ici un aperçu de la méthodologie que nous avons mise en œuvre dans l'assistant vocal Djingo d'Orange pour évaluer et améliorer la qualité de ces interactions, en mettant l'accent sur la partie compréhension du langage naturel par le système.

1 Architecture d'un assistant vocal

Un assistant vocal est un périphérique de la taille d'un petit haut-parleur qui permet d'accéder en vocal à des services d'information, et de piloter par la voix une télé ou des appareils domotisés. D'un point de vue fonctionnelle, il se décompose en modules : détection de la parole, transcription en mots, compréhension du langage naturel, gestion de dialogue, des composants de service (par exemple musique, nouvelles, météo ...), et synthèse vocale. (voir figure 1)

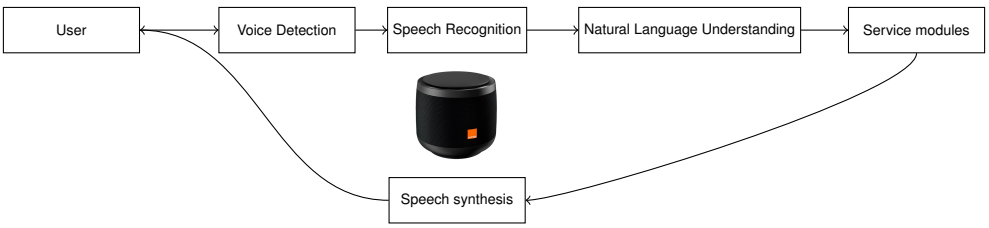


FIG. 1 – *Assistant vocal Djingo et architecture fonctionnelle*

2 Natural Language Understanding

L'interaction avec l'utilisateur est au cœur de la proposition de valeur d'un assistant vocal. Le module de « Natural Language Understanding (NLU) » est l'étape clé qui encode les mots

de l'utilisateur en information structurée exploitable qui représente la demande de service. C'est en fait un système de classification automatique qui va renvoyer une structure de données contenant 3 catégories prédéfinies et hiérarchiques : l'intention de la phrase, appelé ici intent (parmi une liste prédéfinine), une liste de concepts (également parmi une liste prédéfinie), et des valeurs associées extraites du texte.

Dans notre cas, le NLU de Djingo utilise des réseaux de neurones profonds pour prédire l'intention en début de phrase, et un étiquetage par mot, en s'appuyant sur une décomposition de type Begin / Inside / Outside : si un mot du concept est le premier, il est étiqueté « B-*concept* », sinon, il est étiqueté « I-*concept* ». Enfin, si le mot n'appartient pas au concept, il est étiqueté « O » (voir figure 2) :

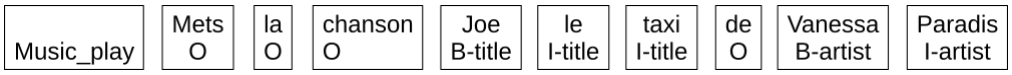


FIG. 2 – *Exemple d'annotation projetée sur les mots : l'intent détecté est « music_play », le concept « title » s'étend sur les mots « Joe le taxi », et le concept « artist » s'étend sur les mots « Vanessa Paradis »*

3 Évaluation du NLU

Pour évaluer un système de classification, il « suffit » de comparer la sortie du système avec sa sortie idéale attendue, et de compter les différences. Ce qui pose de fait trois problématiques.

3.1 Comment compter les différences ?

Les entrées se décomposent en phrases du domaine et phrases hors-domaine : les phrases du domaine représentent les demandes utilisateur que l'assistant est capable de traiter, par exemple : musique, télé, météo ; les phrases hors domaine représentent des demandes comme l'automobile, le spatial.

En sortie, le NLU peut renvoyer soit un rejet (« no intent ») quand il n'arrive pas à interpréter la demande, soit proposer une interprétation avec un score de confiance.

Le croisements de ces entrées/sorties mènent à des catégories de comportement analysables : rejet correct, rejet à tort, acception à tort, interprétation correcte ou confusion (voir figure 3) :

D'un point de vue expérience utilisateur, les différentes erreurs n'ont pas le même impact. Se tromper dans la thématique demandée par l'utilisateur est plus gênant que se tromper dans la reconnaissance d'un titre de chanson ou de ville. De même, se tromper sur un énoncé fréquent est beaucoup plus impactant que se tromper sur un énoncé jamais entendu au préalable.

3.2 Quelle sortie idéale ?

Le développement d'un assistant vocal complexe nécessite de nombreux cycles d'évolution, où sont ajustés les services proposés et les formulations utilisateurs reconnues. En formalisant des guides d'annotations pour chaque cycle, qui documentent les évolutions, il devient

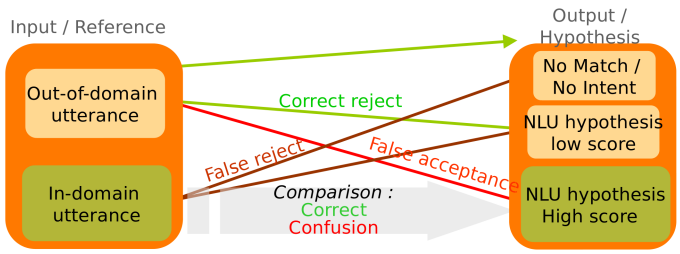


FIG. 3 – *Typologie des erreurs*

plus facile de faire évoluer les annotations de données déjà réalisées pour mieux capitaliser sur les besoins des utilisateurs.

3.3 Sur quelles données ?

Au démarrage du projet, il est nécessaire de s'appuyer sur des premiers énoncés imaginés par les designers du service pour initialiser les premiers modèles par apprentissage. Ensuite, il est essentiel de mettre au plus vite en place une boucle de collecte pour remplacer progressivement les énoncés inventés par des phrases réellement prononcées par les utilisateurs. Cependant, la collecte des énoncés doit respecter strictement la Réglementation Générale sur la Protection des Données. Ce cadre fort complique l'accès et le partage des données, mais est incontournable pour construire un service responsable envers l'utilisateur.

4 La chaîne de traitement

L'évaluation du NLU passe par la mise en place d'une chaîne de traitement orientée donnée, qui contient les cycles de collecte, d'annotation, de création de modèles de classification, et d'analyse métier pour les valider. Cette chaîne de traitement (voir figure 4) présente de nombreuses boucles, et il est très important de tenir compte du parcours de la donnée dans les développements.

La planification d'un nouveau service doit se penser sur un temps long, car elle entraîne des évolutions d'annotation, et demande d'avoir collecté suffisamment d'énoncés de testeurs et être en mesure de réaliser l'évaluation avant que le service atteigne une qualité suffisante pour être ouvert à tous les utilisateurs.

Cette chaîne contient de nombreux cycles de durées différentes : de quelques minutes pour la boucle d'amélioration du modèle NLU sur un jeu d'apprentissage donné, on passe à quelques semaines pour la boucle d'annotations / réannotations des données pour une évolution du service. Le développement d'un assistant vocal demande de bien synchroniser ces cycles, et une forte coopération entre les multiples acteurs.

L'amélioration du modèle statistique peut se réaliser sur des cycles plus courts, à travers de nombreuses expériences hors-ligne à base d'échantillonnage sur les énoncés déjà collectées.

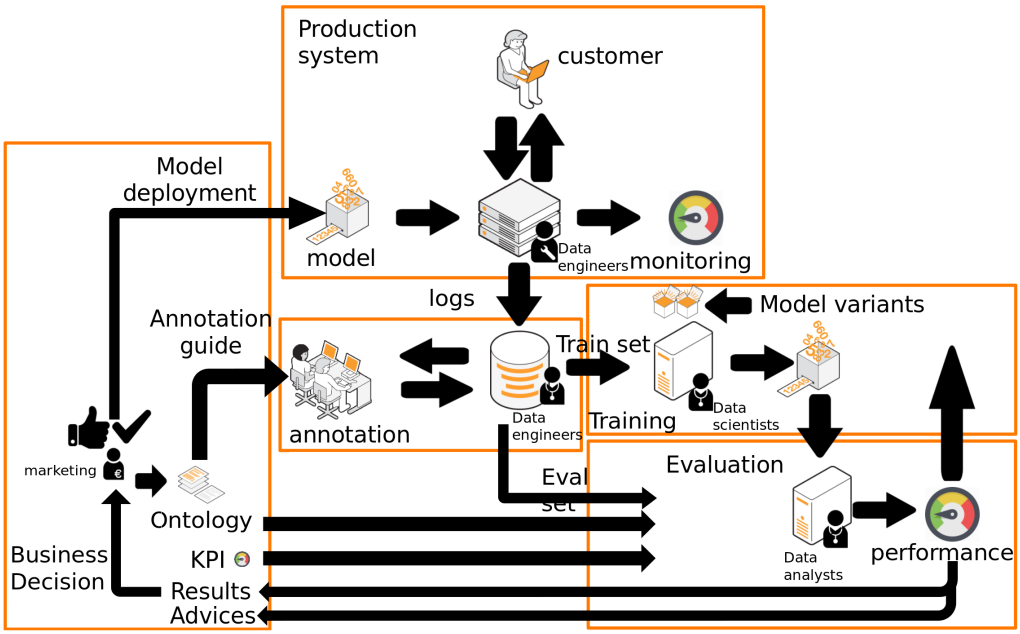


FIG. 4 – *Vue statique de la chaîne de traitement*

5 Performances et perspectives

La mise en place de cette démarche centrée sur les données a permis d'obtenir un gain de performances du composant NLU de plus de 20% pour sa sortie commerciale. Ces performances varient selon la complexité des énoncés, et l'analyse des données a permis la construction d'une métrique d'évaluation en phase avec les usages réels.

La sortie commerciale du produit a marqué un jalon important : notre travail se concentre maintenant sur l'optimisation des capacités d'annotation pour tenir compte de la longue traîne dans les gros volumes d'énoncés, et continuer à faire évoluer modèles statistiques et métriques d'évaluation pour rester pertinent sur les services rendus.

Summary

Designing a vocal assistant is a daunting task, off the beaten track of software developpement uses and methodologies. The major difference lies in the utmost importance of user interaction in the perceived service quality. We present here an overview of how we evaluate and improve the quality of these interactions in the Orange Djingo vocal assistant, by focussing on the Natural Language Understanding part of the system.

Génération de résumés abstractifs de commentaires sportifs

Aurélien Bossard*, David-Stéphane Belemkoabga** Abdallah Essa**, Valentin Nyzam*,
Christophe Rodrigues**, Kevin Sylla**

*Laboratoire d'Informatique Avancée de Saint-Denis (LIASD) : EA 4383
2,rue de la Liberté, 93526 Saint-Denis, France
prénom.nom@iut.univ-paris8.fr
** Léonard de Vinci Pôle Universitaire, Research Center
92916 Paris La Défense, France
prénom.nom@devinci.fr

Résumé. Dans ce papier, nous proposons une méthode permettant de générer automatiquement à partir de commentaires réalisés en direct par des journalistes sportifs un résumé de match de football. Nous montrons que cette tâche difficile met en échec les approches extractives et proposons dans un premier temps un modèle d'apprentissage reposant sur des réseaux de neurones profonds afin de sélectionner les phrases les plus pertinentes. Dans un second temps, cette réduction du bruit sur les commentaires nous permet d'apprendre à générer des résumés abstractifs à l'aide d'un réseau de neurones de type *pointer-generator* et nous montrons l'intérêt de la sélection des phrases pertinentes ainsi que la qualité des résumés créés automatiquement. Nous présentons des premiers résultats encourageants.

1 Introduction

Dans cet article, nous nous intéressons au résumé de commentaires sportifs. Ces commentaires constituent une ressource intéressante, car les commentaires en direct dont nous disposons sont associés au résumé rédigé par le commentateur à la fin de la rencontre, ainsi que de données annexes liées au match. Pouvoir générer automatiquement les commentaires en direct permettrait de décharger le commentateur d'une tâche lourde et ainsi lui libérer du temps pour des tâches plus complexes et valorisantes, comme l'analyse des rencontres. Le résumé de ces commentaires constitue un défi important. Tout d'abord, ils sont générés au fur et à mesure du déroulement du match ; des informations sont donc susceptibles d'être mises à jour en cours de match et d'entrer en conflit les unes avec les autres, ou de se compléter mutuellement. Par exemple, si un joueur de football marque deux buts au cours d'un match, il ne convient pas de simplement sélectionner les deux commentaires concernant les buts (qui constituent sûrement chacun une information pertinente) sous peine d'être extrêmement redondant. De plus, l'ensemble des commentaires d'un match est particulièrement bruité. En effet, les commentaires décrivent l'ensemble du déroulement d'un match. Or, il y aura, sauf exception, plus de tirs non cadrés, d'arrêts du gardien, de déviation en corner, voire de remplacements, que de buts au cours d'un match, qui sont pourtant les informations les plus pertinentes d'un match. Enfin, le

style des résumés rédigés par le commentateur diffère radicalement de celui des commentaires en direct. Ces éléments sont autant de facteurs expliquant le manque de pertinence des modèles extractifs de résumé sur ce type de données. En effet, les modèles extractifs consistent à extraire, depuis les textes source, des phrases jugées pertinentes – souvent par une analyse fréquentielle – et de les assembler pour constituer un résumé. La différence de style entre résumé et commentaires ainsi que le bruit de ces commentaires constituent des freins importants à ces modèles sur ce type de données. Quant aux modèles abstractifs neuronaux, en plein essor ces dernières années, ils sont performants avec des corpus de taille conséquente : plusieurs centaines de milliers de documents. Or, nous disposons seulement de cinq années de matchs de Ligue 1 de football, soit environ 1700 paires document/résumé. De plus, ils sont généralement entraînés en vue de générer des résumés généraux, et non pas des résumés spécifiques de niche comme le résumé de commentaires sportifs.

