TextMine'24

Cédric Lopez, Pascal Cuxac

TextMine '24

Atelier sur la Fouille de Textes

Organisateurs :

Pascal Cuxac (INIST - CNRS),
Cédric Lopez (Emvista)

Organisé conjointement à la conférence EGC
(Extraction et Gestion des Connaissances)
le 23 janvier 2024 à Dijon

Editeurs :

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

Cédric Lopez - Emvista
Espace Bocaud, 42 rue de la Pierre Plantée, 34830 Jacou
Email : cedric.lopez@emvista.com

Publisher:

Pascal Cuxac, Cédric Lopez
2 rue Jean Zay
54519 Vandoeuvre les Nancy Cedex

Vandoeuvre les Nancy Cedex, France, 2024

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 textes au sens large et de leurs applications.

P. Cuxac — INIST-CNRS
C. Lopez — Emvista

Membres du comité de lecture

Le Comité de Lecture est constitué de:

Kevin Cousot (Emvista, Montpellier)

Nicolas Dugué (LIUM, Le Mans)

Natalia Grabar (STL - Lille3, Lille)

Adrien Guille (ERIC, Univ. Lyon2, Lyon)

Jean-Charles Lamirel (LORIA, Nancy)

Denis Maurel (Lifat, Université F. Rabelais, Tours)

Ellouze Mourad (MIRACL Laboratory, FSEGS, Sfax, Tunisie)

Anna Pappa (LIASD, Univ. Paris 8, Paris)

Valentin Pelloin (INA, Paris)

Thibault Prouteau (LIUM, Le Mans)

David Reymond (Université de Toulon, Toulon)

Philippe Suignard (EDF, Paris)

Andon Tchechmedjiev (IMT, Alès)

Sylvain Verdy (Emvista, Montpellier)

TABLE DES MATIÈRES

Session Exposés

Session Exposés Invités

Session Exposés

Session Défi

Extraction de connaissances basée sur l'analyse formelle de concepts en vue de l'assistance aux débats en ligne

Imen Ben Sassi, Hani Guenoune, Alexandre Bazin, Marianne Huchard,
Mathieu Lafourcade, Jean Sallantin

LIRMM, Université de Montpellier, CNRS, Montpellier, France
imen.ben-sassi@lirmm.fr, prenom.nom@lirmm.fr

Résumé. Nous présentons un processus automatisé d'accompagnement de débats visant à extraire des associations entre termes à partir des listes de termes-clés des arguments, listes co-construites par les utilisateurs et notre système d'indexation. L'indexation encourage les utilisateurs à compléter ou corriger la liste des termes-clés, agissant comme un outil incitatif à l'élaboration de points de vue plus structurés. L'algorithme est basé sur l'analyse formelle de concepts et s'appuie sur la base de connaissances JeuxDeMots (JDM). La procédure fait intervenir plusieurs modules menant à une étape d'extraction de connaissances sous forme d'implications destinées à être intégrées dans JDM. Cette approche coopérative permet à la base de connaissances de s'enrichir à mesure que les débats sont analysés, améliorant les termes-clés suggérés par la plate-forme.

1 Introduction

Le projet AREN-DIA (ARgumentation Et Numérique - Didactique & Intelligence Artificielle) [1] vise à sensibiliser les élèves à la pratique du débat dans le cadre de leur éducation à la citoyenneté. La plate-forme de débats en ligne ainsi conçue est également disponible à la société civile, assurant une éthique aux débats. La plate-forme offre la possibilité d'engager des débats structurés à partir d'un texte, renouvelant l'approche traditionnelle des échanges argumentatifs. Elle présente la particularité d'intégrer une technologie collaborative de Traitement Automatique du Langage en vue d'augmenter l'efficacité du processus de débat. Dans cette perspective, AREN-DIA se déploie selon un axe didactique et un axe IA. Les résultats des études menées au sein des lycées révèlent que l'utilisation judicieuse de ce logiciel, insérée dans un dispositif didactique approprié, conduit à un essor marqué des compétences argumentatives chez les élèves (Bächtold et al., 2023).

L'axe IA et ses enjeux font l'objet de cet article. Nous nous intéressons à la manière de concevoir un mécanisme de renforcement incitant les utilisateurs à participer à l'amélioration du système d'IA produisant une représentation structurée des propos d'un débat. AREN [2] est une application web qui offre un espace de débat, réunissant un ensemble d'utilisateurs. Le débat porte sur un texte support publié en amont sur la plate-forme. Les utilisateurs interviennent à travers des commentaires exprimant une opinion, une argumentation ou un avis

1. Ce projet est financé par l'Agence Nationale de la Recherche : ANR-22-FRAN-0001.
2. La plate-forme est accessible via le lien suivant : https://portail-aren.lirmm.fr/aren2023/

sur un segment du texte support ou un commentaire préalablement publié, créant ainsi des embranchements dans l'arbre général du débat. Outre l'intervention des débattants, une procédure automatique vient compléter chaque commentaire en suggérant des termes-clés synthétisant les propos tenus. Cette opération d'indexation représente le point de départ de l'analyse et de l'accompagnement du débat par la machine. Elle est soumise à une complétion par les utilisateurs, qui seront invités à valider, invalider ou compléter ces termes-clés par ceux qu'ils estiment manquants. Afin de lever l'ambiguïté sémantique résultant de la polysémie des termes proposés, nous avons recours à une étape d'enrichissement sémantique des termes pour les préparer à l'opération d'extraction de connaissances basée sur l'analyse formelle de concepts (AFC). Ces connaissances, sous forme d'implications, seront utilisées pour mettre à jour les relations dans la base de connaissances exploitée lors de ces processus, JeuxDeMots (JDM).

L'article s'organise comme suit. Nous détaillons les étapes du fonctionnement général d'AREN dans la section 2. Nous nous pencherons également sur l'algorithme d'accompagnement du débat dans la section 3, qui consiste en la production de termes-clés et d'une analyse AFC pour produire des associations de termes pertinentes. Nous définissons ensuite, dans la section 4, les différentes métriques utilisées pour évaluer l'utilité de l'augmentation sémantique des termes d'indexation des propos du débat. Nous comparons, dans la section 5, les résultats obtenus avec l'AFC avant et après l'enrichissement sémantique des termes d'indexation.

2 Fonctionnement de la plate-forme AREN

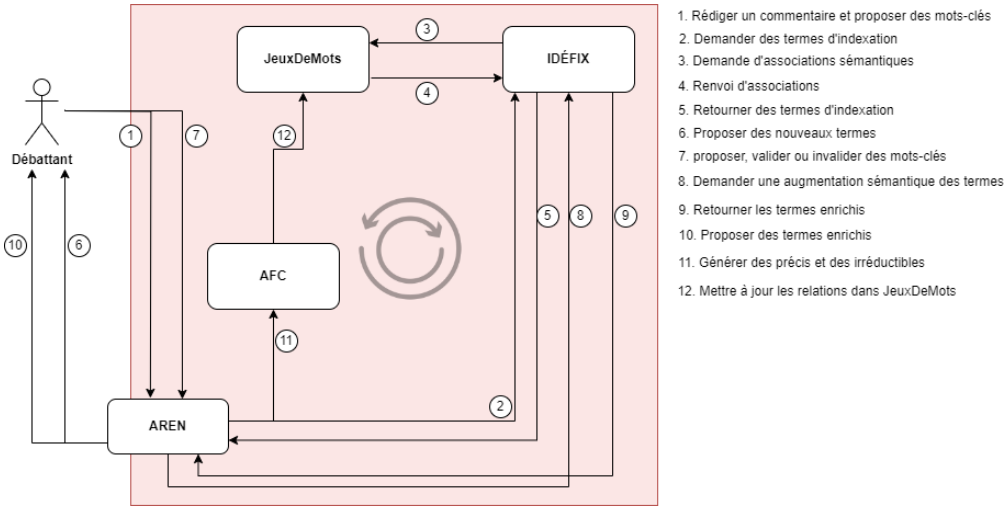

FIG. 1 – *Fonctionnement de la plate-forme AREN sous forme de cycles entre les interactions utilisateurs, le calculateur de termes-clés IDÉFIX, l'analyse formelle de concepts (AFC), et la base de connaissance JeuxDeMots.*

L'application constitue un espace de débat, faisant intervenir un ensemble d'utilisateurs. Un débat porte sur un texte support publié en amont sur la plate-forme, il est conjointement formé par le contenu du texte ainsi qu'un ensemble de commentaires créés par les utilisateurs

et exprimant une opinion ciblée, une argumentation ou un avis sur un segment du texte support ou un commentaire préalablement publié.

Chaque intervention utilisateur comporte une *position* (d'accord ou pas d'accord), une *reformulation*, une *argumentation* et des *mot-clés*. La partie du texte que l'utilisateur souhaite commenter est choisie en sélectionnant, dans le texte de départ, le segment correspondant. La possibilité de sélectionner un segment aussi bien dans le texte support que dans un commentaire préalablement publié, permet la création d'embranchements dans l'arbre général du débat (c.f. Figure 1). Les commentaires sont constitués d'un ensemble d'informations construisant le propos de l'utilisateur, parmi ces informations se définit, entre autres, la position que prend le débattant (d'accord, pas d'accord) vis-à-vis de la sélection (le segment auquel il réagit). Les champs de texte libres, de *reformulation* et *d'argumentation*, sont prévus afin de consolider puis définir, l'avis du débattant.

3 Algorithme d'accompagnement des débats

Nous présentons dans cet article une IA d'accompagnement de débats, KeepTalk (Knowledge Extraction for Enhanced online Public Talks and Argumentative Learning Know-how), dont un des objectifs est d'extraire des associations nouvelles entre termes à partir des listes de termes-clés des arguments d'un débat. L'approche est organisée en plusieurs modules aboutissant à une étape d'extraction de connaissances (c.f. Section 3.3) alimentée par l'analyse formelle de concepts. Après augmentation lexicale (c.f. Section 3.2), cette étape permet de créer des implications entre termes (par exemple, si A est présent alors B et C sont aussi présents). Les implications produites sont destinées à être introduites dans la base de connaissances. Par exemple, si nous avons $A \rightarrow B, C$, alors dans le réseau lexical JDM nous ajouterons : A *r_associated* B et A *r_associated* C. La procédure de description thématique (c.f. Section 3.1) sur laquelle s'assoit l'algorithme s'inscrit dans une démarche collaborative, itérative et incrémentale. Les ensembles de termes indexant chaque commentaire sont co-construits d'un côté, par la procédure automatisée (*IDÉFIX*), et de l'autre, par une *supervision* et *complétion* par les utilisateurs des termes extraits par *IDÉFIX*. Cette supervision est permise en donnant à l'utilisateur la possibilité de *proposer, valider ou invalider* des termes de l'ensemble proposé par l'IA accompagnant le débat. Ce retour est pris en compte lors des itérations de descriptions thématiques ultérieures, menant à une indexation de meilleure qualité. L'objectif étant d'assurer une amélioration de la base de connaissances à mesure que des débats sont analysés, avec en retour une amélioration des termes-clés suggérés par la plate-forme (via *IDÉFIX*) pour les arguments d'un débat. En outre, ce mécanisme est pensé de manière à inciter les utilisateurs à proposer des termes-clés complétant les propos du débat. Plus précisément, le calcul automatique de termes-clés pour un argument est un moyen de donner envie aux utilisateurs, et en particulier à l'auteur de l'argument, de compléter voire de corriger la liste des termes-clés proposés. Un mauvais terme-clé sera en général considéré par l'utilisateur comme une tache/erreur insupportable devant être nettoyée/corrigée.

3.1 Indexation thématique

Les divers arguments des participants au débat sont contenus dans des textes bruts et non-structurés. L'indexation thématique des commentaires a pour objectif d'associer ces données

textuelles à une représentation structurée permettant de synthétiser les propos par des ensembles de termes-clés, référencés dans des bases de connaissances et pouvant servir de point d'entrée à une procédure automatisée. Les termes extraits peuvent désigner des concepts évoqués dans le texte ou des unités lexicales dont la saillance au sein du commentaire est jugée importante. Cette étape d'extraction de mots-clés s'appuie sur des connaissances externes issues du réseau lexico-sémantique *JDM* (Lafourcade et Le Brun, 2023), et est réalisée par le service *IDÉFIX*[3].

JDM est un réseau lexico-sémantique sous forme de graphe orienté. Les nœuds du graphe représentent les termes, tandis que les arcs désignent des relations typées, pondérées et potentiellement annotées entre les termes. Le graphe représente la polysémie des mots en explicitant des raffinements sémantiques hiérarchisés, où un sens spécifique est affilié au sens général du terme. Basé sur une série de notions, principes et outils originaux (ex. la notion de raffinement, la palette des types de relations sémantiques - les éléments d'information, des liens sémantiques entre un type de relation et son inverse (*r_isa* et *r_hypo*, par exemple), l'outil contributif *Diko*, etc.), le réseau JDM est conçu pour une utilisation humaine et comme support de connaissances pour des processus d'intelligence artificielle (analyse sémantique de texte, raisonnement, assistance à la prise de décision, résumé automatique, etc.). Un système de pondération (arcs pondérés, éventuellement négatifs) et de valuation symbolique (annotation en méta-informations, par exemple : rare, pertinent, non pertinent, etc.) a été mis en œuvre pour faciliter des heuristiques de parcours du graphe ainsi que son exploitation.

Le réseau *JDM* peut être utilisé avec des algorithmes classiques liés aux bases de connaissances, mais également sous forme de réseau neuronal (approches hybrides, algorithmes de propagation et de rétro-propagation, etc.). Parmi ces algorithmes, *IDÉFIX* est une sur-couche du réseau *JDM* fondée sur des réseaux de neurones permettant de sélectionner des concepts pertinents pour un texte fourni en entrée. Cette sélection se fait de manière abductive et locale au commentaire, par imitation des exemples déjà appris des interactions précédentes avec l'utilisateur (validation, invalidation et proposition de termes-clés).

3.2 Enrichissement sémantique

Afin d'assurer une représentativité des propos des utilisateurs, nous procédons à l'enrichissement des ensembles de mots-clés produits à l'étape précédente. Nous cherchons, en premier lieu, à assurer une couverture sémantique suffisante en nous occupant des éventuels phénomènes d'ambiguïté lexicale[4] engendrés par la polysémie des termes-clés (c.f. Figure 2).
Ceci revient à *séparer les termes semblables en apparence, mais dont les sens sont différents*, en identifiant les raffinements sémantiques adéquats dans le réseau *JDM*.

L'enrichissement des termes de description par leurs termes synonymes ou hyperonymes pertinents, permet, à l'inverse de la désambiguïsation, de *regrouper les termes différents en apparence, dont les sens sont (quasi-)semblables*. L'ajout d'un synonyme dans les termes-clés d'un propos peut être réalisé sans ou avec restriction, c'est-à-dire dans ce dernier cas, si ce synonyme existe déjà comme mot-clé d'un autre propos du débat, ceci afin d'éviter une dérive liée à des cas de synonymie foisonnante.

3. L'outil *IDÉFIX* est accessible via le lien : https://www.jeuxdemots.org/intern_extract.php
4. Ambiguïté traitée via le service *Bellérophon* : https://www.jeuxdemots.org/intern_desamb.php

Commentaire : *la monnaie locale est un outil financier.*
Indexation : outil conceptuel ; être utile ; outil>moyen d'action ; MLC ; économie locale ; moyen d'action ; monnaie locale ; crise commerciale ; monnaie locale complémentaire et citoyenne ; outil ; financier ; Sol-violette ; économie ; monnaie locale complémentaire ; monnaie ; outil financier ; local
Désambiguïsation de l'indexation : outil>moyen d'action ; monnaie>argent ; économie>activité économique ; financier>finance ; MLC>monnaie locale complémentaire ; monnaie>unité monétaire ; local>propre à un lieu
Augmentation sémantique - synonymes : régional (depuis local>propre à un lieu) ; sous>argent (depuis monnaie>argent)

FIG. 2 – *Exemple de désambiguïsation et d'augmentation sémantique d'indexation d'un propos d'un débat sur les monnaies locales. L'ajout du synonyme régional n'est autorisé que parce qu'il est présent ailleurs dans le débat (dans le cas de l'augmentation avec restriction).*

3.3 Extraction de connaissances

L'extraction de connaissances à partir de l'indexation des commentaires utilise l'AFC, un cadre mathématique basé sur la théorie des treillis permettant la représentation de l'information contenue dans des données sous des formes algébriques ou logiques (Ganter et Wille, 2012).

3.3.1 Contexte formel et fermeture de Galois

L'AFC part de données sous la forme d'un *contexte formel* ; un triplet $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ où $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ est une relation binaire entre des *objets* O et les *attributs* A qui les décrivent. Cette relation peut être représentée sous la forme d'un tableau de croix (c.f. Figure 3).

Dans AREN, les objets sont les commentaires du débat et les attributs sont les mots-clés (ou les termes) proposés par les débattants ou ajoutés lors de la phase d'indexation. Un terme est en relation avec un commentaire s'il l'indexe. Par exemple, dans l'exemple de la Figure 3, $(c_4, monnaie > argent) \in \mathcal{R}$ signifie que l'objet c_4 = *la loi donne une existence légale aux monnaies locales* est indexé par le terme $monnaie > argent$.

	$organisation$	$monnaie > argent$	$cours\ légal$	$monnaie\ complémentaire$	$monnaie\ locale$
c_1	×		×		
c_2			×	×	×
c_3		×			×
c_4		×	×	×	×

FIG. 3 – *Exemple de contexte formel avec une relation binaire entre quatre commentaires (c_i) : c_1 = « la monnaie est une manière de faire et d'organiser la société » ; c_2 = « L'acceptation dans le cadre de la loi rend la monnaie locale légale » ; c_3 = « les monnaies locales nous font nous questionner sur un outil que nous banalisons la monnaie » ; c_4 = « la loi donne une existence légale aux monnaies locales » et cinq termes (t_j) : « organisation » ; « monnaie>argent » ; « cours légal » ; « monnaie complémentaire » ; « monnaie locale ».*

Un contexte formel donne lieu à deux *opérateurs de dérivation*, tous deux notés $\cdot'$ et définis tels que

$$\cdot' : 2^{\mathcal{A}} \mapsto 2^{\mathcal{O}}$$

$$A' = \{o \in \mathcal{O} \mid \forall a \in A, (o, a) \in \mathcal{R}\}$$

$$\cdot' : 2^{\mathcal{O}} \mapsto 2^{\mathcal{A}}$$

$$O' = \{a \in \mathcal{A} \mid \forall o \in O, (o, a) \in \mathcal{R}\}$$

Les compositions $\cdot''$ de ces opérateurs forment des opérateurs de fermeture. Par exemple, dans la Figure 3, la fermeture de $monnaie > argent$ est $\{monnaie > argent, monnaie\ locale\}$.

3.3.2 Génération des irréductibles

Un contexte formel est dit *clarifié* s'il n'a pas deux objets ayant exactement la même description ou deux attributs décrivant exactement les mêmes objets. Dans un contexte clarifié, un attribut a est dit *irréductible* si l'ensemble $\{a\}'$ des objets qu'il décrit n'est pas égal à l'intersection des ensembles d'objets décrits par d'autres attributs (Liquiere, 2021), c'est-à-dire qu'il n'existe pas d'ensemble d'attributs X tel que $\{a\}' = \bigcap_{x \in X} = \{x\}'$. Dans l'exemple de la Figure 3, seul l'attribut $monnaie\ complémentaire$ n'est pas irréductible puisque $\{monnaie\ complémentaire\}' = \{cours\ légal\}' \cap \{monnaie\ locale\}'$. Le reste des termes, à savoir $organisation$, $monnaie > argent$, $cours\ légal$ et $monnaie\ locale$, sont tous des irréductibles.

3.3.3 Extraction des implications

Nous cherchons à extraire des régularités dans la cooccurence des mots-clés dans l'indexation des commentaires. L'AFC offre différentes possibilités de représentation de ces régularités : implications, règles d'association, treillis de concepts ou relations causales (Bazin et al., 2022). Une implication est une règle constituée d'une paire d'ensembles d'attributs A et B, habituellement notée $A \rightarrow B$. Une implication est dite *valide* dans un contexte formel donné si et seulement si tous les objets décrits par les attributs de A sont aussi décrits par les attributs de B, c'est-à-dire $B \subseteq A''$. Ainsi, dans l'exemple de la Figure 3, les deux implications $\{cours\ légal, monnaie\ locale\} \rightarrow \{monnaie\ complémentaire\}$ et $\{organisation\} \rightarrow \{cours\ légal\}$ sont valides tandis que $\{cours\ légal\} \rightarrow \{organisation\}$ ne l'est pas. Afin de réduire le nombre de règles à présenter aux débattants, notre attention se focalise spécifiquement sur les implications de la forme $\{a\} \rightarrow B$ telles que a est un terme irréductible.

3.4 Enrichissement de la base de connaissance

Les implications obtenues avec l'AFC sont utilisées pour mettre à jour les relations dans la base de connaissances exploitée lors de ce processus. Donc, depuis une implication de la forme $\{a\} \rightarrow \{b, c, d, e, ...\}$ nous ajoutons dans la base de connaissances des relations $a \rightarrow \mathbf{x}$ $avec$ $\mathbf{x} \in \{b, c, d, e, ...\}$.

Dans l'exemple de la Figure 3, la mise à jour de la base de connaissances *JDM* se fait par l'ajout de l'association des termes *« cours légal »* et *« organisation »* et celle de *« monnaie locale »* et *« monnaie>argent »*. Ces modifications améliorent globalement la composante associative des calculs ultérieurs des indexations des propos.

4 Mesures d'évaluation des règles

Afin d'étudier l'impact de l'augmentation sémantique sur la qualité des règles, nous avons utilisé diverses métriques, notamment le support, la nouveauté et la surprise (fondée sur la co-occurence ou le voisinage des termes).

4.1 Support

Le support peut-être perçu comme un indicateur de « confiance statistique » d'une règle. Le support d'un ensemble d'attributs ou termes T est le nombre d'objets (ou de commentaires) décrits par T divisé par le nombre total d'objets. Il peut être défini par l'Equation 1.

$$Supp(r) = p(T_r^p \ \ T_r^c)/|C| \tag{1}$$

où T_r^p et T_r^c sont respectivement les termes de la prémisse et de la conclusion de la règle r et C sont les commentaires.

4.2 Nouveauté

La nouveauté est une métrique qui a été utilisée dans les domaines de découverte de sous-groupes et de découverte de clauses (Wrobel, 1997). Une règle est considérée nouvelle si sa prémisse et sa conclusion ne sont pas statistiquement indépendantes (Lavrac et al., 1999).

La nouveauté d'une règle est définie par l'Equation 2.

$$Nov(r) = p(T_r^p \ \ T_r^c) - p(T_r^p) \ \ p(T_r^c) \tag{2}$$

où r est une règle (implication), T_r^p et T_r^c sont respectivement les termes de la prémisse et de la conclusion de la règle r.

4.3 Surprise

Bien que la pertinence peut être facilement évaluée à l'aide du support, la mesure de la surprise (ou de l'inattendu) des règles est une tâche complexe qui nécessite souvent des études coûteuses à mener, impliquant des utilisateurs (ou des ressources externes, dans notre cas). Une règle nouvelle peut être rétrospectivement surprenante ou non, dans le sens où la connaissance disponible ne permet pas de l'expliquer rapidement/facilement.

Dans ce travail, nous ajustons deux définitions de la mesure de surprise utilisées dans le domaine de recommandation (Kaminskas et Bridge, 2014), l'une basée sur le degré d'association sémantique entre les termes indexant les propos du débat et l'autre basée sur les termes associés aux termes d'indexation. Les deux mesures produisent un score qui indique le niveau de surprise que le terme cible a apporté à la règle.

4.3.1 Surprise basée sur la co-occurrence des termes

L'information mutuelle spécifique (Point-wise mutual information notée PMI) indique à quel point deux termes sont statistiquement dépendants, en fonction du nombre de propos indexés par les deux termes et chaque terme séparément (c.f. Equation 3). Les valeurs de PMI

varient entre -1 et 1, où -1 signifie que les deux termes ne sont jamais utilisés ensemble pour indexer un propos, 0 signifie l'indépendance des termes et 1 signifie une co-occurrence systématique des termes.

$$PMI(i,j) = log_2 \frac{p(i,j)}{p(i)p(j)} \quad / \quad -log_2 \; p(i,j) \tag{3}$$

où $p(i)$ et $p(j)$ représentent respectivement les probabilités qu'un propos soit indexé par les termes i et j, tandis que $p(i,j)$ est la probabilité qu'un propos soit indexé par les deux termes i et j.

Sur la base de la PMI, la mesure de surprise d'un terme i pour la règle r est définie comme la valeur moyenne de PMI des termes dans la règle (c.f. Equation 4).

$$Surprise^{avg}_{co-occ}(i,r) = 1 - \frac{1}{|T_r|} \sum_{j \in T_r} PMI(i,j) \tag{4}$$

où i est un terme, r est une règle (implication) et T_r sont les termes de la règle r.