Pour pallier ces contraintes, nous avons besoin d'entraîner notre propre modèle abstractif (les modèles extractifs étant inadaptés par essence compte tenu de notre contexte). Pour résoudre le problème du manque de données, il nous faut réduire le bruit des données d'entrée du modèle de résumé pour le faire converger plus rapidement et avec moins de données. Nous émettons à cet effet l'hypothèse forte que la variabilité des résumés rédigés par le commentateur est assez faible pour permettre un apprentissage malgré un corpus de taille relativement réduite.

Ainsi, nous proposons une méthode qui vise à réduire l'entropie des données d'entrée pour améliorer les résultats du modèle de résumé. Cette méthode repose sur une sélection préalable des informations les plus pertinentes, dans le but de réduire la taille des textes d'entrée et de s'affranchir de données inutiles à la génération du résumé.

Dans une première section, nous présentons les travaux connexes au résumé de commentaires sportifs. Dans une seconde section, nous présentons notre corpus et ses particularités. Nous décrivons ensuite notre méthode, les expérimentations menées ainsi que les résultats obtenus. Nous terminons l'article par une discussion et nos perspectives.

2 Travaux connexes

La génération automatique de résumés de commentaires sportifs est à notre connaissance un sujet très peu abordé dans la littérature. Nous pouvons principalement citer les travaux de Zhang et al. (2016) qui, à partir de commentaires de match disponibles en ligne, va générer un résumé extractif. La méthode est constituée de trois grandes étapes : une première étape de modélisation des phrases en fonction d'indices de surface définis empiriquement comme la somme des *tf-idf* des mots d'une phrase, la présence ou non de mots importants dans la phrase comme "carton rouge", "but"... Ensuite, à partir d'un ensemble de phrases de référence, une étape d'apprentissage va prédire le score ROUGE des phrases en fonction de ces indices de surface. Enfin, une dernière étape d'ordonnancement va permettre d'incorporer au résumé les phrases avec les meilleurs scores. Selon ces auteurs, l'approche souffre cependant de plusieurs limites. Le processus étant centré sur les phrases, les résumés engendrés comportent beaucoup de redondance entre phrases. De plus, l'apprentissage a tendance à pénaliser les phrases courtes considérées parfois à tort comme moins informatives car leur contribution directe au score ROUGE est moindre.

Sur cette même tâche, nous pouvons citer également les travaux de Bouayad-Agha et al. (2012) ayant pour particularité de proposer un système de résumé génératif. Celui-ci repose sur la définition d'une ontologie spécialisée aux matchs de football. Ainsi, à partir des données extraites depuis des commentaires dans une ontologie, des règles établies à la main vont se déclencher et reformuler ces informations afin de générer un résumé. Les principales limitations de l'approche sont la nécessité d'un peuplement exhaustif de l'ontologie (joueurs, équipes,...) ainsi qu'une génération de résumés stéréotypés car construits à partir des mêmes règles.

Sur la même problématique mais sous un angle très différent, les travaux de Corney et al. (2014) partent des commentaires des utilisateurs de twitter lors de matchs et produisent des résumés subjectifs. A chaque commentaire officiel lié à un événement dans le match, les commentaires des fans sur twitter sont analysés sur une fenêtre de 4 minutes, le but étant d'extraire le tweet le plus représentatif de cet événement du point de vue subjectif des fans de chaque équipe. Pour ce faire, les tweets sont d'abord répartis entre les deux équipes. Simplement, un utilisateur est fan de l'équipe A si dans ses commentaires elle est surreprésentée par rapport à d'autres équipes. Dans une deuxième étape, pour chaque équipe, les sujets les plus importants sont détectés à partir d'une variante du *tf-idf*. Finalement, pour chaque équipe, le tweet le plus représentatif des sujets trouvés est sélectionné, sans traitement supplémentaire.

Compte tenu de la différence de style entre les résumés et les commentaires en direct, du bruit très présent dans les commentaires, ainsi que du faible volume des données, nous choisissons de nous différencier des méthodes précédentes en profitant des récentes avancées en matière de génération neuronale afin d'éviter l'écueil de la génération de résumés stéréotypés. Pour cela, nous abordons le problème sous l'angle de la génération neuronale précédée d'une étape de réduction du bruit dans les textes source afin de permettre au modèle génératif de converger rapidement.

3 Corpus

Notre corpus est composé de 1700 paires de commentaires de matchs de football de Ligue 1 et du résumé manuel correspondant, issus du site de L'Equipe (`http://www.lequipe.fr`) soit l'intégralité des matchs sur une période de cinq saisons, de 2014-2015 à 2018-2019. Les commentaires de matchs sont composés en moyenne de 8000 mots dans 80 interventions du commentateur, et les commentaires en direct sont entrecoupés de faits concernant des acteurs du match : informations sur un transfert récent, sur une série de buts marqués en match... Les résumés manuels sont quant à eux composés de 55 mots en moyenne (cf. Figure 1) Les commentaires sont donc d'une taille bien supérieure aux 400 premiers mots d'un document utilisés par la plupart des méthodes abstractives récentes pour générer un résumé. La complexité d'une tâche de génération automatique d'un résumé fondée sur de tels commentaires est par conséquent beaucoup trop importante eu égard au faible nombre de documents qui composent notre corpus.

La figure 1, qui présente les trois dernières minutes de commentaires en direct du match PSG-Lille du 22 novembre 2019, montre bien le bruit de ces documents. On y voit notamment un sondage posté par le commentateur entre la 90+1' et la 90+2' minutes, ainsi que trois commentaires que l'on peut considérer comme du bruit dans le cadre de l'élaboration d'un résumé court : l'annonce du temps additionnel ainsi que trois actions non conclues. On peut aussi remarquer la distance entre le style du résumé et le style des commentaires.

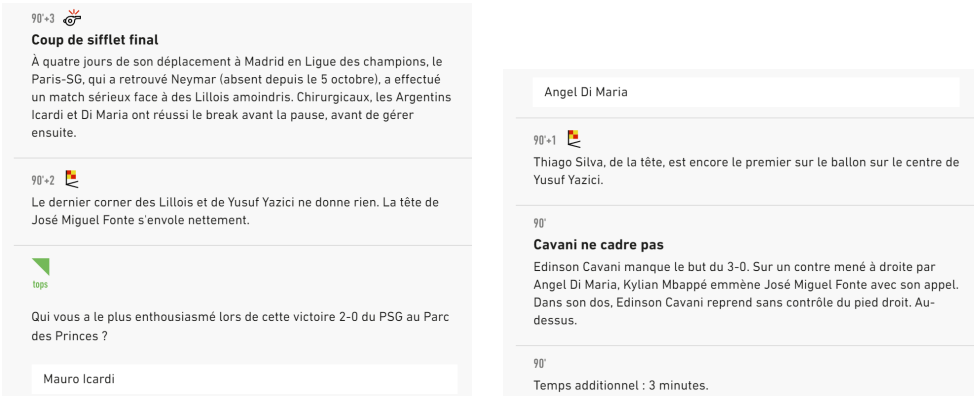


FIG. 1: Exemple d'un commentaire en direct et de son résumé (en haut à gauche, 90'+3) extraits du site Lequipe.fr

Les spécificités de ces documents nous obligent à repenser le processus de résumé automatique, en adoptant d'abord une étape de sélection des informations importantes, suivie d'une étape de génération de résumé, afin de se rapprocher le plus possible du style d'un résumé manuel.

4 Notre méthode

Résumer automatiquement des commentaires de match de football présente une difficulté majeure : les techniques classiques fréquentielles d'évaluation de l'importance d'un mot ou d'une phrase sont rendues inopérantes par la nature même des documents considérés : les informations les plus importantes d'un match sont souvent les plus rares – le résultat, les expulsions, les buts.

Cela ajouté au style du résumé, radicalement différent de celui des commentaires, nous a poussés à abandonner les méthodes extractives pour une méthode abstractive. Cependant, le faible nombre de documents pouvant servir à l'apprentissage est très contraignant [1]. Nous faisons l'hypothèse qu'étant donnée la petite taille moyenne des résumés et leur faible variabilité linguistique, la partie décodeur d'un système encodeur/décodeur peut apprendre à générer un résumé en utilisant des éléments de style des résumés manuels. En revanche, la taille relativement importante des commentaires rend la tâche d'encodage plus compliquée, voire impossible eu égard à la taille réduite du corpus d'apprentissage.

Nous prenons donc le parti de réduire l'entropie des textes source afin de permettre à un encodeur/décodeur d'apprendre à générer des résumés avec un corpus de taille restreinte. L'idée la plus simple consiste à ne garder dans les commentaires en entrée que ceux qui sont jugés pertinents pour l'élaboration d'un résumé, avant d'apprendre un modèle de résumé automatique sur ces commentaires filtrés. Nous détaillons ici ces deux étapes.

1. Quand bien même nous disposions des résumés sur vingt ans, nous aurions 500 fois moins de documents que les corpus CNN/Dailymail traditionnellement utilisés dans le cadre du résumé abstractif neuronal.

4.1 Filtrage des phrases

Afin de filtrer les phrases en entrée, nous avons besoin de caractériser les phrases qui sont porteuses d'informations importantes, et celles qui ne le sont pas. Pour cela, nous avons décidé de nous fonder sur les résumés manuels, écrits par les commentateurs des matchs. Nous faisons l'hypothèse que ces résumés ne contiennent que des informations pertinentes.

4.1.1 Annotation manuelle du corpus

Nous avons analysé une année entière de Ligue 1 (donc 380 paires commentaires en direct / résumé) et typé les informations trouvées dans les résumés. La présence d'une information au sein d'un résumé témoigne de sa pertinence. Nous avons donc identifié les types d'information, puis compté les occurrences des différents types d'information et gardé les plus fréquents. Cela nous a amené à la liste de catégories d'information suivante : équipe réaliste, classement en championnat, qualité d'une mi-temps, blessure d'un joueur, expulsion d'un joueur, fin de série de victoires / défaites d'une équipe, but sur penalty, résultat du match, but / doublé / triplé d'un joueur, penalty raté, premier match d'un joueur pour sa nouvelle équipe, absence / retour d'un joueur, qualité du coaching, domination d'une équipe, match équilibré.

Nous avons alors systématiquement recherché ces informations dans les commentaires en direct et annoté les commentaires selon le type d'information qu'ils véhiculent. À titre d'exemple, si la figure 1 ne comporte pas de commentaire pertinent, le résumé du match présente des informations importantes : retour de Neymar, réalisme du PSG ("chirurgicaux, les Argentins..."), but d'Icardi, but de Di Maria, victoire du PSG (dérivé de la victoire à la mi-temps et de la gestion par la suite). Dans ce résumé comme dans beaucoup d'autres, des informations sont implicites et dérivées d'autres informations, ce qui a rendu la tâche de définition des types d'informations particulièrement complexe. Nous disposons ainsi d'un corpus annoté pour apprendre à typer les commentaires selon les informations qu'ils portent.

4.1.2 Catégorisation des commentaires

Nous avons gardé uniquement les 17 classes les plus fréquentes, considérant qu'en dessous d'un certain seuil – fixé empiriquement – la fréquence d'apparition d'un type d'information au sein des résumés était trop faible pour qu'il soit jugé important.

Avant de procéder à la classification de phrases, nous avons entraîné un modèle de langage Bengio et al. (2003) (Sundermeyer et al., 2015) sur le corpus de commentaires et de résumés uniquement afin de prendre en compte les spécificités de ce corpus particulier (vocabulaire spécifique, style différent de la langue générale). Ce modèle représenté à la figure 2 apprend des plongements de mots grâce à un réseau de neurones de type bi-LSTM qui vise à améliorer la prédiction de la probabilité du mot suivant une séquence.

Nous avons alors procédé à l'entraînement d'un modèle de classification binaire des commentaires sur un échantillon d'une année de commentaires annotés. Le modèle utilisé représenté à la figure 3 est un Bi-LSTM (deux couches de LSTM successives, une procédant du début à la fin de la phrase, l'autre de la fin au début de la phrase). Cette architecture bi-directionnelle permet d'obtenir de meilleurs résultats dans des tâches de traitement du langage. La couche d'entrée est un commentaire de match et la couche de sortie une valeur binaire. Le

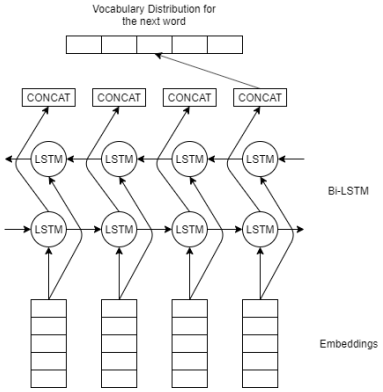


FIG. 2: Architecture du modèle de langage

commentaire est classé pertinent s'il couvre une des 17 classes annotées, autrement le commentaire est classé comme non pertinent.