La surprise basée sur la co-occurrence permet de tenir compte du contexte local du débat et des rapprochements de termes que celui-ci peut engendrer. Toutefois, l'indépendance statistique n'implique pas une similarité sémantique faible. En effet, deux contributeurs peuvent respectivement préférer utiliser le terme « *vélo* » et « *bicyclette* ». Ces deux termes sont alors, dans le débat, en co-occurrence nulle ou faible, alors qu'ils sont sémantiquement très proches. La surprise basée sur le contenu sémantique des termes (leur voisinage, c.f. Section 4.3.2) permet de tenir compte de ce type de phénomènes.

4.3.2 Surprise basée sur le voisinage des termes

Notre deuxième mesure de surprise est basée sur la distance appliquée aux termes associés aux termes cibles. Le voisinage d'un terme t dans la base de connaissances JDM est l'ensemble des termes auquel t est relié par la relation d'association d'idées. Nous avons utilisé le complément de la métrique de similarité de Jaccard pour comparer les termes (c.f. Equation 5).

Par exemple, sont associés au terme « *monnaie* » de façon non-exhaustive les termes : « *argent, pièce, billet, euro, devise* », le terme « *fric* » aura comme termes associées : « *argent, pièce, billet, euro, thune* ». La distance entre ces deux termes est de $1 - 4/6 = 1/3$.

$$dist(i,j) = 1 - \frac{A_i \cap A_j}{A_i \cup A_j} \tag{5}$$

où A_i et A_j sont respectivement les ensembles de termes associés aux termes i et j. Dans le cas où le terme A est polysémique, on considère sa désambiguïsation lexicale pour extraire les termes qui sont associés au contexte des règles.

Pour mesurer la surprise d'un terme, nous calculons la distance moyenne entre le terme cible i et les autres termes T_r de la règle r comme indiqué dans l'équation 6.

$$Surprise^{avg}_{vois}(i,r) = \frac{1}{|T_r|} \sum_{j \in T_r} dist(i,j) \tag{6}$$

où i est un terme, r une règle (implication) et T_r sont les termes de la règle r.

5 Résultats et discussions

Afin d'aider à l'interprétation des résultats de l'algorithme, nous commençons dans cette section par présenter les données ayant servi à cette évaluation. Nous cherchons ensuite à mettre en évidence la pertinence de chaque module employé, ceci en mettant en place des configurations contrastives de l'algorithme rendant possible la comparaison des résultats permis par chaque sous-module.

5.1 Jeux de données et configurations

Nous procédons à l'évaluation de notre approche à l'aide des données issues d'un débat sur la plate-forme AREN concernant les monnaies locales [5] intitulé « Les monnaies locales sont-elles un outil pour sauver l'économie locale et dans quelles conditions ? ». Les principales caractéristiques de notre jeu de données sont présentées dans le Tableau 1.

Débattants	Arguments	Mots-clés	Période
8	48	464	Mars 2020 – Mai 2023

TAB. 1 – *Statistiques du débat sur les monnaies locales.*

Chaque argument d'un débattant est associé à un texte initial du débat et décrit par une reformulation, une phrase qui reflète sa compréhension du texte argumenté (*« La monnaie locale est un outil financier »* : Figure 2), et une opinion (83.33% des arguments sont « *plutôt d'accord* » et 16.67% ne sont « *plutôt pas d'accord* »). En total, 464 mots-clés distincts ont été utilisés pour indexer les reformulations dont 125 termes uniques sont proposés par les utilisateurs et 339 par *IDÉFIX*. En moyenne, chaque débattant a utilisé 5.39 termes par argument.

Nous comparons les résultats de trois variantes de notre approche pour mesurer l'effet de l'augmentation sémantique sur la qualité des résultats de l'AFC. Les détails de nos méthodes sont énumérés ci-dessous :

- KT : Les implications sont calculées à partir du contexte d'extraction initial, défini par la relation binaire entre les reformulations des débattants et les termes-clés qui les indexent.
- $KT^{\dagger}$: Le contexte d'extraction est enrichi par les synonymes des termes qui définissent les attributs pour générer les implications. On s'intéresse aux termes synonymes qui sont déjà utilisés lors de l'indexation initiale ($R^{\bullet}$) et aussi ceux qui ne le sont pas (R°), donc, de nouveaux termes n'apparaissant pas dans le débat.
- $KT^{\ddagger}$: Identique à la configuration précédente avec des hyperonymes au lieu de synonymes.

5.2 Résultats

Nous commençons par proposer une vue quantitative des résultats des différentes configurations. Nous rapportons, dans le Tableau 2, le nombre d'*attributs*, *irréductibles* et *implications* dans les configurations se limitant, ou pas, aux termes-clés du débat.

Quand l'ajout de termes n'est pas restreint à ceux du débat, nous observons une augmentation du nombre d'attributs. Inversement, si on se restreint aux termes du débat, le nombre

5. https://portail-aren.lirmm.fr/aren2023/debates/6

d'attributs est constant. Dans tous les cas, l'utilisation de la désambiguïsation lexicale réduit le nombre d'objets produits (irréductibles et implications), ceci est conforme à l'intuition car la désambiguïsation réduit l'éparpillement lexical. Par ailleurs, l'ajout d'hyperonymes est plus productif que l'ajout de synonymes, car il est possible de trouver au moins un hyperonyme pour la quasi-totalité des attributs (qui sont des termes), ceci est beaucoup moins vrai pour les synonymes.

	KT	$KT^{\dagger}$		$KT^{\ddagger}$	
		DL°	$DL^{\bullet}$	DL°	$DL^{\bullet}$
Avec restriction aux termes-clés du débat					
Attributs	464	464	464	464	464
Irréductibles	70	73	68	83	79
Implications	43	54	46	75	71
Sans restriction aux termes-clés du débat					
Attributs	464	3831	2240	1125	866
Irréductibles	70	161	85	121	103
Implications	43	114	50	106	88

TAB. 2 – *Résultats de KT (KeepTalk) avec et sans restriction aux termes-clés du débat : Le nombre d'objets demeure constant pour toutes les configurations et est égal à* 48. $DL^{\circ}/DL^{\bullet}$ *désignent l'utilisation ou non de la tâche de désambiguïsation lexicale.*

La première expérimentation rapporte la proportion des relations d'association qui sont considérées comme correctes/pertinentes. Ces associations sont générées à partir des règles produites (implications). Cette étape consiste en une évaluation menée manuellement par 4 intervenants adoptant le rôle « d'experts ».

Sans augmentation	60.11 %	
	DL°	$DL^{\bullet}$
Augmentation avec restriction ($R^{\bullet}$)	63.12 % (1)	72.07 % (2)
Augmentation sans restriction (R°)	42.60 % (4)	76.77 % (3)

TAB. 3 – *Pourcentage des bonnes associations selon une évaluation manuelle menée par 4 experts.*

Nous cherchons à travers le Tableau 3 à classer les configurations de notre système en termes de « qualité », du point de vue d'utilisateurs humains. Le cas 1 signifie que dans une configuration se restreignant aux termes du débat et sans procédure de désambiguïsation, 63.12% des relations d'association à ajouter à la base de connaissance sont jugés correctes par les experts. Pour la meilleure configuration (cas 3), où nous procédons à une désambiguïsation sans se limiter aux termes-clés du débat, nous obtenons 76.77% de bonnes associations.

Dans le Tableau 4, nous constatons que la nouveauté est globalement très faible, ce qui indique que l'on trouve peu d'associations n'existant pas dans la base de connaissances. Ceci est positif du point de vue de la complétude de la base. On constate par ailleurs que la surprise est globalement très haute, ce qui veut dire qu'une information nouvelle n'aurait pas pu être

	KT	$KT^{\dagger}$		$KT^{\ddagger}$	
		DL°	$DL^{\bullet}$	DL°	$DL^{\bullet}$
Avec restriction aux termes-clés du débat : $R^{\bullet}$					
Support	0.0667	0.1300	0.0836	**0.1658**	0.1390
Nouveauté	0.0546	0.0681	0.0564	**0.0733**	0.0690
Surprise par co-occurrence	0.5662	**0.7995**	0.5304	0.1270	0.1294
Surprise par voisinage	0.9488	0.9623	0.9706	0.9674	**0.9718**
Score agrégé	0.2103	0.2872	0.2215	0.1965	0.1863
Score agrégé syn+hyper		0.4838		0.4079	
Sans restriction aux termes-clés du débat : R°					
Support	0.0667	0.1352	0.0978	**0.1667**	0.1359
Nouveauté	0.0546	0.0780	0.0615	**0.0799**	0.0676
Surprise par co-occurrence	0.5662	**0.7167**	0.4534	0.3481	0.1957
Surprise par voisinage	0.9488	0.9502	**0.9602**	0.8997	0.9497
Score agrégé	0.2103	0.2911	0.2262	0.2541	0.2032
Score agrégé syn+hyper		0.5452		0.4294	

TAB. 4 – *Comparaison des résultats de l'analyse formelle de concepts : KT avec le contexte initial ; $KT^{\dagger}$ avec le contexte augmenté avec les synonymes ; $KT^{\ddagger}$ avec le contexte augmenté avec les hyperonymes. L'augmentation est faite avec et sans restriction aux termes du débat.*

inférée, dans la quasi-totalité des cas. Ceci est un autre résultat très positif qui justifie l'utilité de notre approche d'extraction de connaissances. Nous observons un effet conjoint à l'étape de

Sans augmentation	0.1264	
	DL°	$DL^{\bullet}$
Augmentation avec restriction ($R^{\bullet}$)	0.3048 (1)	0.2939 (2)
Augmentation sans restriction (R°)	0.2290 (4)	0.3264 (3)

TAB. 5 – *Combinaison des résultats des métriques avec la proportion des bonnes associations (évaluation manuelle) - Il s'agit d'un score et non d'un pourcentage.*

désambiguïsation DL et à la restriction R ou non aux termes-clés du débat. Ce croisement est clarifié dans le Tableau 5. La configuration la plus favorable est celle avec une augmentation avec synonymes et hyperonymes sans R et avec étape de DL (cas 3). La seconde meilleure configuration est celle sans DL et avec R (cas 1). La pire configuration, qui a de très mauvais résultats, est la combinaison de R° et DL° (cas 4). Le score du dernier cas (cas 2 : $R^{\bullet}$ et $DL^{\bullet}$), bien que correct, est inférieur aux cas 1 et 3.

La désambiguïsation lexicale (DL) et la restriction (R) aux termes déjà présents dans le débat, visent le même but, contrôler le foisonnement lexical, et ne pas tomber dans le piège de polysémie. L'approche avec R permet de ne pas introduire de termes qui ne sont pas apparus dans le débat, il n'y a donc aucune chance d'introduire, par accident, un terme sans rapport. Le cas 2 est intéressant car il n'est pas intuitif : en effet, on s'attendrait à ce que l'action conjointe de D et R donne les meilleurs résultats, or ce n'est pas le cas. A priori l'effet restrictif conjoint de DL et R empêche un rapprochement efficace des propos du débat. Ne pas faire de R permet d'augmenter la richesse des associations, toutefois cette richesse doit être contrôlée par la DL.

6 Conclusion et perspectives

Les résultats obtenus sont prometteurs et soulignent l'efficacité d'effectuer conjointement une analyse basée sur l'AFC et une augmentation lexicale à partir d'une base de connaissances. En perspective, il serait important, sur la base des scores de l'évaluation manuelle, d'agréger les scores des métriques automatiques de manière à obtenir un score global qui serait représentatif de la qualité (que nous avons cherché à obtenir ici par une évaluation manuelle). Concernant l'extraction des connaissances, en perspective, le projet explorera d'autres représentations des régularités : autres implications, règles d'association et relations causales ; ce qui permettra d'ajouter dans la base de connaissances des informations sur des types de relations plus précises (autres que les associations d'idées).

Références

Bächtold, M., G. Pallarès, K. De Checchi, et V. Munier (2023). Combining debates and reflective activities to develop students' argumentation on socioscientific issues. *Journal of Research in Science Teaching 60*(4), 761–806. 1

Bazin, A., M. Couceiro, M.-D. Devignes, et A. Napoli (2022). Steps towards causal formal concept analysis. *International Journal of Approximate Reasoning 142*, 338–348. 6

Ganter, B. et R. Wille (2012). *Formal concept analysis : mathematical foundations*. Springer Science & Business Media. 5

Kaminskas, M. et D. Bridge (2014). Measuring surprise in recommender systems. In *Proceedings of the Workshop on recommender systems evaluation : dimensions and design held in conjunction with RecSys 2014*, REDD '14, Silicon Valley, USA. ACM. 7

Lafourcade, M. et N. Le Brun (2023). Apport du jeu pour la construction de connaissances : le projet jeuxdemots. *Technologie et innovation 8*(Le jeu pour innover). 4

Lavrac, N., P. A. Flach, et B. Zupan (1999). Rule evaluation measures : A unifying view. In *Proceedings of the 9th International Workshop on Inductive Logic Programming*, ILP '99, Berlin, Heidelberg, pp. 174–185. Springer-Verlag. 7

Liquiere, M. (2021). Utilisation des irréductibles d'un treillis de concepts pour la sélection de motifs. Technical report, LIRMM, Université de Montpellier. 6

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery*, Berlin, Heidelberg, pp. 78–87. 7

Summary

We present an online debate analysis algorithm aiming to extract new associations between terms from co-constructed keyword lists of arguments by users and our indexing system.The calculation of keywords encourages users to supplement or correct the keyword list, serving as an incentive tool for developing more structured contributions. The algorithm is based on formal concept analysis and relies on the JeuxDeMots knowledge base. The procedure involves multiple modules leading to a knowledge extraction step in the form of implications which are to be integrated into JDM. This cooperative approach allows the knowledge base to enrich itself as debates are analyzed, improving the platform's suggested keywords.