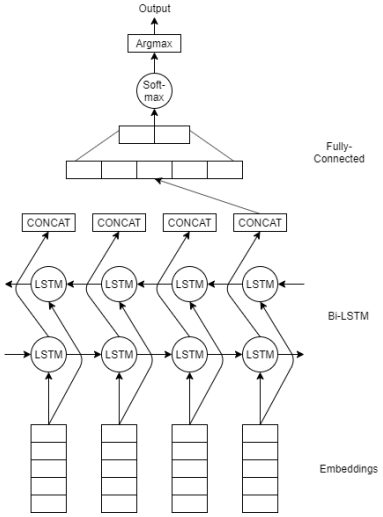


FIG. 3: Architecture du modèle de classification

Nous avons appliqué ce modèle sur l'ensemble des commentaires hors corpus d'apprentissage. Ainsi, nous pouvons filtrer les phrases des commentaires afin de présenter à un modèle de résumé neuronal uniquement les phrases jugées pertinentes, et ainsi améliorer la réponse du modèle grâce à la réduction de l'entropie des données en entrée. Les résultats de la classification binaire sont présentés en tableau 1.

Méthode	Rappel	Précision	F-mesure
Classification binaire	0.87	0.89	0.87

TAB. 1: Précision du modèle de classification

4.2 Modèle de résumé génératif

Nous avons utilisé un *pointer-generator network* (See et al., 2017) représenté à la figure 4. Nous l'avons entraîné sur deux jeux de données différents :

— Corpus brut ;
— Corpus filtré (classes binaires).

Le *pointer-generator* est une méthode d'apprentissage supervisé issue des modèles de traduction à partir de séquences et vers des séquences (Bahdanau et al., 2014) avec un mécanisme d'attention (Nallapati et al., 2016). Ce modèle a permis de grandes avancées dans le domaine de la traduction automatique et dans la génération de résumés.

L'architecture "séquence à séquence" est composée de deux parties : un encodeur et un décodeur, chacun composé de LSTMs bidirectionnels, c'est-à-dire que la séquence est traitée du début vers la fin et de la fin vers le début, pour avoir une meilleure représentation de la séquence. Le LSTM est un neurone qui prend une donnée en entrée et génère une sortie ainsi qu'un état qui sera transmis au neurone contigu de la même couche. Ainsi, l'information de temporalité est prise en compte dans ce type de neurone et se révèle très utile pour les données telles que les textes ou les séries temporelles.

De ce fait, l'état final généré par les chaînes de LSTM de l'encodeur est une représentation de la phrase entière. Cette représentation est ensuite donnée comme état initial pour le décodeur et est utilisée pour calculer les poids du mécanisme d'attention. Le mécanisme d'attention est un calcul de distribution qui va permettre d'évaluer, à chaque étape de génération quelles sont les parties de la phrase qui sont les plus pertinentes.

Ainsi, le décodeur génère une distribution de probabilités sur le vocabulaire du modèle en prenant en compte l'attention correspondant à son étape de génération.

Ce modèle offre de nombreux avantages mais souffre de plusieurs problèmes. Premièrement, ce modèle repose sur un vocabulaire prédéterminé et ne peut pas générer de mots en dehors de son vocabulaire, ce qui est très limitant pour les noms propres ou les mots qu'il n'a jamais vus. Deuxièmement, les résultats observés par ce modèle tendent à se répéter dans les données générées.

Afin de remédier à ces limitations, le modèle du *pointer-generator* introduit deux mécanismes supplémentaires par rapport aux modèles "séquence à séquence" avec Attention. Un premier mécanisme de pointage (pointer) permet d'inclure les mots hors du vocabulaire dans les calculs d'attention et de les copier à certaines étapes de génération si besoin. Un deuxième mécanisme de couverture permet quant à lui de prendre en compte les poids d'attention des itérations précédentes pour ne pas se concentrer plusieurs fois sur les mêmes passages et éviter de se répéter au moment de la génération de séquences. La distribution de probabilité finale est donc une combinaison entre la distribution de génération et la distribution de pointage.

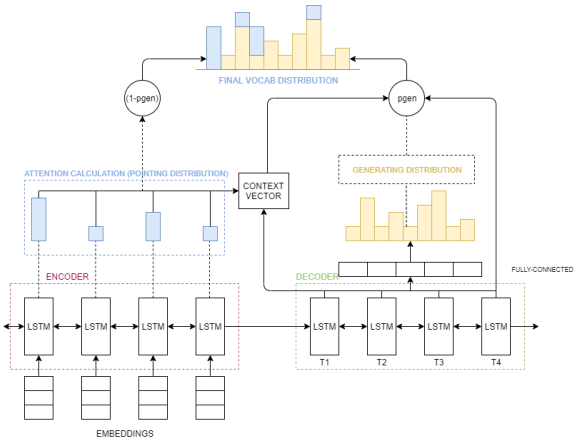


FIG. 4: Architecture du pointer-generator

5 Expériences

Pour tester notre approche, nous avons comparé deux modèles de résumé automatique : une méthode extractive, LexRank Radev (2004) ainsi qu'une méthode générative, le *pointer-generator* sur l'échantillon des 167 derniers matches de Ligue 1. Nous avons utilisé l'implémentation de LexRank réalisée par Nyzam et Bossard (2019) [2].

Les deux méthodes sont testées sans et avec filtrage préalable des phrases jugées pertinentes selon la méthode présentée en §4.1.2, ce afin de valider l'hypothèse selon laquelle la réduction de l'entropie dans les textes source a un effet positif sur la convergence du modèle et la qualité des résumés produits. Les chaînes de traitement des méthodes génératives sans et avec filtrage préalable sont présentées en figures 5 et 6.

Les résumés sont évalués avec ROUGE (Lin, 2004), d'après le score ROUGE-2, avec la configuration qui a montré la meilleure corrélation avec les résultats humains dans l'étude de Graham (2015).

Le score ROUGE-N est une métrique qui compare les N-grammes en communs entre le résumé de référence et les résumés à évaluer. Nous avons pris comme résumé de référence les résumés rédigés par le commentateur à la fin du match.

Le modèle de langage utilise une couche de plongements de mots de dimension 64. Les cellules récurrentes de l'encodeur et du décodeur sont de dimension 64. La couche de sortie a pour dimension la taille du vocabulaire, qui est de 4480.

Le modèle de classification utilise la même couche de plongements de mots qu'il récupère du modèle de langage après entraînement et deux couches de LSTM (bi-LSTM) chacun de dimension 16, la couche de sortie est de taille 2 (0 ou 1 pour importante et non importante).

Le modèle du *pointer-generator* utilise un encodeur et un décodeur avec des bi-LSTM de dimension 128. La taille du vocabulaire du modèle est de 50000. Durant l'entraînement, le modèle prend des textes tronqués à 400 mots et produit des résumés ne dépassant pas 100

2. MOTS : `https://github.com/ToolAutomaticSum/MOTS/`

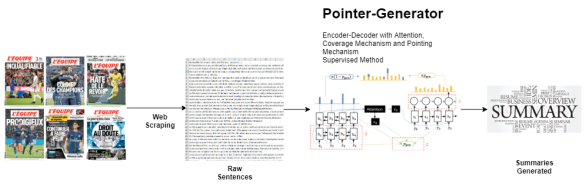


FIG. 5: Architecture de la pipeline sans classification

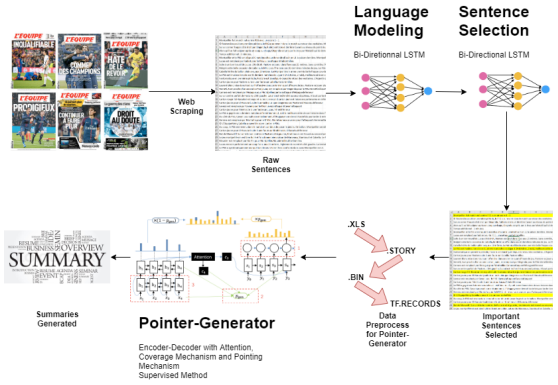


FIG. 6: Architecture de la pipeline avec classification

mots, ce qui est bien plus que le nombre de mots utilisés dans les résumés humains. Afin de réduire les dimensions du problème et accélérer l'entraînement, nous avons un batch de taille 4. La rétropropagation de l'erreur se fait avec l'optimiseur Adam avec un taux d'apprentissage de 0.15. L'apprentissage des modèles a été effectué sur 30000 itérations (80 epochs). Cela représente moins d'itérations que le modèle de See et al. (2017) pour obtenir des résultats car notre jeu d'apprentissage est beaucoup plus petit.

6 Résultats

Les résultats sont présentés dans le tableau 2. Nous observons une amélioration conséquente des scores ROUGE des modèles extractifs et génératifs lorsqu'ils sont lancés sur les commentaires filtrés.

Nous constatons également que les modèles extractifs sont meilleurs en rappel mais moins précis que les modèles génératifs. Le modèle extractif utilisé ici maximise en effet le nombre de mots dans les résumés, au contraire du modèle génératif. Ainsi, les résumés extractifs sont composés de 69 mots en moyenne contre 44 mots pour les résumés génératifs. Dans les tâches classiques de résumé par extraction, les résumés générés par différents systèmes ont à peu près la même taille en raison des stratégies d'extraction. La mesure de rappel est donc la plus

Dans un match nul à domicile pour Nantes qui s'enfonce au score logiquement, Lille se quittent pour la zone de relégation. Le FC Nantes s ' incline pour des Lillois courageux , mais se retrouve le doublé de tableau.

FIG. 7: Exemple de résumé généré par notre système génératif

Le FC Nantes et Lille se séparent sur un match nul après une fin de match haletante. Malgré un Sala décisif (doublé), les Canaris pourront nourrir des regrets après avoir mené deux fois au score. Nantes reprend sa 5e place devant Montpellier tandis que Lille sort de la zone rouge.

FIG. 8: Exemple de résumé manuel rédigé par le commentateur de L'Équipe

importante. Ici, en revanche, nous comparons des résumés de tailles différentes. La F-mesure est donc ici l'indicateur le plus pertinent pour évaluer la qualité des résumés.

Méthode	Rappel	Précision	F-mesure
Extractif	3.5	1.7	2.3
Extractif + classif	3.7	2.4	2.7
Génératif	2.5	3.4	2.9
Génératif + classif	2.9	4.1	**3.3**

TAB. 2: Scores ROUGE-2 des différents systèmes de résumé

On constate également que si les résumés génératifs présentent des informations pertinentes, celles-ci sont souvent mal exprimées. La figure 7 présente un exemple de résumé généré par notre système. Les informations suivantes sont communes entre ce résumé et le résumé manuel présenté en figure 8 :

— Match nul entre Nantes et Lille ;
— Lille quitte la zone de relégation ;
— Un doublé a été inscrit.

Les résumés extractifs comportent énormément de redondance et d'informations non pertinentes. Utiliser des indices de surface autres que des indices purement fréquentiels, à l'instar de Zhang et al. (2016), permettrait sûrement de résoudre en partie ce problème. La figure 9 présente un résumé extractif pour le même match que les résumés des figures 7 et 8.

7 Discussions

Notre modèle propose donc des informations pertinentes mais bruitées par une qualité linguistique approximative. Nous supposons que ce manque de cohérence linguistique est principalement dû au manque de données d'entraînement ; cela empêche le modèle de capter les spécificités de la langue malgré une variabilité linguistique faible due à la spécialisation des documents.

Sala signe un doublé ! Thomasson insiste à gauche dans la surface mais se heurte à un Lillois. Lucas Lima hérite du ballon et propose un centre du pied gauche. Totalement libre aux 6 mètres, Sala place sa tête et trompe Maignan sur la droite. Sala attaquant © L'Equipe Grâce à son buteur Sala (9e but en L1 cette saison), le FC Nantes mène très logiquement à la pause tant les Canaris ont été solides et efficaces. But de Sala ! Il centre pour Thomasson qui prolonge vers Sala. Sala attaquant

FIG. 9: Exemple de résumé extractif

Comme nous l'avions supposé, le modèle extractif propose des juxtapositions de commentaires très bruitées qui contiennent beaucoup d'informations non pertinentes. Les résultats sont donc très éloignés des résumés humains. Ce constat est montré d'une part à la lecture des résumés produits et d'autre part par l'évaluation du score ROUGE.

Le filtrage des phrases en amont de l'apprentissage permet bien d'améliorer la qualité des résumés générés. L'apprentissage est simplifié par la réduction du bruit et de la taille des données en entrée.

8 Conclusion et perspectives

Dans cet article nous avons présenté un modèle qui permet de générer des résumés abstractifs pour des résumés de documents de spécialité avec peu de données en langue française. Notre objectif a été de montrer que pour la génération de résumés dans des contextes spécifiques, on pouvait faire converger des modèles abstractifs plus rapidement en réduisant l'entropie des données d'entrée. En utilisant des réseaux de neurones, nous nous sommes affranchis du bruit des données et avons significativement amélioré les résultats des modèles de résumé extractif comme abstractif.

Nous avons constaté que beaucoup d'informations nécessaires à la génération manuelle des résumés ne sont pas présentes dans les commentaires en direct. En effet, beaucoup de faits importants : absence d'un joueur, réalisme, domination d'une équipe, match équilibré, sont souvent uniquement dérivables depuis des données chiffrées hors texte. Fournir systématiquement ces données en entrée à un système génératif peut l'aider à mieux générer les textes liés à leur interprétation. Ainsi, nous projetons d'ajouter aux séquences textuelles les statistiques pertinentes pour la génération des résumés.

Remerciements

Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-16-CE38-0008 (projet ANR JCJC ASADERA).

Références

Bahdanau, D., K. Cho, et Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. cite arxiv :1409.0473Comment : Accepted at ICLR 2015 as oral

presentation.

Bengio, Y., R. Ducharme, P. Vincent, et C. Janvin (2003). A neural probabilistic language model. *J. Mach. Learn. Res. 3*, 1137–1155.

Bouayad-Agha, N., G. Casamayor, S. Mille, et L. Wanner (2012). Perspective-oriented generation of football match summaries : Old tasks, new challenges. *ACM Trans. Speech Lang. Process. 9*(2), 3 :1–3 :31.

Corney, D., C. Martín Dancausa, et A. Goker (2014). Two sides to every story : Subjective event summarization of sports events using twitter. Volume 1198.

Graham, Y. (2015). Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 128–137.

Lin, C.-Y. (2004). ROUGE : A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain, pp. 74–81. Association for Computational Linguistics.

Nallapati, R., B. Xiang, et B. Zhou (2016). Sequence-to-sequence rnns for text summarization. *CoRR abs/1602.06023*.

Nyzam, V. et A. Bossard (2019). A modular tool for automatic summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3 : System Demonstrations*, pp. 189–194.

Radev, D. R. (2004). Lexrank : Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research 22*, 2004.

See, A., P. J. Liu, et C. D. Manning (2017). Get to the point : Summarization with pointer-generator networks. *CoRR abs/1704.04368*.

Sundermeyer, M., H. Ney, et R. Schlüter (2015). From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc. 23*(3), 517–529.

Zhang, J., J.-g. Yao, et X. Wan (2016). Towards constructing sports news from live text commentary. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, Berlin, Germany, pp. 1361–1371. Association for Computational Linguistics.

Summary

In this paper, we propose a method allowing to generate football games summaries based on live comments made by sports journalists. We show that this difficult task has proven hard to realize with extractive approaches. First, we propose a learning model relying on deep neuronal networks in order to select the most accurate sentences. In a second time, this reduction of noise in comments allows us to learn how to generate abstractive summaries using a pointer-generator neuronal network. We run an experimental study and show the benefit gained by selecting accurate sentences as well as the quality of the automatic summaries generated. We present some encouraging first results.

Extraction du contenu principal de pages web

Guillaume Bruneval*, Mohamed Lacarne*, Mohamed Kone*,
François-Xavier Bois*, Stanislas Morbieu*,**

*Kernix, 6 rue Lalande, 75104 Paris, France
lab@kernix.com
**LIPADE, Université de Paris, 75006 Paris, France
stanislas.morbieu@parisdescartes.fr

Résumé. L'extraction du contenu principal d'une page web constitue un enjeu majeur de la fouille de textes permettant de fournir du contenu moins bruité en entrée de méthodes d'analyses ou de prédiction. Nous souhaitons pouvoir extraire du contenu sur tout type de page y compris les pages présentant plusieurs zones de texte principal. Nous présentons une méthode d'apprentissage non supervisée permettant d'extraire le contenu textuel principal d'une page web. Celle-ci est constituée de trois étapes : une phase de classification non supervisée de blocs de texte au sein d'une même page, une phase de sélection des clusters associés au contenu principal, puis une phase d'apprentissage "supervisé" entraîné sur les données labellisées par les deux étapes précédentes. Des expériences sont menées pour valider la généralisation du classifieur et la qualité des résultats obtenus.

1 Contexte industriel

Les pages *web* constituent un large vivier de contenu disponible et facilement accessible. Cependant, le contenu principal se trouve mélangé à du contenu annexe tel que les menus de navigation, des encarts publicitaires et les pieds de page. Afin de caractériser le plus fidèlement la sémantique d'une page, enlever le bruit formé par les contenus annexes constitue une étape préalable en amont des méthodes de traitement du langage. Ceci se révèle aussi pertinent pour offrir une navigation apaisée à un visiteur en supprimant les distractions visuelles. Le HTML 5 a fourni aux développeurs la possibilité de définir la sémantique des différentes parties d'une page web *via* des balises telles que `<header>`, `<footer>`, `<nav>`, `<article>` ou `<aside>`. Cependant, ces balises sont peu utilisées en pratique [1]. Certaines heuristiques telles qu'implémentées dans Boilerpipe (Kohlschütter et al., 2010) [2] ou Newspaper3k [3] permettent de résoudre le problème d'extraction pour des pages de certains types, notamment les articles de presse mais se révèlent moins efficaces pour des pages de structures plus variées. De même, les méthodes d'apprentissage supervisé telle que (Vogels et al., 2018) s'avèrent

1. Sur un échantillon de 370 000 domaines étudiés, 33% ne contiennent aucune de ces balises.
2. `https://github.com/kohlschutter/boilerpipe`
3. `https://pypi.org/project/newspaper3k/`

efficaces sur les jeux de données classiques du type CleanEval (Baroni et al., 2008). L'apprentissage automatique offre des possibilités de procéder à l'extraction de manière plus robuste, cependant deux aspects importants sont à considérer : la capacité à entraîner un modèle à prix non prohibitif, impliquant donc des modèles non supervisés, ainsi que le faible temps d'exécution lors de la prédiction pour pouvoir passer à l'échelle. Pour résumer, nous avons cherché à mettre en place une méthode agile capable d'effectuer des extractions sur des types de page variés, dans un temps relativement court et à moindres coûts.

2 Travaux connexes

L'extraction de contenu est souvent effectuée à l'aide d'heuristiques qui s'appuient sur la proportion de balises HTML dans un bloc de texte (Weninger et al., 2010; Sun et al., 2011). L'idée sous-jacente est alors qu'un menu de navigation a tendance à avoir plus de liens (balise `<a>`) que le contenu principal, ou qu'il apparaît sous forme de liste (balises `<li>`).

Des travaux plus récents mettent en place des methodes de classification supervisée. C'est le cas de (Vogels et al., 2018) qui utilise un réseau de convolution au service des tâches d'extraction. Ces méthodes obtiennent généralement des résultats probants sur des jeux de données classiques tels que CleanEval. Cependant elles ne sont pas destinées à extraire le contenu sur tout type de pages. De plus elles nécessitent un réentraînement dès lors que l'on change de contexte. Son utilisation impliquerait donc que nous annotions un jeu de données. Or c'est de cette étape de labellisation manuelle, dont, dans notre contexte particulier, nous souhaiterions nous abstenir afin de minimiser le temps et les coûts.

Nous nous sommes donc naturellement dirigés vers les méthodes automatiques, c'est-à-dire : soit celles qui sont non supervisées, soit celles qui proposent une labellisation automatique avant un apprentissage. Zhou et Mashuq (2014) proposent une méthode en trois étapes : classification non supervisée de blocs de textes, sélection du cluster correspondant au contenu principal pour labelliser des blocs de textes et entraînement d'un classifieur supervisé pour prédire sur de nouvelles pages. La première étape nécessite de parcourir plusieurs pages de *même nature* d'un même site ce qui rend son application difficile. De plus, l'utilisation de DBSCAN (Ester et al., 1996) nécessite un choix judicieux des hyperparamètres. La seconde phase utilise une similarité textuelle entre la meta description et le texte des blocs, ce qui se révèle être peu performant en dehors des sites de presse qui utilisent le chapeau comme meta description. Nous utilisons le même cadre en trois étapes mais en modifions les composantes. Miao et al. (2009) effectuent une classification non supervisée des blocs HTML afin de regrouper des blocs de même nature. Nous nous inspirons de ces travaux pour la première phase de notre méthode.

3 Extraction du contenu principal

Une page web est constituée de blocs HTML qui peuvent être représentés par :

— Le DOM (arbre des balises HTML) : chaque bloc de texte correspond à un chemin du DOM. Nous donnons tout d'abord de ce chemin une représentation matricielle : chaque ligne correspond au nom de la balise HTML et chaque colonne correspond à sa position dans le chemin. Nous pondérons par une harmonique en prenant comme dénominateur

la position des balises dans le chemin (tableau 1). Cette matrice est ensuite lissée de manière à obtenir une représentation vectorielle d'un chemin et donc d'un bloc (figure 1).

— Les propriétés CSS qui définissent comment le bloc apparaît visuellement : les coordonnées des sommets du rectangle représentant le bloc, la taille de la police de caractère, l'alignement du texte, l'épaisseur du trait, etc.

— Le contenu textuel : sous forme de chaîne de caractères ou comme ensemble de mots. Nous utilisons dans la suite la représentation vectorielle où les dimensions correspondent aux mots du vocabulaire et les éléments sont le nombre d'occurrences du mot dans le bloc.

	Position			
	1	2	3	4
	1	0	0	$\frac{1}{4}$
<p>	0	0	$\frac{1}{3}$	0
	0	$\frac{1}{2}$	0	0
<h1>	0	0	0	0

TAB. 1: Représentation matricielle du chemin /<ul>/<p>/

$$v = \left(1, 0, 0, \tfrac{1}{4}, 0, 0, \tfrac{1}{3}, 0, 0, \tfrac{1}{2}, 0, 0, 0, 0, 0, 0\right)$$

FIG. 1: Représentation vectorielle du chemin /<ul>/<p>/

La méthode proposée est constituée de trois étapes :

1. Pour une page donnée, une classification non supervisée produit un ensemble de clusters de blocs de textes. Pour cette étape, nous utilisons une classification spectrale avec un noyau Gaussien sur des graphes produits par une distance basée sur les chemins de balises HTML : Soit deux blocs i et j, l'ensemble $\mathcal{B}$ des balises HTML et n la longueur de l'arbre DOM. Soit δ la fonction indicatrice telle que $\delta_{p,b}^{(i)} = 1$ si la balise b est à la position p dans l'arbre DOM pour le bloc i, et 0 sinon. On définit la distance d par :

$$d(i,j) = \sqrt{\sum_{p=1}^{n} \sum_{b \in \mathcal{B}} \frac{1}{p} \left(\delta_{p,b}^{(i)} - \delta_{p,b}^{(j)}\right)^2}$$

2. Une fois l'étape de clustering effectuée avec six clusters, nous avons constaté que notre algorithme parvenait à bien distinguer le contenu du bruit. Notre problème était que parfois le contenu pertinent était rassemblé dans un seul cluster, ce qui est typiquement le cas sur les pages présentant un unique bloc de texte homogène, mais il arrivait également qu'il soit séparé en deux ou trois clusters. La question du bon choix des clusters était donc centrale. Nous avons donc décidé d'effectuer la sélection des clusters correspondant au contenu principal selon un score lié à la proportion de mots courants (*stop-words*[4]). Pour chaque cluster, nous calculons donc le nombre de *stop-words* contenus

4. La liste des *stop-words* est fournie par le package Python NLTK

dans l'ensemble des blocs du cluster et le divisons par le nombre total de *stop-words* contenus dans la page. Pour une page donnée, nous avons alors un vecteur de six scores. Nous utilisons ensuite deux heuristiques : Si l'écart type de ces six scores est supérieur à 0.2, on ne sélectionne que le cluster de score le plus élevé. Sinon, on applique la deuxième heuristique : il s'agit de déterminer un "saut" dans notre vecteur de score trié dans l'ordre décroissant. On considère que l'on a affaire à un saut lorsque la variation de score entre deux clusters est inférieure de 50% à la variation de score précédente. Une fois ce saut identifié, on sélectionne les clusters situés avant ce "saut". On considère que les blocs contenus dans ces clusters contiennent le texte principal de la page.

3. Un algorithme de classification supervisée est ensuite entraîné avec les données précédemment labellisées pour prédire si un bloc de texte est pertinent ou non. Nous utilisons les trois représentations de blocs précédentes et un algorithme adapté à chacune des différentes représentations DOM, CSS et textuelle. Pour le DOM nous effectuons une régression logistique. Pour les propriétés CSS, nous avons fait le choix d'un gradient boosting. Les données ont été centrées et réduites. Enfin, pous les données textuelles, nous avons opté pour une classification naïve bayésienne multinomiale. Un consensus par vote au suffrage universel de ces trois méthodes forme le modèle de classification afin d'être plus robuste et de palier des chutes de la mesure de rappel sur certaines pages.

4 Expériences

Pour valider l'intérêt qualitatif de notre méthode, nous considérons trois jeux de données (tableau 2) :

A. Un échantillon pour labelliser les blocs de textes (phase 1 et 2) de manière non supervisée. Les pages proviennent de sites d'institutions culturelles ;

B. Un échantillon pour évaluer la classification des blocs de texte (phase 3) sur des données de même nature, les pages provenant aussi de sites d'institutions culturelles ;

C. Un échantillon pour évaluer la classification des blocs (phase 3) sur des données de nature différente, les pages provenant de sites d'entreprises.