Employing Graph Neural Network for Syntactic Dependency-based Document Classification

Kevin Cousot*, Shaghayegh Momtaz**, Mehdi Mirzapour***

*Emvista, Jacou, France
kevin.cousot@emvista.com,
** Economics Dept, University of Tehran, Iran
shaghayegh.momtaz@email.kpu.ca
***LIRMM, Montpellier, France.
mehdi.mirzapour@lirmm.fr

Abstract. The typical approach to document classification involves treating text as sequential data and employing transformer-based architectures for classification. However, this sequential representation poses challenges in identifying implicit relationships within the text and leveraging them effectively. This paper investigates the use of syntactic dependency graphs as alternative relational representation, thus reformulating document classification as graph classification. Such graph encode the grammatical structure, connecting words based on their relationships and roles within the sentence. This explicit representation facilitates the understanding of the underlying syntactic and semantic structures. Interpretable dependency graphs are then processed by Graph Neural Networks, a type of architecture that excels at capturing complex relationship between entities in a graph by propagating information through the graph structure. To test this approach, text grammatical structure is computed with a dependency parser and individual node representations initialized with word-embeddings. We experiment with six distinct Graph Neural Networks architectures as classifiers. Results are then evaluated on three datasets, namely AG-News, TREC-6 and Arxiv and compared to state-of-the-art. Experiments show representing texts by their syntactic dependency graph and using graph neural networks for classification achieves very competitive results compared to classical sequence-based model.

1 Introduction

Document classification is an important task in Natural Language Processing (NLP) which can be formulated as classifying a given document based on some pre-defined categories. It covers a broad range of downstream tasks such as topic classification (Xia et al., 2019), sentiment analysis (Tan et al., 2023), and opinion mining (Ravi and Ravi, 2015). Traditional document classification methods use different architectures, such as support vector machines (SVMs) from Cortes and Vapnik (1995), convolutional neural networks (CNN) as proposed

by LeCun et al. (1989), and recurrent neural networks (RNN) as introduced by Hochreiter and Schmidhuber (1997). Recently the research is more oriented toward Transformer-based models such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019).

Although the main state-of-the-art approaches are clustered around BERT family models, Graph Neural Networks also have been recently used as a powerful architecture for document classification (Wang et al., 2023). To do so, a graph is normally represented as the interconnection among documents, where nodes represent textual components like words and documents, and the edges are established according to the syntactic or semantic relations between these nodes. This approach facilitates a comprehensive analysis of the intricate relationships within textual data, enabling a deeper understanding of the underlying syntactic and semantic structures.

The adoption of GNN-based architectures offers distinct advantages over the conventional BERT-based choices. GNNs inherently possess the capability to capture complex relationships between entities within a graph, making them well-suited for modeling and learning from structured data, such as dependency graphs. This enables the extraction of richer contextual information and the incorporation of global document-level semantics, thus enhancing the overall discriminative power of the classification model. Additionally, the use of GNNs facilitates the integration of domain-specific knowledge and prior information, enabling the development of more interpretable and explainable models, which is crucial for tasks that demand transparency and accountability, such as document categorization in legal or regulatory contexts.

Current studies in Graph-based document classification are diverse in many aspects. Yao et al. (2019) uses Convolutional Networks (Text GCN) with heterogeneous word document graph for a whole corpus. Han et al. (2022) conducts a comprehensive analysis of the role of node and edge embeddings in a graph and its GCN learning techniques. In another study (Hua et al., 2022) a semantic hierarchical graph neural network (HieGNN) is introduced to construct graphs from word-level, sentence-level, and document-level, respectively. BertGCN, presented by Lin et al. (2021), integrates large-scale pretraining and transductive learning for text categorization. BertGCN establishes a mixed graph across the dataset and defines documents as nodes employing BERT representations.

There are important considerations for the choice of syntactic dependency graph derived from texts instead of using simple sequential representations. The dependency graph explicitly represent syntactic relationships between words, capturing dependencies like subject-verb-object or modifiers. This can provide a more structured and syntactically informed representation compared to a linear sequence of words. The dependency graphs can help model long-range dependencies more effectively than a linear sequence. GNNs can propagate information through the graph structure, allowing the model to capture dependencies that span across a greater distance in the input text. Syntactical structures implicitly encode information about entities, relationships, and actions in the text and they are often more robust to variations in word order. Since the graph structure captures relationships, the model may generalize better to different word orders and syntactic constructions. Moreover, graphs can offer more interpretable representations of the relationships between words or entities.

The synergy between BERT and Dependency-based GNNs presents a promising hybrid approach in classification tasks. By combining the strengths of both models, this approach leverages the rich contextual embeddings of BERT while benefiting from the explicit struc-

tural information and interpretability offered by dependency graphs and GNNs. In classification tasks, the hybrid approach allows for a nuanced understanding of relationships within the data, providing a more comprehensive basis for decision-making. This amalgamation of contextual embeddings and explicit structural information bridges the gap between the power of BERT and the transparency of dependency-based GNNs, offering a balanced and effective solution for a wide range of natural language processing challenges.

Our three main contributions are:

1. We have utilized for the first time –as far as we know– the syntactic dependency graph as input to Graph Neural network in text classification task.

2. We studied the impact of using different word representation as initial node embeddings.

3. Finally, we used six GNN architectures and compared them with other sequenced-based and graph-based architectures achieving competitive results.

The paper is organized as follows: Section 2 studies the previous approaches and current state of the art, with a focus on data representation, either as a sequence or graph. Section 3 defines the problem of document classification. Section 4 presents our method centered around the idea of reformulating the task as a graph classification problem. We describe how the text is transformed into a graph by exploiting its syntactic tree and the GNN architectures used to perform the classification. Section 5 describes experiments and obtained results carried out to determine the effects of using text syntactic structure rather than its original sequential form. Section 6 discusses and analyzes the obtained results. Finally, section 7 concludes our paper and lays out the way for future work and experiments. Experiments code and model configurations are freely available on github[1].

2 Literature Review

In recent years, the field of document classification has witnessed substantial advancements. They can be broadly categorized into two distinct approaches: sequence-based document classification and graph-based document classification.

Sequence-based document classification focuses on the sequential structure of text data, where the contextual information is predominantly derived from the linear arrangement of words or tokens within the document. This approach often involves the utilization of Recurrent Neural Networks (Sherstinsky, 2020) or more sophisticated architectures like transformers, which excel in capturing patterns and dependencies within the sequential data. The development of pre-trained language models, such as BERT, GPT, and their subsequent variations, has significantly boosted the performance of sequence-based document classification tasks, allowing for a more nuanced understanding and representation of textual data. As follows, we explain some of the main state-of-the-arts approaches.

[1] https://github.com/Emvista/Gnn4DependencyDocumentClassification

2.1 Sequence-based Classification

Many classification architectures were introduced for the sequence-based classification task. Johnson and Zhang (2016) focus on Long Short-Term Memory (LSTM) networks and region embeddings for both supervised and semi-supervised text categorization tasks. This is achieved by combining LSTM and convolution layers trained on unlabeled data which indicates that embeddings of text regions convey complex concepts better than isolated word embeddings. In contrast to the previous approach, another LSTM-based model called Deep Pyramid Convolutional Neural Networks (Liang et al., 2021) incorporates a deep pyramid structure with shortcut connections, allowing for efficient feature extraction and classification. In another research (Conneau et al., 2016), inspired by the computer vision field, a new model is created using deep convolutional networks. This model is more complicated in comparison to classical LSTMs, and convolutional neural networks. Based on this strategy a new architecture, VDCNN, is introduced for text processing. It operates directly at the character level and uses only small convolutions and pooling operations. One of the nice properties of this approach is that the performance of this model increases with the convolutional layers depth.

Yang et al. (2019), introduces XLNet, a novel approach to language understanding through autoregressive pretraining, inspired by GPT. This large language model has achieved better results compared to BERT in the classification task. Unlike traditional autoregressive models that use a left-to-right or right-to-left approach, XLNet employs a permutation-based training technique that considers all possible permutations. This method allows the model to capture bidirectional contexts and overcome the deficiencies of traditional autoregressive models. The main advantage of XLNet is that it achieves a better understanding of language by incorporating both global and local context.

2.2 Graph-based Classification

Graph-based document classification leverages the inherent structural information within the text, emphasizing the relationships and dependencies between words or entities as represented by graph structures. This approach typically involves the extraction of syntactic and semantic relationships using techniques such as dependency parsing, co-reference resolution, or entity linking, which are then utilized to construct informative graph representations of the document. Interested readers can find a great introductory survey to Graph Neural Networks (GNNs) for text classification in Wang et al. (2023). In this section, we focus more on related literature to our research. Yao et al. (2019) propose a text classification method named Text Graph Convolutional Networks (Text GCN) which uses a simple two-layer GCN. It employs a heterogeneous word document graph for a whole corpus and turned document classification into a node classification problem. The edges are built among nodes based on word occurrence in documents (document-word edges) and word co-occurrence in the whole corpus (word-word edges). The weight of the edge between a document node and a word node is the term frequency-inverse document frequency (TF-IDF) of the word in the document. Text GCN can capture global word cooccurrence information and utilize limited labeled documents well.

Han et al. (2022) conducts a comprehensive analysis of the role of node and edge embeddings in a graph and its GCN learning techniques in text classification. The general methodol-

ogy is similar to Yao et al. (2019). In another study Hua et al. (2022), a semantic hierarchical graph neural network (HieGNN) is introduced to construct graphs from word-level, sentence-level and document-level, respectively. It uses graph neural networks (GNNs) to act on the three levels mentioned above to obtain the output of the three-level vector representations. In the end, the embeddings are merged and fed as input to the softmax and linear layer for classification. BertGCN, presented by Lin et al. (2021), integrates large scale pretraining and transductive learning for text categorization. BertGCN establishes a mixed graph across the dataset and defines documents as nodes employing BERT representations. It establishes connections between words and documents, as well as between words themselves, relying on term frequency-inverse document frequency (TF-IDF) and positive point-wise mutual information correspondingly. The architecture of BertGCN is adaptable and can be implemented on various document encoders and graph models.

3 Problem Definition

According to Jurafsky and Martin (2009), document classification is the process of assigning labels to documents based on their content in order to categorize them into predefined categories or topics. In our specific setting, we can formally define document classification as follows: let $\mathcal{D} = \{d_1, \ldots, d_n\}$ be the set of documents and $\mathcal{L} = \{l_1, \ldots, l_m\}$ the set of corresponding labels. With g, a function that maps the sequential text to its syntactic dependency graph, we can create the graph dataset $\mathcal{G} = \{(g(d_1), l_1), \ldots, (g(d_n), l_n)\}$. The dataset is partitioned in two subsets, $\mathcal{G}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$. The aim is to learn a parametric classification function $f_\Theta : \mathcal{G} \to \mathcal{L}$ using $\mathcal{G}_{\text{train}}$, that predicts the label of a given document. Evaluation is then performed on $\mathcal{G}_{\text{test}}$ to determine the model ability to generalize on unseen examples.

4 Method

Effective data representation is a decisive factor for achieving good results in text classification task. Raw text is inherently sequential and most often used in that form, making it challenging to take advantage of its implicit relationships. At the core of our method lies the transformation of sequential text documents into relational graphs. This transformation allows the introduction of relational information explicitly and the reformulation of the task from document classification to graph classification.

To construct the graph we chose to use the text's syntactic dependency tree, connecting words according to their grammatical relationships. GNNs ability to capture and exploit relational information is then leveraged to perform the classification. Because GNNs require nodes to start with an embedding, document word representations are computed with a transformer-based word embedder and used as initialized node weights in the graph.

4.1 Sequence to Dependency Graph Transformation

The transformation of textual documents into their graph representations is done by analyzing the grammatical structure of each document using Stanza, a syntactic parser. It performs dependency parsing, according to the Universal Dependencies (UD) annotation framework

representing grammatical relationships between words. Edges are labeled with the dependency type, denoting the nature of the relationship, while directionality specifies the word's role within that relationship.