Données	Type de pages	#pages	#domaines	#blocs
A	Institution culturelle	3 006	107	134 535
B	Institution culturelle	106	31	9 213
C	Entreprise	30	29	2 928

TAB. 2: Caractéristiques des jeux de données

Pour l'évaluation nous utilisons la précision, le rappel et la mesure F1, où les individus considérés sont les mots et nous comparons à Boilerpipe et Newspaper3k qui implémentent des séries de règles.

— La précision est le nombre de mots pertinents retrouvés par rapport au nombre de mots retournés comme contenu principal par le classifieur.

— Le rappel est le nombre de mots pertinents retrouvés par rapport au nombre de mots du contenu principal annoté manuellement.

Les résultats sont reportés dans le tableau 3. Bien qu'étant entraîné sur un jeu de données spécialisées (A), le modèle de classification "supervisée" s'est révélé être efficace aussi bien sur des pages de même nature (B) que différente (C).

Nous avons choisi de comparer notre travail à Newspaper et Boilerpipe parce que ce sont deux méthodes largement utilisées. Nous avons également effectué une comparaison avec (Zhou et Mashuq, 2014) mais les résultats n'étaient pas satisfaisants sur notre jeu de données, premièrement parce que certains domaines comportaient trop peu de pages, deuxièmement parce que la meta-description pouvait être inexistante ou trop réduite. Boilerpipe utilise, comme nous le faisons, une grande diversité de caractéristiques, mais le fait que nous exploitions des méthodes d'apprentissage dépendant de paramètres, plutôt que des heuristiques dépendant de seuils à valeur fixe nous permet de gagner en robustesse. Newspaper s'est révélé peu adapté aux pages de nos jeux de données. On peut avancer deux raisons à cela. Premièrement, Newspaper fonctionne également à l'aide d'heuristiques, mais à la différence de Boilerpipe, cette méthode exploite moins de caractéristiques ce qui peut la faire échouer lorsqu'elle doit opérer sur des pages de structures variées.

	Données B			Données C		
Méthode	Précision	Rappel	F1	Précision	Rappel	F1
Notre méthode	72,6	**85,3**	**78,4**	74,8	**87,4**	**80,6**
Boilerpipe	**73,1**	69,9	71,5	**81,1**	73,2	76,9
Newspaper3k	55,4	45,4	49,9	84,8	54,9	66,7

TAB. 3: Résultats

5 Conclusion et perspectives

Bien que nécessitant une étude plus approfondie sur des jeux de données variés, les résultats sont prometteurs : la phase de classification non supervisée permet de labelliser à moindre coût des volumes importants et la partie "supervisée" permet de passer à l'échelle tout en généralisant à des types de pages différents. Pour chaque étape de notre méthode, plusieurs choix sont possibles (représentation des blocs de textes, algorithmes de classification, association des clusters correspondant au contenu pertinent, etc.). Nous envisageons donc une étude plus approfondie pour associer des typologies de pages se prêtant bien aux différentes variantes. Celle-ci pourrait être détectée en amont afin d'exécuter la méthode fonctionnant le mieux. Nous explorerons aussi des méthodes de consensus pour combiner les variantes. Il semblerait également pertinent de trouver des heuristiques plus claires et plus robustes pour sélectionner le cluster contenant le texte principal. Par ailleurs, les propriétés CSS utilisées dans la partie supervisée ont été sélectionnées sur des bases intuitives. Il serait cependant intéressant d'effectuer un travail d'analyse exploratoire afin d'exploiter les variables les plus pertinentes parmi les quelques trois cents propriétés CSS recensées.

Références

Baroni, M., F. Chantree, A. Kilgarriff, et S. Sharoff (2008). Cleaneval : a competition for cleaning web pages. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ester, M., H.-P. Kriegel, J. Sander, X. Xu, et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, Volume 96, pp. 226–231.

Kohlschütter, C., P. Fankhauser, et W. Nejdl (2010). Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, New York, NY, USA, pp. 441–450. Association for Computing Machinery.

Miao, G., J. Tatemura, W.-P. Hsiung, A. Sawires, et L. E. Moser (2009). Extracting data records from the web using tag path clustering. In *Proceedings of the 18th international conference on World wide web*, pp. 981–990. ACM.

Sun, F., D. Song, et L. Liao (2011). Dom based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 245–254. ACM.

Vogels, T., O. Ganea, et C. Eickhoff (2018). Web2text : Deep structured boilerplate removal. *CoRR abs/1801.02607*.

Weninger, T., W. H. Hsu, et J. Han (2010). Cetr : content extraction via tag ratios. In *Proceedings of the 19th international conference on World wide web*, pp. 971–980. ACM.

Zhou, Z. et M. Mashuq (2014). Web content extraction through machine learning.

Summary

The extraction of the main content of a web page is a major issue in text mining. It provides less noisy input content to analysis or prediction methods. We present an unsupervised learning method to extract the main textual content of a web page. It relies on three stages: a clustering phase of text blocks within a single page, a phase of selection of the clusters associated with the main content, and a "supervised" learning phase carried out on the data labeled by the two previous steps. Experiments are conducted to validate the generalization of the classifier and the quality of the obtained results.

Classification de phrases courtes : des approches non-supervisées aux approches faiblement supervisées

Illustration sur une problématique industrielle

Kaoutar Ghazi*, Sébastien Marchal*, Andon Tchechmedjiev*
Pierre-Antoine Jean*, Nicolas Sutton-Charani*, Sébastien Harispe*

*LGI2P, IMT Mines Alès, Univ. Montpellier, Alès, France
prénom.nom@mines-ales.fr

Résumé. Cette note présente une étude, menée dans un contexte industriel, de différentes approches de classification non supervisée ou faiblement supervisée de courtes requêtes exprimées en langage naturel. Nous présentons et comparons différentes approches basées à la fois sur des techniques de Recherche d'Information et d'apprentissage machine exploitant différents types de plongements sémantiques (*embeddings*). Nous discutons les résultats obtenus avant de proposer différentes alternatives à base d'apprentissage automatique supervisé ne nécessitant que peu de données labelisées – des approches de type *few shot learning*. Cette note vise ainsi à faire une synthèse des approches dites état de l'art qui peuvent être utilisées pour traiter cette problématique de classification dans un contexte fréquemment rencontrée dans l'industrie où peu ou pas de données d'apprentissage labélisées sont disponibles (e.g., chatbot).

1 Introduction

De nombreuses problématiques de Traitement Automatique du Langage Naturel ($TALN$) qui sont d'intérêt pour l'industrie concernent la classification de textes courts, e.g. ChatBots, analyse de tweets ou d'avis client lors d'analyses d'opinions. Cette note se concentre sur le contexte que nous avons rencontré en pratique, potentiellement fréquent dans l'industrie, dans lequel peu ou pas de données labélisées sont disponibles pour l'apprentissage de modèles de classification, i.e. seules des descriptions des classes sous forme de textes courts, et éventuellement quelques exemples labelisées sont fournis.

Des modèles classiques de Recherche d'Information, aux modèles à base d'apprentissage machine, nous présentons dans cette note une synthèse de différentes approches éprouvées de l'état de l'art et outillées, qui peuvent aujourd'hui être facilement mises en oeuvre pour aborder ce type de problématique de classification, notamment dans l'industrie. Nous illustrons plus particulièrement l'utilisation de ces approches dans un contexte correspondant à la classification de requêtes exprimées sous la forme de textes courts (une à deux phrases), dans lequel nous disposons *a priori* seulement d'un court descriptif en langage naturel pour chaque classe. Nous distinguons par la suite deux cas : (i) *cold-start* – pas de donnée labélisée, (ii) *few-shot* – peu de données labélisées.

Il est en effet fréquent dans l'industrie d'être confronté au cas où : (a) seuls des descriptions des classes sous forme de textes courts et éventuellement et leurs descriptions sont fournies dans un premier temps avec peu ou pas de données labélisées, (b) une labélisation des données est éventuellement effectuée suite au déploiement en production du modèle initial, e.g. en analysant les retours du système, une fois déployé, suite au traitement de certaines données (i.e. validation de la décision ou non). On se retrouve alors ensuite dans un contexte d'apprentissage dit en ligne (*online learning*). Cette note se concentre donc sur la phase (a) précitée et est structurée comme suit.

Nous présentons dans un premier temps les approches étudiées pour élaborer un système de classification dans les contextes non-supervisés (problématique *cold start*) et faiblement supervisés. L'objectif n'est pas ici d'être exhaustif, mais plutôt de discuter des approches aujourd'hui simples à mettre en oeuvre – a minima dans un contexte R&D. Nous présentons et discutons par la suite les résultats obtenus sur un jeu de données spécifique. Des perspectives pour la mise en place de systèmes plus raffinés sont par la suite proposées.

2 Approches étudiées

La problématique que nous traitons dans cette note correspond à la classification de textes courts (requêtes), étant donné un descriptif textuel de chacune des classes. Nous présentons ci-après les approches considérées pour traiter cette problématique.

2.1 Approches de type Recherche d'Information (RI)

Les approches ici étudiées pour la classification de requêtes, sans données labélisées, se basent sur des méthodes éprouvées en Recherche d'Information, notamment celles qui reposent sur des pondérations des termes de manière à tenir compte de leur importance thématique dans un document.

L'importance d'un terme est généralement mesurée par rapport à la fréquence de ses occurrences ou co-occurences dans un contexte donné (fenêtre glissante, une phrase, un paragraphe, un document), qu'il s'agisse du nombre de ses occurrences dans un document, de son TF-IDF [1] par rapport à un document, du nombre de ses co-ocurrences dans l'ensemble des documents, ou de son PPMI (*Positive Pointwise Mutual Information*) [2] avec un autre terme dans le corpus.

Matrice terme-document (TF-IDF)

Analyser l'importance des termes décrivant une classe peut être utile pour notre problème de classification d'une requête. Comme nous procédons par une classification non-supervisée, nous proposons de calculer la matrice terme-document pour chaque requête telle que l'ensemble des documents correspondra aux descriptifs des classes plus la requête. Les éléments

1. $tf-idf(m,d) = tf(m,d) * idf(m)$, avec $tf(m,d)$ la fréquence du mot m dans le document d et $idf(m) = \log\left(\frac{n}{df(m)}\right) + 1$ où n représente le nombre total de documents et $df(m)$ le nombre de documents contenant le terme m.

2. $PPMI(m_1, m_2) = max(log_2(\frac{P(m_1,m_2)}{P(m_1)P(m_2)}, 0)$ avec $P(m_1, m_2)$ est la probabilité d'observer les mots m_1 et m_2 ensemble et $P(m_1)$ la probabilité d'observer m_1 indépendemment de m_2, toutes ces probabilités étant estimées en pratique par leur fréquences dans un document.

de la matrice sont le TF-IDF d'un mot dans un document. Nous obtenons par la suite une représentation des classes et de la requête (vecteurs colonnes de la matrice). Enfin, nous déduisons la classe d'appartenance de la requête en mesurant la similarité (euclidienne, cosinus ou corrélation) entre sa représentation et celle de chacune des classes et en gardant la classe la plus proche selon la distance considérée. Pour illustrer cette approche, nous considérons m classes, $C_1 \dots C_m$, et une requête R à classifier. Nous calculons alors la matrice terme-document sur le vocabulaire constitué des mots contenus dans l'ensemble formé par les descriptifs des m classes et une requête R (voir le tableau ci-dessous). De cette manière nous obtenons une représentation des classes et de la requête par un vecteur de nombres réels $[tf\_idf_{1,i}, \dots, tf\_idf_{n,i}]$ pour i allant de 1 à $m+1$ ($i = m+1$ pour la requête R). La classe C_k de R est la plus proche de R dans l'espace formé par cette représentation, i.e.

$$C_k = \underset{i=1,\dots,m}{\operatorname{argmin}} d\Big([tf\_idf_{1,i}, \dots, tf\_idf_{n,i}], [tf\_idf_{1,R}, \dots, tf\_idf_{n,R}]\Big)$$

.

Vocabulaire	C_1	C_2	C_3	...	C_m	R
m_1	tf_idf$_{1,1}$	...	...	...	tf_idf$_{1,m}$	tf_idf$_{1,R}$
m_2	...	...	...	...	...	...
m_3	...	...	...	...	...	...
m_4	...	...	...	...	...	...
m_5	...	...	...	...	...	...
...	...	...	...	...	...	...
m_n	tf_idf$_{n,1}$	...	...	...	tf_idf$_{n,m}$	tf_idf$_{n,R}$

Matrice terme-terme (PPMI)

Suivant le même procédé, nous proposons de calculer la matrice terme-terme pour chaque requête telle que les éléments de la matrice correspondent aux PPMI de deux mots dans le corpus constitué des descriptifs des classes et de la requête. Nous calculons ensuite une représentation des classes et de la requête par une agrégation (souvent une moyenne, mais il existe d'autres techniques état de l'art de reprojection [3]) des vecteurs représentant leur mots (vecteurs lignes de la matrice). Enfin, la classe d'appartenance attribuée à la requête est celle possédant la représentation vectorielle la plus proche (en fonction de la distance considérée) de la requête en question.

2.2 Approches par plongement de mots

Au-delà des approches distributionnelles mentionnées ci-avant (e.g. représentation de mots à base de PPMI), nous avons opté pour des représentations sémantiques plus génériques des mots issues de plongements pré-entraînés (plongements classiques, ou modèles de langue contextualisés) : les plongements CBOW (resp. SkipGram) qui sont appris par l'architecture neuronale de word2vec et qui permettent de prédire un mot sachant son contexte (resp. prédire le contexte étant donné un mot), puis les plongements FastText qui suivent la même procédure

3. https ://github.com/zalandoresearch/flair

que SkipGram, mais avec des n-grammes au lieu de tokens. Un n-gramme est une séquence de n caractères consécutifs d'un mot, e.g. un mot à 4 lettres est composé de trois 2-grammes.

Dans cette note, nous considérons des plongements pré-calculés disponibles en ligne [4] pour la langue Française : 1/ plongements CBOW appris sur le corpus WaC et de dimension 500 (taille du vecteur de plongement), 2/ plongements SkipGram appris sur le corpus WaC et de dimension 500, 3/ plongements CBOW appris sur le corpus Wikipédia et de dimension 700, 4/ plongements SkipGram appris sur le corpus Wikipédia et de dimension 1000.

De plus, nous générons des plongements FastText à l'aide de la libraire Flair Akbik et al. (2019), qui sont appris sur Wikipédia et de dimension 300.

Cette approche à base de plongements de mots, nous permettra de réaliser une classification par rapprochement sémantique d'une requête aux descriptifs des classes.

2.3 Approches à base d'apprentissage supervisé

En nous basant sur la librairie Flair, nous avons également mis en place une architecture de classification de textes supervisée, avec les plongements contextualisés les plus récents pour la langue Française – CamemBert Martin et al. (2019) – en entrée, qui alimentent un encodeur (vecteurs de documents) à base d'un réseau profond récurrent (biGRU), qui lui même alimente une couche de classification linéaire. Nous explorons les deux stratégies : cold start, en prenant les descriptifs des classes sur-échantillonées (209 $\times$ nombre de classes) comme un jeu d'entraînement ; *few-shot learning*, en considérant, dans un premier temps, 90% des requêtes dont nous disposons pour les évaluations comme jeu d'entraînement (les 10% destinés aux tests), puis dans un second temps, en combinant les descriptifs des classes avec des données labélisées (90% des requêtes extraites du jeu de données de test) pour constituer la base d'apprentissage. Dans ce dernier cas, nous évaluons les performances du système par une validation croisée (10 divisions) que nous détaillons dans la suite.

3 Résultats

3.1 Description des données

Nous avons testé les approches décrites auparavant sur un un jeu de données contenant les motifs de visite d'une administration associés aux 18 services aptes à traiter les besoins identifiés par ces motifs (e.g. Conseils budgétaires). Cette administration comprend 18 services différents qui sont ici considérés comme les classes à associer aux motifs de visite. La plupart des classes possèdent un descriptif, e.g. « Le crédit municipal de Paris vous aide à analyser vos dépenses et recettes, vous renseigne sur les différentes aides existantes » . Les requêtes sont exprimées en language quotidien e.g. « Je débute dans l'entreprenariat j'aurais besoin dans suivi budgétaire » et sont au nombre de 180 (10 requêtes par classe).

3.2 Évaluations

Nous évaluons la performance des approches considérées dans cette note par la précision qui correspond au nombre de requêtes que le système a réussi à correctement classifier (i.e. le

4. https ://fauconnier.github.io/data

bon service est associé au motif de visite en question) sur le nombre total de requêtes (taille du jeu de données de test).

Pour l'approche à base d'apprentissage supervisé, une validation croisée à 10 couches est réalisée. Elle consiste en la subdivision des données en 10 sous-ensembles comprenant les mêmes proportions de classe, une évaluation est ensuite effectuée sur chaque couche après apprentissage sur les 9 autres, la performance finale correspond à la moyenne des 10 performances ainsi calculées.

3.3 Analyse

Les résultats de classification obtenus pour les différentes approches sur notre jeu de données montrent que le mode *cold start* demeure une solution non négligeable pour une classification de textes courts, étant donné les descriptifs des classes et en l'absence de données labélisées. La meilleure performance est obtenue par les plongements CBOW entraînés sur le corpus Wikipédia en utilisant la correlation (ou même le cosinus) comme mesure de similarité.

Les plongements FastText étant de dimension réduite par rapport à ceux de CBOW ou SkipGram (300 vs plus de 500), il n'est pas étonnant d'obtenir de moins bon résultats avec FastText qu'avec CBOW ou SkipGram. Même si les deux techniques CBOW et SkipGram sont relativement similaires, il s'avère que CBOW est plus adaptée à notre jeux de données (Voir CBOW$_{WaC}$ vs SkipGram$_{WaC}$). En comparant les résultats de CBOW$_{WaC}$ et CBOW$_{Wiki}$ (resp. SkipGram$_{WaC}$ et SkipGram$_{Wiki}$), nous observons l'impact de la nature du corpus d'apprentissage des plongements de mots sur les résultats obtenus. Dans les deux cas le corpus Wikipédia semble plus adéquat pour capturer le sens des termes de notre vocabulaire métier que le corpus WaC dont le contenu correspond à des articles tirés du journal Monde Diplomatique. Nous observons que la technique TF-IDF est compétitive, dans notre contexte, que les autres méthodes ici considérées.

Pour les représentations calculées avec $PPMI$, malgré le fait qu'elles capturent le sens des termes dans leur contexte métier (corpus = Descriptifs des classes + requête) de manière plus spécifique que l'approche Word2Vec qui utilise des contextes plus génériques, les faibles résultats obtenus semblent indiquer que ce type de représentation n'est pas adapté à notre contexte d'apprentissage. Nous pouvons justifier ceci par le fait que les requêtes ont des représentations $PPMI$ sous forme de vecteurs creux (les termes d'une requête co-occurent rarement avec les termes des descriptifs des classes) ce qui n'est pas le cas pour les vecteurs représentants les classes, et peut donc entraîner des erreurs de classification. On peut remarquer qu'avec des représentations TF-IDF, nous obtenons des représentations par des vecteurs creux pour les requêtes mais aussi pour les classes grâce à la pénalisation des termes fréquents.

Distance	TF-IDF	PPMI	FastText	CBOW		SkipGram	
				WaC	Wiki	WaC	Wiki
Euclidienne	44%	5%	31%	32%	26%	27%	27%
Cosinus	44%	5%	27%	44%	55%	39%	37%
Correlation	46%	5%	31%	44%	**56**%	40%	37%

TAB. 1 – *Les résultats des différentes approches sans données de supervision*

Nous observons sur le Tab. 2 que l'approche basée sur une classification supervisée par les descriptifs des classes donne des résultats relativement similaires à ceux obtenus en suivant une approche non-supervisée. Cependant, avec peu de données annotées, nous avons pu doubler les performances du classifieur.

Corpus	Précision
Descriptifs des classes	39%
180 requêtes	95%
Descriptifs des classes + 162 requêtes	95%

TAB. 2 – *Les résultats de l'approche faiblement supervisée*

Finalement, le classifieur Flair semble ne pas être aussi pertinent que les meilleures approches précitées, mais en procédant par une approche de type *few-shot learning*, ou suivant une stratégie d'augmentation des données par les descriptifs des classes, nous arrivons à améliorer considérablement les résultats de la classification. Notons que nous avons appliqué une phase de pré-traitement préalable (lemmatisation et suppression des mots vides et des ponctuations) sur nos données textuelles. La suppression des mots vides n'est pas toujours une bonne solution pour les problèmes de classification car pouvant augmenter le risque de mauvaise classification, comme illustré par l'exemple suivant : « je viens récupérer le document ~~que~~ j'ai déposé » correspond à un retrait et non à un dépôt. Cependant, les problématiques pouvant être associées à la suppression des mots vides n'étant pas mis en évidence sur notre jeu de données, nous constatons ainsi que les meilleurs résultats que nous avons obtenus coïncident avec ceux sur lesquels nous avons réalisé une phase de pré-traitement complète.

4 Élargissement

Les résultats de classification dans notre contexte ne dépassent pas 60% avec les approches non supervisées basées sur l'utilisation de techniques de recherche d'information ou de plongement de mots. Les résultats que nous avons obtenus en testant une autre méthode de pondération, le « BM25 » développé par Robertson et Sparck Jones, sont aussi de même ordre de grandeur que le TF-IDF. Nous pensons qu'en combinant la méthode de pondération avec les plongements de mots Bao et al. (2019) nous pourrions améliorer cette approche. Une des conclusions principales de cette étude est la suivante : les performances peuvent être considérablement améliorées en tenant compte de données labélisées. Par conséquent, une approche de type *few-shot learning* suivant une stratégie d'augmentation des données par des réseaux de neurones type « Siamois » ou autres est largement recommandée pour traiter des problématiques semblables. Il est également possible d'utiliser des générateurs de phrases, e.g. GPT-2 à partir des mots-clés caractérisant les classes. Des approches dîtes N-way K-shot peuvent aussi être appliquées ; elles consistent à entraîner un modèle épisodiquement tel qu'à chaque épisode, l'entraînement est réalisé sur un sous-ensemble des données d'entraînement tiré aléatoirement tout en exploitant les données restantes pour calculer une représentation des classes, Snell et al. (2017), Vinyals et al. (2016).

Des approches d'augmentation basées sur la recherche de classement par ordre de pertinence entre les requêtes sont aussi exploitables, e.g. Triantafillou et al. (2017). On peut égale-

ment imaginer un processus d'augmentation de données basé sur le principe d'une classification hiérarchique, en rajoutant des meta-classes en tenant éventuellement compte des proximités entre classes. Dans le contexte de l'administration que nous avons considéré, on peut ainsi supposer qu'une même demande peut éventuellement être traitée par différents services. Enfin, des approches récentes proposent de tirer parti d'approches d'*adversarial learning* comme stratégie d'augmentation des données pour le *few-shot learning*, Zakharov et al. (2019).

5 Conclusion

Les problématiques de classification de textes courts sans données labélisées sont fréquentes dans le contexte industriel. Bien que des approches éprouvées de l'état de l'art permettent d'aborder la tâche en contexte non supervisé (*cold start*), notamment à l'aide d'approches à base de TF-IDF ou de techniques de plongements plus récentes, cette note souligne la pertinence d'étudier la considération de contextes faiblement supervisés peu contraignants en pratique (des dizaines de données labélisées peuvent suffire dans certains cas). Le passage vers des approches faiblement supervisées récentes (*few-shot* learning) puis, si possible, des approches d'apprentissage supervisé (en ligne) permet en effet très souvent de garantir une nette augmentation de la qualité des systèmes de classification.

Références

Akbik, A., T. Bergmann, D. Blythe, K. Rasul, S. Schweter, et R. Vollgraf (2019). FLAIR : An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, Minneapolis, Minnesota, pp. 54–59. Association for Computational Linguistics.

Bao, Y., M. Wu, S. Chang, et R. Barzilay (2019). Few-shot text classification with distributional signatures. *arXiv preprint arXiv :1908.06039*.

Martin, L., B. Muller, P. J. Ortiz Suárez, Y. Dupont, L. Romary, É. Villemonte de la Clergerie, D. Seddah, et B. Sagot (2019). CamemBERT : a Tasty French Language Model. *arXiv e-prints*, arXiv :1911.03894.

Snell, J., K. Swersky, et R. Zemel (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087.

Triantafillou, E., R. Zemel, et R. Urtasun (2017). Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pp. 2255–2265.

Vinyals, O., C. Blundell, T. Lillicrap, D. Wierstra, et al. (2016). Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638.

Zakharov, E., A. Shysheya, E. Burkov, et V. Lempitsky (2019). Few-shot adversarial learning of realistic neural talking head models. *arXiv preprint arXiv :1905.08233*.

Classification de phrases courtes en contexte non ou faiblement supervisé

Summary

In this note, we discuss different approaches of classification, without any supervision, of queries expressed on natural language into few classes given a textual description of each. We present then we compare approaches based on Information Retrieval as well as those based on word embeddings. Given few data of supervision, we show how can we process following a few-shot learning approach for further improvement of our classification using few annotated data then by exploring the data augmentation strategy with classes descriptions.

Extraction automatique de noms d'entreprises à partir de titres de presse : un exemple d'application chez ReportLinker

Marilyne Latour*, Jocelyn Bernard*, Corentin Regal*

* 21, quai Antoine Riboud - 69002 Lyon, France
mla/jbe/cre@reportlinker.com,
https://www.reportlinker.com/

Résumé. Cet article présente un retour d'expérience sur le processus de reconnaissance d'Entités Nommées (REN). L'expérience consiste à traiter des données non structurées à partir de dépêches d'actualité. Notre objectif est d'extraire automatiquement les noms d'entreprises contenues dans des titres d'articles de presse économique. L'article décrit les expérimentations effectuées aboutissant à une meilleure détection des titres de 27,4% et tire les premières conclusions de cette méthode.