Unlike the sequential representation, which treats text simply as a sequence of words, dependency trees explicitly represent the syntactic relationship between words and their role within the sentence. Since dependency relationships are distance-independent the model can access long-range relationships in the same sentence. Althought syntax and semantic are distinct, we hypothesize that making syntax explicit should also help the model to capture semantic.

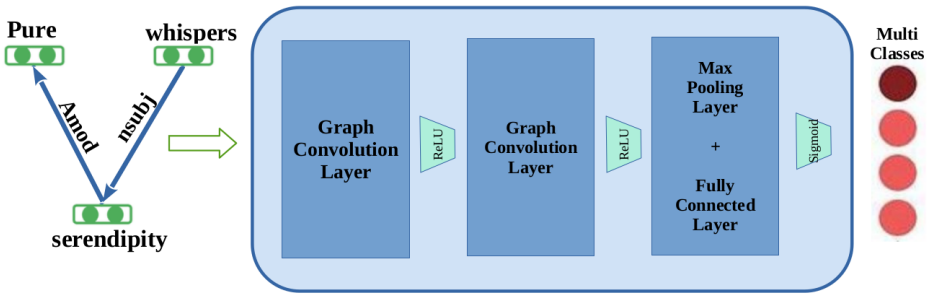

FIG. 1: Pipeline - Graph classification

4.2 Initializing Graph Node Embeddings

GNN require each node to be initialized with an embedding vector. To provide such initial embedding, the entire document is fed to an embedder from which word embeddings are extracted. In this study, we experiment with two models, DistilBERT (Sanh et al., 2020) and BERT-large (Devlin et al., 2018). Both these models use WordPiece tokenization (Wu et al., 2016), a subword tokenization model. Because every node was built from a word, there may be nodes that correspond to multiple tokens. In such cases, only the first token of the word is pooled.

4.3 Graph Neural Network Architectures

This section describes the six architectures we experimented with. They all proceed in three steps: (i) update initial node embeddings X with an GNN-based encoder (ii) aggregate all node embeddings in a single, fixed-size vector using a global pooling function ϕ and (iii) use this graph-level embedding to perform the classification. All the architectures used here follow the same sequence:

$$X' = \text{encoder}(X)$$
$$h_g = \phi(X')$$
$$out = \text{classifier}(h_g)$$

4.3.1 GCN

GCN were originally introduced for semi-supervised node classifications on graphs (Kipf and Welling, 2016). It learns hidden layer representations that encode both local graph structure and features of nodes. Node embeddings are updated according to the following rule:

$$x'_i = W^\top \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j, \hat{d}_i}} x_j$$

with W the learnable weights, $\mathcal{N}$ the neighbors function, $\hat{d}_i$ the incoming degree of node i and $e_{j,i}$ the weight of the edge from j to i (defaults to 1). The GCN architecture is made of two layers of GCN, also using batch normalization (Ioffe and Szegedy, 2015) and dropout inbetween. After a global pooling function, a single linear classifier with sigmoid activation is applied.

4.3.2 GAT

Graph attention networks (GAT) introduce the use of masked self-attention layers to attend node neighborhoods (Veličković et al., 2018). Attention coefficients α are used to weigh the degree of contribution.

$$x'_i = \alpha_{i,j} W_s x_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_t x_j$$

$$\alpha_{i,j} = \frac{\exp(a^\top \text{LeakyReLu}(W_s x_i + W_t x_j))}{\sum_{k \in \mathcal{N}(i) \cup \{u\}} \exp(a^\top \text{LeakyReLu}(W_s x_i + W_t x_k))}$$

with W_t and W_s the weight matrices for source and target respectively. This architecture stacks two layers of GAT followed by a classifier made out of two linears with a sigmoid.

4.3.3 GCGAT

k-GNN (Morris et al., 2021) generalizes GNN in their ability to distinguish non-isomorphic graphs and take higher-order graph structures into account.

$$x'_i = W_1 x_i + W_2 \sum_{j \in \mathcal{N}(i)} e_{j,i} \cdot x_j$$

Brody et al. (2022) shows that because GAT attention is static and the model expressivity is limited. GATv2 addresses this restriction by recalculating attention weights at each layer.

$$x'_i = \alpha_{i,j} W_s x_i + \sum_{j \in \mathcal{N}} \alpha_{i,j} W_t x_j$$

$$\alpha_{i,j} = \frac{\exp(a^\top \text{LeakyReLu}(W_s x_i + W_t x_j)}{\sum_{k \in \mathcal{N}(i)} exp(a^\top \text{LeakyReLu}' W_s x_i + W_t x_k)}$$

This architecture consists in two k-GNN layers with batch normalization and dropout. It is then chained with a GATv2 layer. The classifier is a single linear layer.

4.3.4 GIN

Graph Isomorphism Network (Xu et al., 2019) are Message Passing based network that combine aggregated features with the original node features through a neural network.

$$x'_i = \Theta((1+\epsilon) \cdot x_i + \sum_{j \in \mathcal{N}(i)} x_j)$$

with Θ a neural network. The architecture is a two-layer GIN model with Θ set as two linear layers with ReLu activation. The final classifier is a single linear layer with sigmoid activation.

4.3.5 TransCN

Graph Transformer Network (Shi et al., 2021) uses the same multi-head attention as regular transformers. It also uses Unified Message Passaging Model a novel message passing method that combines node features propagation with labels.

$$x'_i = W_1 x_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_2 x_j$$

$$\alpha_{i,j} = \text{softmax}(\frac{(W_3 x_i)^\top (W_4 x_j)}{\sqrt{d}})$$

Our TransCN architecture consists of three Graph Transformer layers, with batch normalization and dropout inbetween. The classifier is a single linear layer.

4.3.6 R-GCN

Relational-GCN (Schlichtkrull et al., 2017) are an extension of regular GCNs for large-scale relational data, specifically for link prediction and entity classification. Because each relation has its own weights W_r, the model is capable to learn relation-specific transformations.

$$x'_i = W_{root} \cdot x_i + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}(i} \frac{1}{|\mathcal{N}_r(i)|} W_r \cdot x_j$$

Relation Graph Attention Networks (Busbridge et al., 2019) then extend Relational-GCN adding attention mechanism. In our architecture, we stack two layers of Relational-GCN and one Relation Graph Attention Network, also using batch normalization (Ioffe and Szegedy, 2015) and dropout in-between. After a global pooling function, a single linear classifier is applied.

5 Experiments

5.1 Datasets

Experiments are performed on three classic datasets for document classification, namely AG-News, TREC-6, and a subset of Arxiv. AG-News (Zhang et al., 2016) is a collection of news articles with four classes, 120k training examples and 7.6k test examples. TREC-6 (Li

and Roth, 2002) is a question classification dataset labeled with six coarse classes. It contains 5.8k training examples and 500 test examples Arxiv is a collection of research paper. We randomly select 3k articles alonst 4 classes, and kept title and abstract. Final dataset contains $2,250$ training examples, 750 test examples.

5.2 Baselines

Model	Accuracy
Cer et al. (2018)	98.07
Chen et al. (2022)	97.40
Sun et al. (2020)	96.80
Our best results	97.12

TAB. 1: Baseline results for TREC-6.

Model	Accuracy
Yang et al. (2020)	95.55
Sun et al. (2020)	95.20
Sachan et al. (2019)	95.05
Our best results	94.26

TAB. 2: Baseline results for AG-News.

Tables 1 and 2 report baseline results for TREC-6 and AG-News respectively. Because we have selected a particular subset of Arxiv, there is no baseline to be found in the literature.

5.3 Training

Hyperparameters for the architectures were tuned through a grid-search-like approach. We tried different values for the following parameters:

- **Activation function**: ReLU, LearkyReLU, TanH and GELU.
- **Global pooling function**: mean and max.
- **Hidden layer size**: $1,536$ or $2,304$ for all models, except for RGCN, where a size of 256 was utilized.

Training was done using Adam optimizer in conjunction with a cross-entropy loss function. Learning rate was set to $1e^{-5}$ for all models except GCN and RGCN for which it was adjusted to $1e^{-2}$. Each model underwent training for 50 epochs, with the exception of GCN and RGCN, both of which were trained for an extended duration of 200 epochs. The batch size was held constant at 64 throughout all training. Only the checkpoint with the best accuracy was kept.

5.4 Results

In order to evaluate the different architectures, we report precision, recall and F1 score with macro averaging. Tables 3 to 8 show results for each dataset and embedder. Table 9 shows average of best results.

6 Discussion

Tables 3, 5 and 7 show the performance of different GNN models–initialized with DistilBERT pre-trained embedding– on Arxiv, TREC-6, and AG-News datasets, respectivley. The

Model	Precision	Recall	F1	Accuracy
GAT	0.9705	**0.9705**	0.9704	**0.9705**
GCGAT	0.9691	0.9693	0.9691	0.9693
GCN	**0.9710**	0.9704	0.9704	0.9704
GIN	0.9691	0.9693	0.9692	0.9693
RGCN	0.9355	0.9340	0.9344	0.9340
TransCN	0.9708	**0.9705**	**0.9705**	**0.9705**

TAB. 3: Arxiv with DistilBERT.

Model	Precision	Recall	F1	Accuracy
GAT	0.9504	0.9497	0.9492	0.9497
GCGAT	**0.9545**	**0.9537**	**0.9532**	0.9537
GCN	0.9522	0.9522	0.9518	**0.9522**
GIN	0.9511	0.9509	0.9507	0.9509
RGCN	0.8343	0.8281	0.8294	0.8281
TransCN	0.9542	0.9522	0.9521	**0.9522**

TAB. 4: Arxiv with BERT-large.

Model	Precision	Recall	F1	Accuracy
GAT	0.7782	0.7836	0.7790	0.7836
GCGAT	0.9582	0.9544	0.9549	0.9544
GCN	0.9406	0.9514	0.9446	0.9514
GIN	0.7698	0.7728	0.7704	0.7728
RGCN	0.8288	0.6921	0.7339	0.6921
TransCN	**0.9615**	**0.9600**	**0.9605**	**0.9600**

TAB. 5: TREC-6 with DistilBERT.

Model	Precision	Recall	F1	Accuracy
GAT	0.7609	0.7766	0.7680	0.7766
GCGAT	0.9291	0.9553	0.9401	0.9553
GCN	0.9237	0.9462	0.9313	0.9462
GIN	0.7567	0.7716	0.7637	0.7716
RGCN	0.8131	0.6784	0.7134	0.6784
TransCN	**0.9601**	**0.9712**	**0.9648**	**0.9712**

TAB. 6: TREC-6 with BERT-large.

Model	Precision	Recall	F1	Accuracy
GAT	**0.9427**	**0.9426**	**0.9426**	**0.9426**
GCGAT	0.9406	0.9407	0.9406	0.9407
GCN	0.9254	0.9257	0.9254	0.9257
GIN	0.9348	0.9342	0.9341	0.9342
RGCN	0.9263	0.9263	0.9261	0.9263
TransCN	0.9408	0.9407	0.9406	0.9407

TAB. 7: AG-News with DistilBERT.

Model	Precision	Recall	F1	Accuracy
GAT	**0.9379**	**0.9379**	**0.9379**	**0.9379**
GCGAT	0.9373	0.9374	0.9373	0.9374
GCN	0.9214	0.9217	0.9215	0.9217
GIN	0.9296	0.9293	0.9292	0.9293
RGCN	0.9259	0.9254	0.9254	0.9254
TransCN	0.9360	0.9359	0.9359	0.9359

TAB. 8: AG-News with BERT-large.

tables 4, 6 and 8 illustrate the identical datasets, but initialized with BERT-large model.

Among GNN models initialized with DistilBERT and trained on the Arxiv dataset, TransCN and GAT have almost the best scores with 0.9705 for F1 and Recall. The satisfying fact is that the delta range of scores is around 4. Although DistilBERT has around 4.2 times fewer parameters comparing to BERT-large, its performance is 3 points above. The best precision score 0.9710 belongs to simple GCN model with DistilBERT-initialized embeddings.

The best GNN model built for the TREC-6 dataset is TransCN with around 0.96 and 0.97 accuracy score for DistilBERT and BERT-large models, respectively. Almost the same result can be observed for AG-News dataset. The TransCN and GAT models have the best results with around 0.94 and 0.93 accuracy score for DistilBERT and BERT-large models, respectively. One important observation is that the DistilBERT-based TransCN model consistently outperforms the other models. The TransCN model generalizes well based on the precision, recall, F1 and accuracy metrics.

We have initialized the models with different two pre-trained BERT model, namely DistilBERT and BERT-large to explore their effect on the model performance. Table 9 summarizes average accuracy for each embedding type and dataset across all models evaluated in our experiments. Unexpectedly, DistilBERT superseeds the performance of BERT-large model. We normaly expect to observe better performance for a model like BERT-large which has 4.2 times more parameters comparing to DistilBERT. This expectation was not held.