1 Introduction

La reconnaissance d'entités nommées (REN) est une des tâches principales du traitement automatique de la langue naturelle (TALN). Elle concerne la reconnaissance et la désambiguïsation d'entités nommées (EN). C'est un traitement précieux pour de nombreuses applications linguistiques notamment dans des situations de traductions, de réponse automatique ou de création de résumé.

Nous proposons dans cet article d'utiliser des méthodes REN afin de détecter et d'extraire automatiquement les noms d'entreprises contenus dans les titres d'articles de presse en langue anglaise. Notre objectif est de pouvoir utiliser les EN extraites afin d'enrichir les bases de données pour notre moteur de recherche, ReportLinker. Notre approche repose sur un repérage d'une classe de verbes de transactions à partir d'une analyse par Stanford CoreNLP (Manning et al. (2014)) que nous avons amélioré via l'ajout de patrons lexicaux.

2 État de l'art

Lors des premières conférences MUC-6 [1] (Marsh et Perzanowski (1998)), la reconnaissance des EN s'intéressait uniquement à des noms de personnes, d'organisation, de lieux, des expressions temporelles, des chiffres, des indications monétaires ou encore des pourcentages. Plus récemment, la REN a évolué sur la reconnaissance d'entités précises dans des contextes spécifiques. Par exemple, on va chercher à reconnaître les noms de gènes dans le cadre de l'étude

1. Message Understanding Conference

de textes en biologie. Des campagnes d'évaluation comme ESTER1 (Gravier et al. (2004)) sur le français ont montré la complexité des phénomènes à prendre en compte. Par exemple, les cas d'entités métonymiques où les noms de lieux sont finalement répartis en trois classes selon que le référent est un nom de lieu, un groupe de personnes ou une organisation. Le nombre des classes évaluées par les modèles peut aller jusqu'à 200 (Sekine et Nobata (2004); Grouin et al. (2011)) en fonction de la finesse recherchée.

L'extraction d'EN repose sur différents types de modèles.

Les modèles à base de traits linguistiques utilisent des informations morpho-syntaxiques (étiquetage grammatical). Ces informations sont ensuite combinées par des patrons (*i.e.* des classifieurs) faits à la main par des experts. Ce type de modèle est efficace sur les domaines spécifiques comme la biologie qui se base sur une terminologie précise. Les principaux inconvénients sont qu'il prend du temps à élaborer et qu'il nécessite de l'expertise sur le vocabulaire du domaine. Le Pevedic et Maurel (2016) dénotent également leur manque de portabilité. Il est nécessaire de refaire les corpus d'apprentissage et d'adapter les systèmes à base de règles pour chaque problème étudié.

Les modèles par *apprentissage*, sont susceptibles d'utiliser les mêmes traits, cependant le classifieur est généré automatiquement via un protocole d'apprentissage. L'inconvénient de ce type de modèle est qu'il faut un certain nombre de données pour réaliser l'apprentissage.

Plusieurs de ces modèles sont présentés dans l'état de l'art de Allahyari et al. (2017)).

3 Contexte applicatif

Notre moteur de recherche Reportlinker utilise un agrégateur d'actualités généralistes, *Moreover Technologies*. Il fournit mensuellement 4 à 6 millions de dépêches d'actualités en langue anglaise, réparties sur 18000 journaux ou sites web.

Dans ce contexte, notre objectif est de détecter les noms d'entreprises et d'organisations mentionnées dans ce flux d'article sans faire appel à aucun lexique (*i.e.* listes d'entreprises déjà pré-définies). L'idée est de créer une liste évolutive afin de s'adapter rapidement aux évolutions économiques du marché (e.g. apparition de nouvelles entreprises sur le marché, obsolescences dans des cas de fusion, etc., voir Stern et Sagot (2010)).

Les titres de presse offrent plusieurs caractéristiques : ils sont généralement très courts (5 mots en moyenne), peuvent être en majuscules (en partie ou sur les premières lettres de tous les mots constituants) ou peuvent encore présenter des singularités stylistiques (par exemple "*CompanyA* to acquire *CompanyB*"). Ces caractéristiques ont des objectifs journalistiques (*i.e.* délivrer des informations concises aux lecteurs) ou sont des erreurs syntaxiques. Ces caractéristiques apportent de l'ambiguïté qui entraîne des erreurs d'étiquetage.

Ce sont sur ces caractéristiques qu'a porté notre travail. Nous nous sommes concentrés sur les transactions d'achats, ventes et fusions d'entreprises. Ces transactions T sont la combinaison de trois composantes : une entreprise "source" S, un verbe de transaction VT et une entreprise "cible" C. Elles sont représentées sous la forme d'un triplet $T\langle S, VT, C\rangle$. Après étude du corpus de *Moreover*, nous sélectionnons trois VT pour notre expérimentation : "to acquire", "to buy" et "to purchase". Nous incluons dans notre liste de VT les formes fléchies (*i.e.* "acquired") ainsi les syntagmes verbaux les plus récurrents contenant au moins un de ces trois verbes (*i.e.* "set to acquire").

L'étape de repérage et d'écriture des patrons a été assez peu coûteuse en temps. Ceci car nous nous focalisons sur un domaine très spécifique (*i.e.* économique et financier) et sur une petite sélection de V T (*i.e.* au nombre de trois). Les titres journalistiques sont en effet généralement très codifiés. L'étape de repérage présente en plus l'avantage d'obtenir un repérage de qualité. L'inconvénient de cette méthode est qu'elle nécessite une exhaustivité : les différentes formes des VT doivent nécessairement être listées pour être repérées.

3.1 Expérimentation

Nous avons constitué un corpus de 1500 titres comprenant un des trois VT listés ci-dessus et au moins deux entités étiquetées "nom propre" (NNP) par la version 3.9.2 de Stanford CoreNLP (Manning et al. (2014)). Nous obtenons alors un étiquetage morpho-syntaxique et une représentation schématique des éléments de la phrase.

Nous présentons, dans la figure 1, un exemple de transaction. Le sujet nominal est indiqué par *nsubj* pour *nominal subject* (*i.e* entreprise "source" S dans notre cas) et l'objet direct est indiqué par *dobj* pour *direct object* (entreprise "cible" C).

La figure 2 présente un cas problématique dû aux majuscules. Dans ce cas précis Standford est trompé par les majuscules du titre et étiquette l'ensemble des mots comme nom propre.

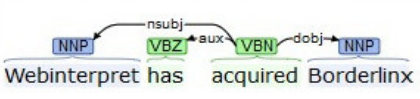


FIG. 1 – *Détection d'une entreprise en acquérant une autre*

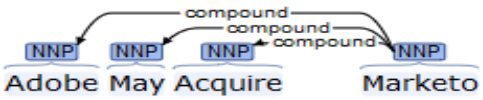


FIG. 2 – *Détection problématique conséquente de majuscules*

La figure 3 souligne que nous pouvons également traiter des informations complémentaires comme le montant de la transaction (étiquette *nmod :for* avec la mention d'un modificateur numérique (*nummod*)). D'autres informations comme la date de l'évènement ou le pourcentage du capital acheté peuvent également être remontées.

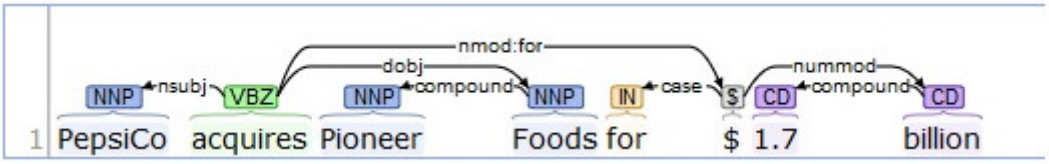


FIG. 3 – *Détection d'une acquisition avec valeur*

Pour améliorer les résultats, nous avons mis en place deux pré-traitements portant sur :

— les majuscules : nous remplaçons les majuscules en minuscules lorsque les mots correspondent à des variations des verbes de transactions VT

— les syntagmes verbaux VT non reconnus par Stanford CoreNLP ont été remplacés par des syntagmes verbaux qui ont la même sémantique ("to acquire" → "will acquire" ; "acquire" → "acquires") et qui sont reconnus et étiquetés de façon exacte par le modèle.

La figure 4 présente l'impact du pré-traitement des majuscules par rapport à l'exemple présenté en figure 2. Dans ce cas de figure, la mise en place de minuscule permet de bien détecter le verbe et par conséquent de bien détecter les sociétés.

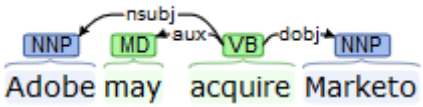


FIG. 4 – *Amélioration de la détection via la mise en place de minuscule.*

Afin d'évaluer nos apports, nous comparons deux modèles :

1. Le premier modèle est le modèle standard de Stanford, il nous sert de référence.
2. Le second modèle correspond au modèle standard auquel nous avons ajouté les deux fonctionnalités de pré-traitements.

3.2 Résultats

Nous avons évalué les modèles sur 1500 titres annotés manuellement. Les résultats obtenus sont présentés dans le tableau 1. On remarque que le modèle augmenté des fonctionnalités de pré-traitements permet de mieux trouver les compagnies : 341 titres comportant des noms de sociétés qui n'étaient pas trouvées le sont (passage de la colonne *Faux Négatif* à *Vrai Positif*). La seconde information intéressante est que l'ajout des deux fonctionnalités ne produit pas de bruit, la colonne de *Faux Positif* restant inchangée entre les deux modèles.

	Vrai Positif	Faux Positif	Faux Négatif	Vrai Négatif
modèle 1	719	78	534	171
modèle 2	1060	78	193	171

TAB. 1 – *Résultats des modèles d'extraction de noms à partir de 1500 titres de presse.*

Le tableau 2 présente le détail des occurrences trouvées dans le corpus pour chaque triplet exploité ainsi que pour l'ajout des majuscules. Les cinq premiers triplets correspondent aux règles présentes dans Standford, les deux dernières correspondent aux nouvelles reconnaissances de syntagmes verbaux. L'ajout des deux fonctionnalités permet donc de mieux classifier 27,4% de l'ensemble des titres comportant deux sociétés.

Le tableau 3 présente les scores de classification des deux modèles. La précision offre une évaluation du bruit en calculant le pourcentage de titres comprenant des sociétés bien classées parmi l'ensemble des titres annotés. Le rappel offre une évaluation du silence en calculant le pourcentage de titres comprenant des sociétés trouvées parmi ceux qui sont censés l'être. La moyenne harmonique de ces valeurs, appelée *F-mesure*, permet d'évaluer les titres bien évalués par les modèles. Le rappel est de 0,57 pour le modèle standard tandis qu'il est de 0,84 pour le modèle amélioré, conséquence de l'ajout des deux fonctionnalités. La précision bénéficie mécaniquement d'une amélioration car l'on classifie mieux les titres. Globalement,

	Distribution des trois éléments (S,VT et C)	Répartition	Pourcentage
$T1$	S ?("has|have") "acquired | bought | purchased" C	317	25,3%
$T2$	S "acquires|buys|purchases" C	169	13,5%
$T3$	C "acquired|bought|purchaded" "by" S	129	10,3%
$T4$	S "signed agreement to" "acquire|buy|purchase" C	95	7,6%
$T5$	S "will" "acquire|buy|purchase" C	9	0,7%
$T6$	S "to" "acquire|buy|purchase" C	169	13,5%
$T7$	S "acquire|buy|purchase" C	142	11,3%
Majuscule	-	30	2,4%
Autres	-	193	15,4%

TAB. 2 – *Présentation des règles permettant de bien classifier les titres.*

nous remarquons que le second modèle permet de mieux trouver les sociétés dans les titres, ce qui est visible par l'amélioration du score de *F-mesure*. Nous expliquons cette différence par l'ajout de pré-traitements portant sur les majuscules et de formes de VT non reconnus par Stanford.

	Rappel	Précision	F-mesure
modèle 1	0.57	0.90	0.70
modèle 2	0.84	0.93	0.88

TAB. 3 – *Scores des modèles d'extraction de noms d'entreprises à partir de titres de presse.*

4 Conclusion et ouvertures

Dans ce papier, nous apportons des améliorations au modèle d'annotation de Standford CoreNLP dans le but d'extraire automatiquement des noms d'entreprises contenues dans des titres d'articles de presse. Les résultats sont prometteurs ; en améliorant la détection de Stanford nous passons d'un score de F-mesure de 0,70 à 0,88, ce qui représente 27,4% de titres mieux annotés. L'étape de listing des verbes de transaction est contraignante mais présente l'avantage d'obtenir un repérage de qualité. Elle permet également de typer automatiquement les noms d'entreprises ainsi trouvés. En effet, la classe de verbes de transactions utilisée met en rapport des noms d'entreprises qui ont exclusivement des relations d'achats, ventes et fusions. Cette méthode pourrait être améliorée en prenant en compte des mots déclencheurs comme les formes juridiques des entreprises ("inc", "corp", etc.). La prochaine étape est de normaliser et de désambiguïser les noms d'entreprises en parcourant le corps de l'article (dans cette expérimentation, nous utilisions uniquement le titre). Cela permettrait d'obtenir du contexte et des données supplémentaires sur l'environnement de l'entreprise : le nom des alias le cas échéant, le nom d'un pays dans lequel elle opère ou un type d'industrie à laquelle elle appartient.

Références

Allahyari, M., S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, et K. Kochut (2017). A brief survey of text mining : Classification, clustering and extraction techniques. In *arXiv preprint arXiv :1707.02919*.

Gravier, G., J. Bonastre, E. Geoffrois, S. Galliano, K. McTait, et K. Choukri (2004). Ester, une campagne d'évaluation des systèmes d'indexation automatique d'émissions radiophoniques en français. In *Proc. Journées d'Etude sur la Parole (JEP)*.

Grouin, C., S. Rosset, P. Zweigenbaum, K. Fort, O. Galibert, et L. Quintard (2011). Proposal for an extension of traditional named entities : From guidelines to evaluation, an overview. In *Proceedings of the 5th linguistic annotation workshop*, pp. 92–100. Association for Computational Linguistics.

Le Pevedic, S. et D. Maurel (2016). Retour sur les annotations des entités nommées dans les campagnes d'évaluation françaises et comparaison avec la tei. *Corela. Cognition, représentation, langage 14*(2), http–corela.

Manning, C., M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, et D. McClosky (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics : system demonstrations*, pp. 55–60.

Marsh, E. et D. Perzanowski (1998). Muc-7 evaluation of ie technology : Overview of results. In *Seventh Message Understanding Conference (MUC-7) : Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Sekine, S. et C. Nobata (2004). Definition, dictionaries and tagger for extended named entity hierarchy. In *LREC*, pp. 1977–1980. Lisbon, Portugal.

Stern, R. et B. Sagot (2010). Détection et résolution d'entités nommées dans des dépêches d'agence. In *Traitement Automatique des Langues Naturelles : TALN 2010*.

Summary

This paper describes a feedback on the Named Entity Recognition process. The experience has been to treat unstructured data sets from economic newspapers. Our aim was to extract automatically the names of companies contained in titles of economic newpapers. We present here the method used by our search engine: ReportLinker. The article describes the experiments carried out leading to a better titer detection of 27.4% and draws the first conclusions from this method.

ARES : un extracteur d'exigences pour la modélisation de systèmes

Aurélien Lamercerie*

*Univ Rennes, Inria, IRISA - UMR 6074, F-35000 Rennes, France
aurelien.lamercerie@inria.fr

Résumé. L'application de méthodes formelles pour assister la conception de systèmes s'appuie sur une modélisation des comportements attendus. La construction de ces représentations nécessite d'extraire les règles comportementales (exigences) généralement définies dans un document de spécifications. Le logiciel ARES (Abstract Requirement Extraction for Systems) répond à ce besoin en partant d'énoncés en langage naturel. Cet outil exploite une représentation sémantique intermédiaire (AMR), et permet de construire un ensemble de définitions abstraites directement exploitables pour modéliser le comportement de systèmes.

1 Introduction

Les représentations sémantiques constituent un intermédiaire intéressant entre l'expression naturelle d'un énoncé et leurs traitements par des méthodes automatiques. Elles permettent de capturer formellement la signification d'un énoncé, le rendant plus accessible pour un traitement automatique. Nous proposons d'exploiter ce type de représentations pour analyser des documents techniques et en extraire les règles comportementales décrivant les évolutions possibles ou interdites d'un système donné.

Nous considérons en particulier des systèmes techniques tels que les avions, les centrales électriques, les véhicules autonomes, etc. Nous nous intéressons plus particulièrement à l'aspect comportemental des systèmes étudiés, spécifié sous la forme d'un ensemble de règles appelées exigences. Par exemple, le cahier des charges d'un système d'accès pour un parking de voitures pourrait contenir des règles telles que les exigences R_1 et R_2 :

— ($R1$) The passage of a car is prohibited if the security gate is closed [1].

— ($R2$) Entry gate must open once a ticket has been issued [2].

L'objectif serait d'assister la conception de tels systèmes dès les premières étapes du cycle de développement. Cela nécessite l'extraction des exigences définies et la construction de représentation formelle. Le logiciel ARES (Abstract Requirement Extraction for Systems) répond à ce besoin. Il permet d'extraire les exigences contenues dans un ou plusieurs documents et fournit en sortie un ensemble de définitions abstraites directement exploitables pour modéliser le comportement de systèmes. Il intègre un analyseur syntaxico-sémantique pour des

1. Le passage d'une voiture est interdit si la barrière de sécurité est en position fermée.
2. La barrière d'entrée doit s'ouvrir une fois le ticket émis.

représentations de sens abstrait (Abstract Meaning Representation, AMR), formalisme qui a fait l'objet de plusieurs travaux récents prometteurs (Banarescu et al. (2013)). La transformation de ces représentations permet de construire des ensembles de définitions abstraites de plus haut niveau, en différenciant les entités du système, les propriétés à évaluer et les exigences à respecter.

2 Chaîne de traitements

Nous cherchons à résoudre un problème, celui de la modélisation du comportement de système, en partant des données telles qu'elles sont disponibles en pratique. Elles prennent généralement la forme de cahier des charges regroupant un ensemble d'exigences, exprimées en langage naturel. La figure 1 donne une vue générale de la chaine de traitements proposée.

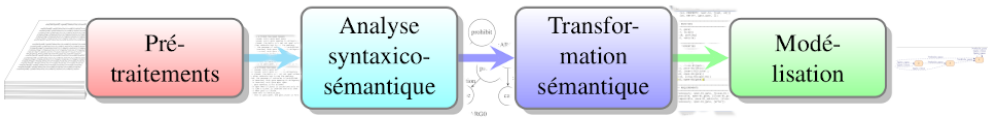


FIG. 1 – *Chaine de traitements du logiciel ARES*

Le processus mis en oeuvre n'implique aucune contrainte particulière sur les données en entrée, que ce soit sur la forme du document ou le texte. La phase de pré-traitement permet d'aboutir à un découpage standard dont l'unité est la phrase. Cette étape, qui n'est pas automatisé dans la version actuelle de l'outil, aboutit à la construction d'une liste d'exigences potentielles. La suite du traitement retiendra les phrases qui correspondent à l'un des motifs sémantiques définis dans l'outil, ces motifs déterminant la structure sémantique attendue des exigences à extraire.

Le traitement principal est entièrement automatisé. Il consiste à produire une représentation abstraite pour chaque exigence, et à transformer chacune de ces représentations en suivant des motifs sémantiques dont la définition est une partie intégrante de l'outil. Le résultat est une construction formelle associant un contexte et une propriété, et caractérisant une évolution possible, interdite ou obligatoire du système.

Les représentations formelles obtenues peuvent être exploitées pour mettre en évidence des erreurs de conception en proposant une modélisation du système étudié. Notre outil intègre une implémentation des interfaces modales sous la forme d'une librairie O'CAML (Caillaud (2011)). Ces modèles supportent des méthodes de raisonnement compositionnel sur les systèmes réactifs, applicables notamment dans l'ingénierie des systèmes, en prenant en compte la variabilité des réalisations possibles induites par une exigence.

2.1 Analyse syntaxico-sémantique

L'objectif d'un analyseur syntaxico-sémantique pour AMR est de construire la représentation abstraite (AMR) d'un énoncé exprimé dans un langage naturel. Plusieurs travaux ont été

publiés ces dernières années avec des approches variées. Le score Smatch, proposé par Cai et Knight (2013), est utilisé pour apprécier les résultats obtenus. Il permet d'évaluer la distance sémantique entre deux représentations, et peut être utilisé pour estimer la qualité d'une représentation construite par une méthode automatique à celle d'un corpus de référence. Les meilleurs parseurs AMR actuels, tels que Liu et al. (2018) ou Zhang et al. (2019) par exemple, obtiennent des scores autour de 70 % sur les corpus de référence.

Notre logiciel intègre le parseur syntaxico-sémantique CAMR, décrit par Wang et al. (2015). La méthodologie proposée consiste à transformer progressivement des analyses de dépendance en AMR, avec un processus itératif sur les noeuds des arbres de dépendance. Ce parseur permet de construire une représentation abstraite pour chaque exigence traitée, comme illustré sur la figure 2 qui montre l'AMR obtenu pour l'exigence $R1$. Sur ce graphe, les arcs définissent des relations entre concepts, chaque arc étant étiqueté par le rôle associé (:ARG0 et :ARG1 sont des rôles centraux, :condition est un rôle spécifique).

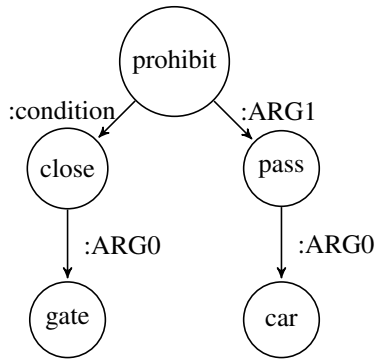


FIG. 2 – *AMR correspondant à l'exigence $R1$*

2.2 Transformation sémantique

La première phase de notre processus produit une représentation abstraite sous forme d'AMR pour chaque exigence potentielle. La seconde étape consiste à transformer ces représentations pour aboutir à un ensemble de définitions abstraites de plus haut niveau, directement exploitables pour la modélisation de systèmes. En sortie, trois ensembles sont proposés : un ensemble d'entités, un ensemble de propriétés et un ensemble d'exigences.

La distinction entre les entités et les propriétés est fondée sur la nature des concepts présents dans la représentation étudiée. Les concepts qui correspondent à une modalité, déontique ou temporelle, ou un opérateur logique (et, ou), sont écartés dans un premier temps. Ils sont exploités pour la définition de l'exigence. Les concepts restants sont considérés comme entité s'ils ne sont pas fondamentalement reliés à d'autres concepts, c'est à dire si aucun rôle de base (core role) ne les relie à un autre concept. A l'inverse, les ensembles de concepts fondamentalement connectés définissent des propriétés.

La définition abstraite d'une exigence est une relation entre un contexte et une propriété. Cette définition peut se représenter comme une AMR de plus haut niveau, où un concept modal

est relié à une évolution envisagée (propriété) et un contexte (ensemble de propriétés). Différents motifs sémantiques sont associés aux définitions abstraites attendues pour déterminer la structure sémantique des exigences à extraire.

A l'issue de ces deux phases, l'analyse de l'exigence $R1$ interdisant le passage d'un véhicule si la barrière est fermée permet de produire un ensemble d'entités, contenant les entités "security-gate" et "vehicle", un ensemble de propriétés, avec les propriétés "security-gate-close" et "vehicle-pass", et la définition "(Interdiction, vehicle-pass, security-gate-close)".

3 Expérimentations et perspectives

Une première expérimentation a été réalisée en s'appuyant sur quelques systèmes théoriques, et donne une indication positive sur l'approche proposée, à confirmer sur un corpus plus complet.

Une analyse préalable a été nécessaire pour adapter manuellement les données en entrée, et construire une liste d'exigences potentielles. L'automatisation de cette étape et différents traitements préliminaires (classification, reconnaissance d'entités nommées) peuvent être envisagée pour améliorer le processus.

De même, plusieurs opérations pourrait être utiles pour ajuster le résultat obtenu, ou classer les exigences. Il est possible qu'une même entité ou propriété soit désignée par des concepts sémantiques différents. Une phase d'alignement, s'appuyant sur une ontologie, permettrait de corriger ces écarts en définissant une équivalence entre différents ensembles de concepts sémantiques.

Références

Banarescu, L. et al. (2013). Abstract meaning representation for sembanking.

Cai, S. et K. Knight (2013). Smatch : an evaluation metric for semantic feature structures.

Caillaud, B. (2011). Mica : A Modal Interface Compositional Analysis Library.

Liu, Y. et al. (2018). An amr aligner tuned by transition-based parser.

Wang, C., N. Xue, et S. Pradhan (2015). A transition-based algorithm for AMR parsing.

Zhang, S. et al. (2019). Amr parsing as sequence-to-graph transduction.

Summary

Using formal methods to assist system design relies on expected behaviors modeling. The construction of these representations requires extracting behavior rules, called requirements, generally defined in a specification document. ARES (Abstract Requirement Extraction for Systems) meets this need starting from a statement of requirements in natural language. This tool operates an intermediate semantic representation (AMR), and converts it into exploitable formal requirements to model the behaviors of systems.

Index

B

Belemkoabga, David-Stéphane 5
Bernard, Jocelyn 31
Bois, François-Xavier 17
Bossard, Aurélien 5
Bruneval, Guillaume 17

E

Essa, Abdallah 5

G

Ghazi, Kaoutar 23

H

Harispe, Sébastien 23

J

Jean, Pierre-Antoine 23

K

Kone, Mohamed 17

L

Lacarne, Mohamed 17
Lamercerie, Aurélien 37
Latour, Marilyne 31

M

Marchal, Sébastien 23
Morbieu, Stanislas 17

N

Nyzam, Valentin 5

P

Putois, Ghislain 1

R

Regal, Corentin 31
Rodrigues, Christophe 5

S

Sutton-Charani, Nicolas 23
Sylla, Kevin 5

T

Tchechmedjiev, Andon 23