Dataset	Embedder	Precision	Recall	F1	Accuracy
AG-News	DistilBERT	**0.9351**	**0.9350**	**0.9349**	**0.9350**
AG-News	BERT-large	0.9324	0.9324	0.9324	0.9324
TREC-6	DistilBERT	**0.8729**	**0.8524**	**0.8572**	**0.8524**
TREC-6	BERT-large	0.8573	0.8499	0.8469	0.8499
Arxiv	DistilBERT	**0.9643**	**0.9640**	**0.9640**	**0.9640**
Arxiv	BERT-large	0.9328	0.9311	0.9311	0.9311

TAB. 9: Averages results for DistilBERT and BERT-large.

For TREC-6 and AG-News datasets, the best sequence-based models accuracy score are 98.07 and 95.55, respectivly. Our best models are just 0.95 and 1.29 points below the best sequence models for TREC-6 and AG-News datasets. This fact endorses the learnability of our models. In particular the models TransCN and GAT are very competive with sequenced-based models while the models introduced in this research take the dependency graphs as input. They enjoy some graph-based properties such as interpretable representations of the relationships between words, incorporation of domain-specific knowledge and custom feature engineering.

7 Conclusion and Future Work

In this research, we investigated different Graph Neural Network architectures for document classification task. Contrary to standard practice, we used dependency graphs, obtained from pure text documents, as input to our models. The choice of dependency graph makes possible the straighforward integration of the efficient, and yet interpretable graphs. Transforming sequential text into relational graph and using graph as input to GNN, allowed us to reformulate document classification as graph classification and perform the task with GNN. We achieved competitve results using three datasets, namely Arxiv, TREC-6 and AG-News and compared them with state-of-the-art sequential models.

There are few areas that demands further investigations: (i) For the syntactic dependency structures of text, we did not explore the effect of parsing quality on results. This area is crucial as different parsers may slightly change the result, especially when the state-of-the-art models are very competitive. (ii) To perform coreference resolution and add coreference chain relations. These relations can improve representations by connecting corefering mentions and facilitating the flow of information between sentences. (iii) DistilBERT and BERT-large were used as embedders, but they were not tuned during training. As shown in Lin et al. (2021), tuning BERT embedders along with GNN models will improve the results. (iv) We have used the text to graph transformation in our experiment by adopting syntactic dependency parsing. We can extend our experiments by using different semantic-based representations such as AMR (Banarescu et al., 2013) or MR4AP (Giordano and Lopez, 2023).

References

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider (2013). Abstract Meaning Representation for sembanking. In A. Pareja-Lora, M. Liakata, and S. Dipper (Eds.), *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, pp. 178–186. Association for Computational Linguistics.

Brody, S., U. Alon, and E. Yahav (2022). How attentive are graph attention networks?

Busbridge, D., D. Sherburn, P. Cavallo, and N. Y. Hammerla (2019). Relational graph attention networks.

Cer, D., Y. Yang, S. yi Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil (2018). Universal sentence encoder.

Chen, Q., R. Zhang, Y. Zheng, and Y. Mao (2022). Dual contrastive learning: Text classification via label-aware data augmentation.

Conneau, A., H. Schwenk, L. Barrault, and Y. Lecun (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine learning 20*(3), 273–297.

Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805*.

Giordano, B. and C. Lopez (2023). MR4AP: Meaning representation for application purposes. In J. Bonn and N. Xue (Eds.), *Proceedings of the Fourth International Workshop on Designing Meaning Representations*, Nancy, France, pp. 110–121. Association for Computational Linguistics.

Han, S. C., Z. Yuan, K. Wang, S. Long, and J. Poon (2022). Understanding graph convolutional networks for text classification. *arXiv preprint arXiv:2203.16060*.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*, 1735–80.

Hua, S., X. Li, Y. Jing, and Q. Liu (2022). A semantic hierarchical graph neural network for text classification. *arXiv preprint arXiv:2209.07031*.

Ioffe, S. and C. Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167*.

Johnson, R. and T. Zhang (2016). Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*, pp. 526–534. PMLR.

Jurafsky, D. and J. H. Martin (2009). *Speech and language processing* (2. ed., [Pearson International Edition] ed.). Prentice Hall series in artificial intelligence. London [u.a.]: Prentice Hall, Pearson Education International.

Kipf, T. N. and M. Welling (2016). Semi-supervised classification with graph convolutional networks. *CoRR abs/1609.02907*.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computa-*

tion 1, 541–551.

Li, X. and D. Roth (2002). Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Liang, Y., Q. Wang, K. Xiong, X. Zheng, Z. Yu, and D. Zeng (2021). Robust detection of malicious urls with self-paced wide & deep learning. *IEEE Transactions on Dependable and Secure Computing 19*(2), 717–730.

Lin, Y., Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu (2021). Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Morris, C., M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe (2021). Weisfeiler and leman go neural: Higher-order graph neural networks.

Ravi, K. and V. Ravi (2015). A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-based systems 89*, 14–46.

Sachan, D. S., M. Zaheer, and R. Salakhutdinov (2019). Revisiting lstm networks for semi-supervised text classification via mixed objective function. *Proceedings of the AAAI Conference on Artificial Intelligence 33*(01), 6940–6948.

Sanh, V., L. Debut, J. Chaumond, and T. Wolf (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Schlichtkrull, M., T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling (2017). Modeling relational data with graph convolutional networks.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena 404*, 132306.

Shi, Y., Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun (2021). Masked label prediction: Unified message passing model for semi-supervised classification.

Sun, C., X. Qiu, Y. Xu, and X. Huang (2020). How to fine-tune bert for text classification?

Tan, K. L., C. P. Lee, and K. M. Lim (2023). A survey of sentiment analysis: Approaches, datasets, and future research. *Applied Sciences 13*(7), 4550.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio (2018). Graph Attention Networks. arXiv:1710.10903 [cs, stat].

Wang, K., Y. Ding, and S. C. Han (2023). Graph neural networks for text classification: A survey. *arXiv preprint arXiv:2304.11534*.

Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xia, L., D. Luo, C. Zhang, and Z. Wu (2019). A survey of topic models in text classification. In *2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 244–250. IEEE.

Xu, K., W. Hu, J. Leskovec, and S. Jegelka (2019). How powerful are graph neural networks?

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le (2020). Xlnet: Generalized autoregressive pretraining for language understanding.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems 32*.

Yao, L., C. Mao, and Y. Luo (2019). Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Volume 33, pp. 7370–7377.

Zhang, X., J. Zhao, and Y. LeCun (2016). Character-level convolutional networks for text classification.

Résumé

L'approche classique pour la classification des documents consiste à traiter le texte comme une donnée séquentielle et à utiliser des architectures de type transformer pour la classification. Cependant, cette représentation rends l'identification et l'exploitation des relations implicites dans le texte difficile. Cet article étudie l'utilisation de graphes de dépendance syntaxique comme représentation relationnelle alternative, reformulant ainsi la classification de documents en classification de graphes. Ces graphes encodent la structure grammaticale, reliant les mots en fonction de leurs relations et de leurs rôles dans la phrase. Cette représentation explicite facilite la compréhension des structures syntaxiques et sémantiques sous-jacentes. Les graphes de dépendance, interprétables, sont ensuite traités par des Graph Neural Network (GNN), un type d'architecture qui excelle à capturer les relations complexes entre les entités d'un graphe en propageant l'information à travers sa structure. Pour tester cette approche, la structure grammaticale du texte est calculée à l'aide d'un analyseur en dépendances et les représentations des noeuds initialisées à l'aide de plongements de mots. Nous expérimentons avec six architectures distinctes de GNN comme classifieurs. Les résultats sont ensuite évalués sur trois jeux de données AG-News, TREC-6 et Arxiv, puis comparés à l'état de l'art. Ces expériences démontrent que la représentation des textes par leur graphe de dépendance syntaxique et l'utilisation de GNN pour la classification permettent d'obtenir des résultats très compétitifs par rapport aux modèles classiques basés sur les séquences.

Extraction d'acronymes torturés dans la littérature scientifique

Alexandre Clausse*, Guillaume Cabanac*,**, Pascal Cuxac*** et Cyril Labbé****

*Université Toulouse III – Paul Sabatier, IRIT UMR 5505 CNRS, Toulouse, France
{alexandre.clausse, guillaume.cabanac}@irit.fr

**Institut Universitaire de France (IUF), Paris, France

***INIST – CNRS, UAR76, Vandœuvre-lès-Nancy, France
pascal.cuxac@inist.fr

****Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble, France
cyril.labbe@univ-grenoble-alpes.fr

Résumé. Les politiques publiques poussent les chercheurs à publier le plus régulièrement possible des articles scientifiques dans des revues réputées. Cette pression amène une minorité de personnes peu scrupuleuses à frauder, en utilisant notamment des phrases torturées afin de déguiser des plagiats. Il arrive que ce type de contenu ne soit pas détecté lors de l'évaluation par les pairs, amenant à sa publication. Dans l'optique de dépolluer la littérature scientifique, nous proposons une tâche d'extraction d'acronymes torturés permettant la détection de publications frauduleuses. Un benchmark est effectué grâce à un corpus de 75 articles scientifiques torturés et en open access, avec une chaîne de traitements permettant l'extraction et de classification d'acronymes. La tâche d'extraction d'acronymes torturés obtient une F-mesure de 0,74, qui pourra être améliorée par l'enrichissement du corpus. Nous avons constitué une ligne de référence *(baseline)* à laquelle toute personne désireuse d'améliorer la performance de cette tâche pourra se référer.

Introduction

Les politiques publiques actuelles exercent une constante pression sur les chercheurs, en les poussant à publier le plus régulièrement possible dans des revues réputées (à fort facteur d'impact), afin d'obtenir de meilleures performances dans les classements internationaux. Communément appelée « publier ou périr », cette situation est susceptible d'amener à un manque de considération pour la rigueur impliquée dans les travaux de recherche scientifique par une minorité de personnes peu scrupuleuses, ayant recours à de la fabrication, de la falsification et au plagiat. Ces problèmes ont été décrits dans un ouvrage édité par (Biagioli et Lippman, 2020). Le plagiat peut être déguisé par l'utilisation de phrases torturées, définies par (Cabanac et al., 2021), correspondant à la déformation d'un phraséoterme (terme scientifique considéré comme établi dans une discipline associée), par la génération de synonymes vidant ce dernier

de toute signification, principalement en utilisant des outils de paraphrase (*spinners*) tels que SpinBot [1]. Par exemple, le terme informatique « *artificial intelligence* » peut être transformé en « *man-made consciousness* », ce qui n'a aucune signification dans un tel domaine métier. De plus, un texte comprenant des phrases torturées peut échapper à la vigilance d'un comité de relecture, amenant à sa publication, et par conséquent à la pollution de la littérature scientifique. Cela met aussi en doute leur probité quant à la tenue des expériences et la rédaction sincère des résultats.

Étant donné ce contexte, le *Problematic Paper Screener* [2] (PPS) a été développé par (Cabanac et al., 2022) afin de permettre la réévaluation collaborative, par le biais de PubPeer [3] (créé et administré par (Barbour et Stell, 2020)), d'articles scientifiques signalés comme étant suspects. Parmi un ensemble de détecteurs (permettant par exemple de détecter des articles torturés ou générés par SciGen [4]), 13 000 articles ont été considérés comme contenant suffisamment de phrases torturées pour que leur rigueur scientifique soit remise en cause. La détection de tels contenus a été explorée par (Lay et al., 2022) par la construction d'un corpus de 2 772 paragraphes contenant ou non des phrases torturées. Ils ont utilisé des algorithmes de classification binaire (forêt d'arbres décisionnels, perceptron, transformeur) sur les 5-grammes et paragraphes. Leur meilleur modèle a obtenu une F-mesure de 0,99 (classe « torturée ») et 0,92 (classe « non torturée »). Cependant cette approche est limitée par la possible présence des mêmes phrases torturées dans les données d'entraînement et de test, car celles-ci ont été séparées de façon aléatoire. (Kashnitsky et al., 2022) ont proposé une tâche partagée avec un corpus de plus de 26 000 documents extraits depuis Scopus, afin de détecter différentes formes de textes générés (par exemple par résumé automatique ou par l'utilisation de *spinners*). Ils ont utilisé la régression logistique ainsi qu'un modèle SciBERT ((Beltagy et al., 2019)) ajusté comme ligne de base, en ayant respectivement obtenu une F-mesure de 0,82 et 0,98. (Becker et al., 2023) ont étendu cette exploration par un corpus totalisant plus de 140 000 paires de phrases et paragraphes paraphrasés par un humain ou une machine. Ils ont utilisé des modèles de plongement lexical tels que GloVe ((Pennington et al., 2014)) et FastText ((Joulin et al., 2016)), ainsi que des modèles de langage tels que BERT ((Devlin et al., 2019)) et T5 ((Raffel et al., 2020)), leur meilleur modèle a obtenu une F-mesure de 0,95.

Cet article présente la problématique de la thèse que le premier auteur vient de débuter, par une approche orientée document et focalisée sur les acronymes torturés (c'est-à-dire des phraséotermes sous forme d'acronymes qui ont été torturés, par exemple « *fake neural system (ANN)* » est une version torturée de « *artificial neural network* »). Nous proposons un algorithme auquel est associé un benchmark, par l'utilisation d'un corpus composé d'articles scientifiques en accès libre. Sont adjoints une chaîne de traitements permettant l'extraction et la classification d'acronymes, ainsi que des métriques d'évaluation des résultats obtenus.

Constitution d'une collection de test

À partir du PPS (détecteur « *tortured* »), nous avons collecté un corpus de 75 articles scientifiques en accès libre, dans le domaine de l'ingénierie (dont les disciplines sont représentés

1. https://spinbot.com
2. https://irit.fr/~Guillaume.Cabanac/problematic-paper-screener
3. https://pubpeer.com
4. https://pdos.csail.mit.edu/archive/scigen/

en Figure 1), publiés entre 2015 et 2023 (dont la distribution des années de publication est représentée en Figure 2). Nous avons effectué une recherche par mot-clé dans le champs « *venue* », c'est-à-dire le titre de la revue ou conférence dans laquelle les articles collectés ont été publiés, en utilisant le nom de 10 disciplines en anglais (par exemple, les articles du domaine de la physique ont été collectés en effectuant une recherche avec le terme « *physics* » présent dans le titre de la revue associée, en vérifiant qu'ils sont bien en accès libre). Malgré cela, notre corpus comprend 37 articles pluridisciplinaires, dont la discipline commune est l'informatique. Puis nous avons effectué une analyse exploratoire des données afin d'en extraire les principales caractéristiques.

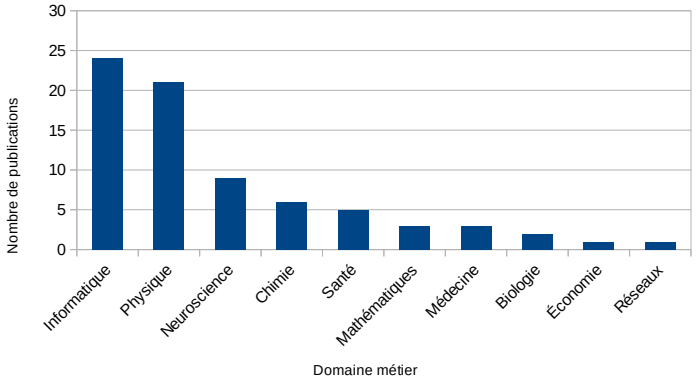

FIG. 1 – *Distribution des domaines métiers dans le corpus. Plus de la moitié de celui-ci contient des articles des domaines de l'informatique (24 articles soit 32 % du corpus) et de la physique (21 articles soit 28 % du corpus), les autres domaines étant peu représentés (entre 1 et 9 articles soit entre 1,3 et 12 % du corpus).*

Le premier auteur a manuellement extrait et annoté 995 acronymes distincts, établissant ainsi un *silver standard*, accessible publiquement (voir la section disponibilité des données). Nous avons récupéré l'ensemble des acronymes torturés (366 acronymes soit 36,8 % du corpus), plusieurs acronymes comportant des fautes de frappe (par exemple « *Tru_-Positive (TP)* ») ont été étiquetés comme suspects (46 acronymes soit 4,6 % du corpus), les autres acronymes ont été annotés comme non torturés (indiqués comme *genuine*). Puis nous avons extrait un ensemble de caractéristiques (décrites dans le Tableau 1 et dont la distribution est décrite en Figure 3) à prendre en compte dans le développement d'un algorithme d'extraction et de classification.

Parmi ces caractéristiques, nous avons remarqué quelques subtilités. La différence entre le nombre d'initiales d'un phraséoterme et son acronyme associé est principalement due à la présence de mots de liaison. Dans certains cas, un mot supplémentaire est présent entre l'acronyme et sa forme développée. Certains acronymes sont déclinés dans leur forme plurielle (par exemple « *CNN* » et « *CNNs* »). D'autres acronymes contiennent des caractères spéciaux tels que des parenthèses (cela concerne principalement les composés chimiques tels que « *poly(ethylene glycol) (PEG)* »), des points d'interrogation (cela concerne les erreurs d'encodage des lettres grecques, telles que « *Tumor Necrosis Factor Alpha (TNF-?)* », liées à la

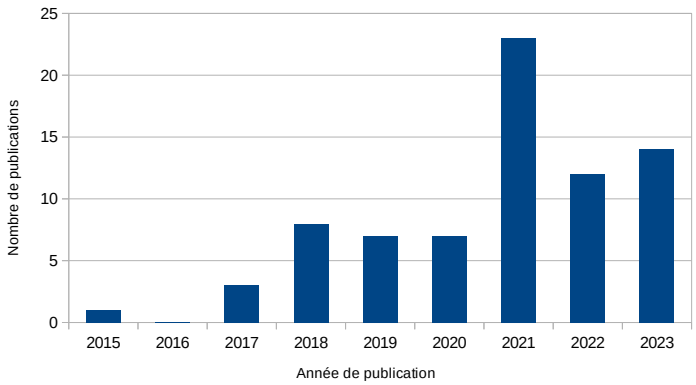

FIG. 2 – *Distribution des années de publication des articles du corpus. Nous pouvons observer une augmentation significative de publications contenant des phrases torturées à partir de 2018, avec un pic en 2021. Ce pic correspond à un grand nombre d'articles publiés en 2021 en accès libre ayant pu être récupérés depuis une même source.*

conversion des fichiers PDF bruts en texte, voir la Figure 4), des tirets et des points. Une quantité négligeable d'acronymes est rédigée dans une forme alternative (par exemple « *(PLSR) Partial Least Squares Regression* » où l'acronyme en parenthèses est positionné avant sa forme développée, et « *CNN (Convolutional Neural Network)* » où c'est la forme développée qui se trouve entre parenthèses au lieu de l'acronyme). Nous évaluons trois tâches distinctes : extraction d'acronymes, classification d'acronymes, extraction d'acronymes torturés. Pour cela, nous calculons les scores de rappel, précision, F-mesure et — dans la mesure du possible — le coefficient de corrélation de Matthews (*MCC*) ainsi que l'aire sous la courbe de la fonction d'efficacité du récepteur (*ROC AUC*). Ces métriques d'évaluation sont calculées différemment d'une tâche à l'autre, de par la nature de celles-ci. En effet, nous comptons différemment les vrai-positifs (VP), vrai-négatifs (VN), faux-positifs (FP) et faux-négatifs (FN) pour la tâche d'extraction (recherche d'information) et la tâche de classification.

Concernant l'évaluation de la tâche d'extraction, les acronymes correctement extraits depuis le *silver* sont considérés comme vrai-positifs (VP), les acronymes extraits qui n'apparaissent pas dans le *silver* sont considérés comme faux-positifs (FP), et les acronymes qui n'ont pas été extraits depuis le *silver* sont considérés comme faux-négatifs (FN). Nous ne calculons pas de vrai-négatifs (VN) dans la mesure où ils seraient définis comme des acronymes absents du *silver* et non extraits, or nous n'avons aucun moyen d'en obtenir. Concernant l'évaluation de la tâche de classification, les acronymes torturés sont considérés comme positifs, et ceux non torturés comme négatifs, les vrai- et faux-positifs / négatifs sont définis en comparant l'étiquetage des acronymes du *silver* et de ceux classifiés. Enfin, concernant l'évaluation de la tâche d'extraction des acronymes torturés, les VP correspondent aux acronymes torturés du *silver* étiquetés en tant que tel par l'algorithme. Les FP correspondent à tous les acronymes non torturés du *silver* qui n'ont pas été extraits ou qui ont été étiquetés comme torturés mais aussi aux acronymes étiquetés comme torturés mais qui n'apparaissent pas dans le *silver*. Les FN correspondent à tous les acronymes torturés du *silver* qui n'ont pas été extraits ou qui ont été étiquetés comme non torturés mais aussi aux acronymes étiquetés comme non torturés mais

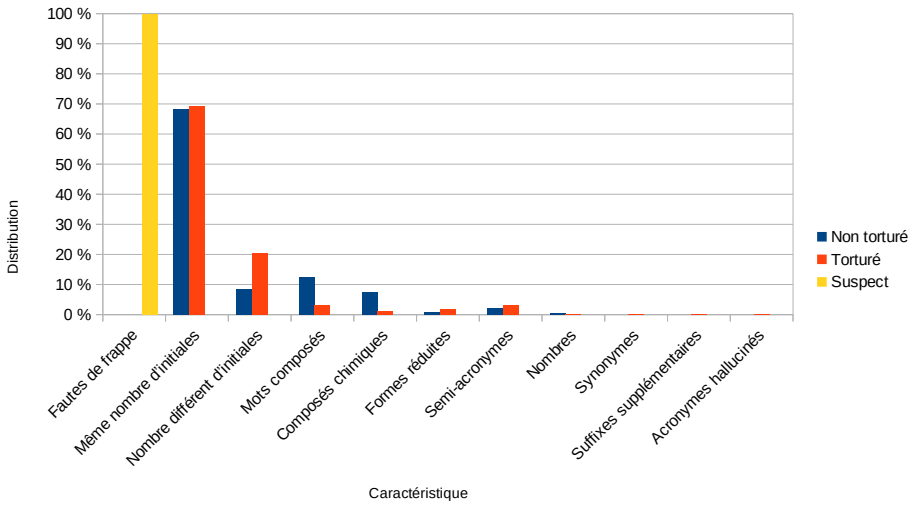

FIG. 3 – *Distribution des caractéristiques des acronymes dans chaque classe associée. La distribution des acronymes torturés comportant un nombre différent d'initiales est plus importante que pour les acronymes non torturés, cela peut être expliqué par la génération de synonymes composés de plusieurs mots. Au contraire, il y a moins d'acronymes torturés comprenant des mots composés et des composés chimiques, car il est plus difficile de trouver des synonymes à ceux-ci.*

qui n'apparaissent pas dans le *silver*. Nous ne calculons pas de VN dans la mesure où ils seraient définis comme des acronymes non torturés dans le *silver* et étiquetés comme tel, ce qui est en dehors du contexte de cette tâche.

Il est important de préciser que les acronymes considérés comme suspects dans le *silver* ont été étiquetés comme n'étant pas torturés (car ils correspondent à des fautes de frappe, c'est-à-dire à des erreurs non intentionnelles) dans la phase d'évaluation.

Extraction et classification d'acronymes torturés

Étant donnés le corpus et ses caractéristiques associées, nous avons élaboré une chaîne de traitements permettant l'extraction et la classification d'acronymes, divisé en quatre tâches principales (représentées en Figure 3). Celui-ci sert de ligne de base pour cette tâche.

La tâche (I) consiste à extraire le contenu des articles au format PDF en TEI-XML par l'utilisation de *Grobid*[5], puis à en extraire le contenu textuel brut par l'utilisation de *BeautifulSoup4*[6]. Les tâches (II) et (III) sont étroitement liées car, pour chaque fichier texte précédemment généré, les fautes de frappe telles que les parenthèses dupliquées sont corrigées, puis les caractères spéciaux (c'est-à-dire les caractères de liaison et de contrôle) sont supprimés ; ce

5. `https://grobid.readthedocs.io/en/latest/`
6. `https://crummy.com/software/BeautifulSoup/`

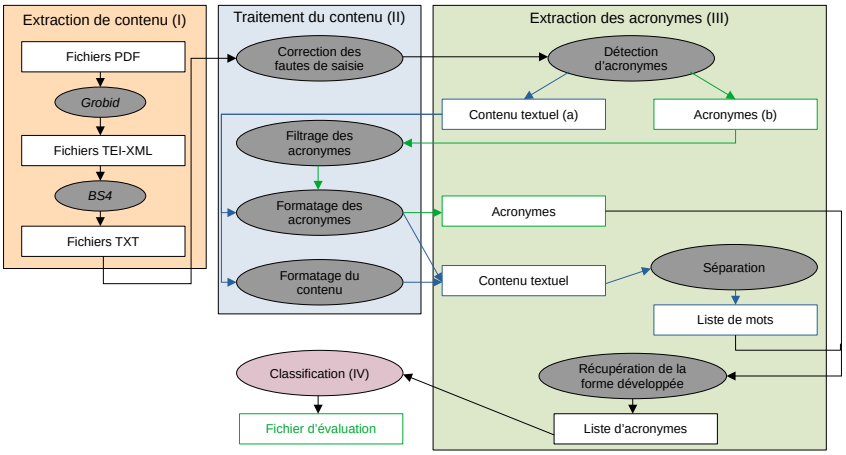

FIG. 4 – *Diagramme de traitement des acronymes. Le fonctionnement de la chaîne de traitements est divisé en quatre étapes, allant des l'extraction des données brutes depuis les fichiers PDF des articles à la création du fichier d'évaluation, en passant par l'extraction et la classification des acronymes. Les données textuelles brutes sont nettoyées en parallèle avec les acronymes extraits afin d'uniformiser les résultats obtenus.*

sont des tâches d'uniformisation de contenu. Une expression régulière est utilisée afin de détecter les acronymes, définis comme une suite de deux lettres capitales (ou plus), non séparées par un espace, le tout contenu entre parenthèses. Ces acronymes peuvent aussi contenir des points, des points d'interrogation, des tirets et esperluettes. À la fin de cette étape, le contenu textuel brut (III.a) et les acronymes (III.b) doivent être nettoyés. Ainsi, les acronymes sont filtrés afin de supprimer les faux-positifs tels que les références, et le texte brut est formaté selon qu'il s'agisse d'un acronyme ou non (par exemple les formes plurielles sont supprimées dans le cas d'un acronyme) puis séparé en liste de mots. La tâche (IV) consiste à classifier les acronymes, en suivant une séquence de règles de filtrage décrites dans l'algorithme 1, basées sur les caractéristiques communes aux acronymes torturés ou non.

Étant donné un acronyme extrait, une correspondance est effectuée sur les initiales (par exemple « *convolutional neural network (CNN)* », étape 1 de l'algorithme), en tenant également compte des mots de liaison (par exemple « *Moroccan agency of press (MAP)* », étape 2 de l'algorithme). Les mêmes comparaisons sont effectuées avec un décalage d'un mot (par exemple « *deep neural network models (DNN)* » et « *centers for disease control and prevention (CDC)* », étapes 3 et 4 de l'algorithme). Une correspondance est aussi effectuée sur les mots composés (par exemple « *chitosan (CS)* », « *electromyogram (EMG)* » et « *rectified linear unit (ReLU)* », étapes 5 et 6 de l'algorithme) et les formes réduites (par exemple « *residual network (ResNet)* », étape 6 de l'algorithme). Si aucune de ces correspondances n'est possible alors l'acronyme est considéré comme étant torturé. Enfin, un fichier d'évaluation est construit, contenant pour chaque acronyme, son étiquette et nombre d'occurrences tels qu'annotés dans le *silver* et tels qu'extraits par l'algorithme. Ce dernier est indépendant de toute liste d'acronymes de référence d'un quelconque domaine métier.

Données : un acronyme de longueur n précédé de $n + 1$ mots
Résultat : un triplet (forme développée, acronyme, étiquette)
$A \leftarrow$ l'acronyme;
$M \leftarrow$ l'acronyme découpé selon ses lettres capitales;
$P \leftarrow$ la liste des $n + 1$ mots avant l'acronyme;
$S \leftarrow$ la liste des $n + 1$ mots avant l'acronyme, sans les mots de liaison;
`// Étape 1`
si *les initiales de sousChaine(P, 1, longueur(P) - 1) = A* **alors**
  renvoyer (sousChaine(P, 1, longueur(P) - 1), A, non torturé);
fin
`// Étape 2`
si *les initiales de sousChaine(S, 1, longueur(S) - 1) = A* **alors**
  renvoyer (sousChaine(S, 1, longueur(S) - 1), A, non torturé);
fin
`// Étape 3`
si *les initiales de sousChaine(P, 2, longueur(P)) = A* **alors**
  renvoyer (sousChaine(P, 2, longueur(P)), A, non torturé);
fin
`// Étape 4`
si *les initiales de sousChaine(S, 2, longueur(S)) = A* **alors**
  renvoyer (sousChaine(S, 2, longueur(S)), A, non torturé);
fin
`// Étape 5`
pour tout *mot $p \in P$* **faire**
  si *les lettres de A se suivent dans p* **alors**
    renvoyer (p, A, non torturé);
  fin
fin
`// Étape 6`
si *les composantes de M sont alignées avec les mots de sousChaine(P, 2, longueur(P))* **alors**
  renvoyer (sousChaine(P, 2, longueur(P)), A) non torturé;
fin
renvoyer (P, A, torturé);

Algorithme 1 : Algorithme de classification d'un acronyme.

Benchmark

La tâche (III) obtient un rappel de 0,99, une précision de 0,89 et une F-mesure de 0,94. Nous avons obtenu pour la tâche (IV) un rappel de 0,95, une précision de 0,79, une F-mesure de 0,86 et un MCC de 0,78. Enfin, nous avons obtenu pour la tâche d'extraction d'acronymes torturés un rappel de 0,93, une précision de 0,62 et une F-mesure de 0,74. En complément, nous avons calculé les matrices de confusion de ces trois tâches (présentées en Figure 5) ainsi que la courbe ROC pour la tâche de classification des acronymes (décrite en Figure 6).

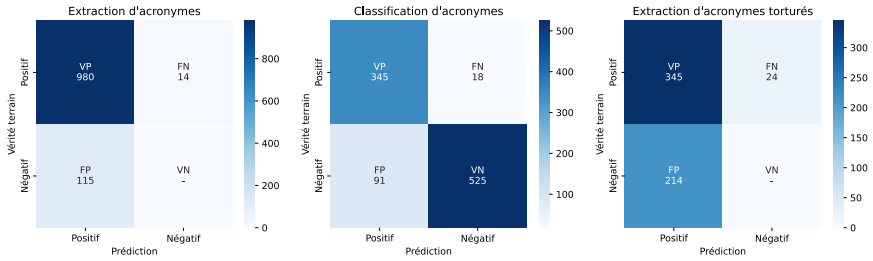

FIG. 5 – *Matrices de confusion pour les trois tâches (extraction, classification, extraction d'acronymes torturés). Nous constatons l'absence de VN dans la première et troisième tâche car ceux-ci n'ont pas été définis ; il y a aussi beaucoup de FP, peu importe la tâche.*

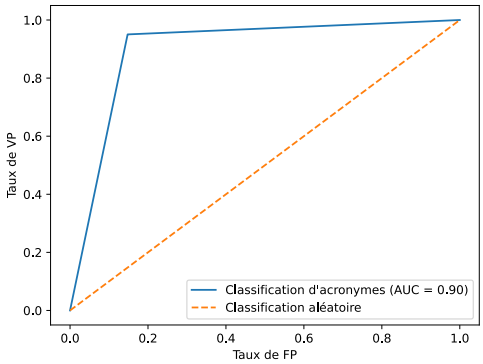

FIG. 6 – *Courbe ROC de la tâche de classification d'acronymes. Nous constatons une aire sous la courbe à 0,90, indiquant une bonne capacité discriminatoire.*

Étant donnés les résultats obtenus, l'évaluation de la tâche d'extraction d'acronymes semble satisfaisante, cependant ce n'est pas le cas des deux autres tâches. En effet, malgré un rappel supérieur à 0,9, les scores de précision indiquent un nombre trop important de FP, ces tâches ont donc tendance à surestimer les acronymes à traiter. Ceci pouvant être expliqué par les limites de notre méthode agnostique (basée sur les caractéristiques connues et communes aux acronymes torturés ou non) mais aussi par un nombre limité d'acronymes distincts dans notre corpus. Ce

dernier comprend un biais de représentativité au niveau des disciplines, avec une forte présence de termes d'informatique. Il devrait donc être enrichi, en effectuant des recherches en dehors des seuls articles dont le domaine métier est explicitement mentionné, mais aussi en y adjoignant d'autres corpus (par exemple celui proposé par (Lay et al., 2022)). Cela permettrait d'étudier davantage la sensibilité de notre algorithme d'extraction et de classification aux diverses subtilités induites par les formes variées d'acronymes. Nous avons constaté la présence d'un acronyme halluciné (« *bolster vector machine (BVM)* », une version torturée et non existante du « *support vector machine (SVM)* »), étiqueté comme étant légitime par notre algorithme. Il s'agit d'un cas anecdotique, cependant il serait intéressant d'étudier le contexte dans lequel a pu apparaître un tel acronyme. Aussi, le *silver* ayant été établi par une personne non polymathe, il se peut que des erreurs d'annotations soient présentes. Certains choix d'annotations peuvent être discutés, par exemple la forme développée du terme « *Bidirectional LSTM (BiLSTM)* » comporte à la fois un acronyme et un mot composé, qui sont deux caractéristiques distinctes.

Conclusion

Pour lutter contre la pollution de la littérature scientifique, nous avons défini une tâche d'extraction d'acronymes torturés : une forme spécifique de phrases torturées. Pour cela, nous avons constitué une collection de test, composée d'articles scientifiques en accès libre, dont nous avons extrait et annoté les acronymes, puis nous avons établi diverses métriques permettant d'évaluer cette tâche. Après avoir proposé une première méthode agnostique, nous avons obtenu une F-mesure de 0,74. Cependant, notre algorithme a tendance à trop surestimer les acronymes, amenant à un nombre trop important de faux-positifs, et par conséquent à une précision améliorable. De futurs travaux devraient être focalisés sur la production d'algorithmes plus performants, en incluant les décisions des experts dans la prise de décision algorithmique. Sur le long terme, il serait intéressant de mettre à disposition cette ligne de base au travers d'un défi TextMine [7] ou Kaggle [8]. Ces travaux pourraient aussi servir aux maisons d'édition, afin de filtrer leur chaîne éditoriale en amont, par l'identification et le signalement de potentiels problèmes aux éditeurs en charge d'organiser l'évaluation par les pairs. Nous avons constitué cette ligne de base afin que toute personne, désireuse de travailler sur l'extraction d'acronymes torturés, puisse s'y référer.

Disponibilité des données

Les données de test supportant cette étude sont disponibles sur Zenodo [9].

7. https://textmine.sciencesconf.org/
8. https://kaggle.com/
9. https://doi.org/10.5281/zenodo.10492230

Remerciements

Le projet NanoBubbles bénéficie d'un financement Synergy grant du Conseil de la recherche européen (European Research Council, ERC), dans le cadre du programme Horizon 2020 de l'Union Européenne, numéro de contrat 951393.

Références

Pennington, J., Socher, R. et Manning, C. (2014). *GloVe: Global Vectors for Word Representation*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'2014), ACL, Octobre 2014, p. 1532–1543 (DOI : https://doi.org/10.3115/v1/D14-1162).

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H et Mikolov, T. (2016). *FastText.zip: Compressing text classification models*. arXiv (DOI : https://doi.org/10.48550/arXiv.1612.03651)

Devlin, J., Chang, M.-W., Kenton, L. et Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv (DOI : https://doi.org/10.48550/arXiv.1810.04805).

Beltagy, I., LO, K. et Cohan, A. (2019). *SciBERT: A Pretrained Language Model for Scientific Text*. arXiv (DOI : https://doi.org/10.48550/arXiv.1903.10676).

Raffel, C., Shazerr, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W et Liu, P.-J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. Journal of Machine Learning Research, vol. 21 (140), 2020, p. 1–67 (URL : https://jmlr.org/papers/v21/20-074.html).

Biagioli, M. et Lippman, A. (2020). *Gaming the Metrics: Misconduct and Manipulation in Academic Research*. MIT Press, 2020, 307 p. (DOI : https://doi.org/10.7551/mitpress/11087.001.0001).

Barbour, B. et Stell, B.-M. (2020). *PubPeer: Scientific Assessment Without Metrics*. Gaming the Metrics : Misconduct and Manipulation in Academic Research, MIT Press, 2020, ch. 11 (DOI : https://doi.org/10.7551/mitpress/11087.003.0015).

Cabanac, G., Labbé, C. et Magazinov, A. (2021). *Tortured phrases: A dubious writing style emerging in science. Evidence of critical issues affecting established journals*. arXiv (DOI : https://doi.org/10.48550/arXiv.2107.06751).

Cabanac, G., Labbé, C. et Magazinov, A. (2022). *The 'Problematic Paper Screener' automatically selects suspect publications for post-publication (re)assessment*. CoRR, vol. abs/2107.06751 (DOI : https://doi.org/10.48550/arXiv.2210.04895).

Lay, P., Landschat, M. et Labbé, C. (2022). *Investigating the detection of Tortured Phrases in Scientific Literature*. In Proceedings of the Third Workshop on Scholarly Document Processing (SDP'2022), ACL, Octobre 2022, p. 32–36 (DOI : https://doi.org/10.48550/arXiv.2210.13024).

Kashnitsky, Y., Herrmannova, D., Waard, de, A., Tsatsaronis, G., Fennell, C. et Labbé, C. (2022). *Overview of the DAGPap22 Shared Task on Detecting Automatically Generated*

Scientific Papers. In Proceedings of the Third Workshop on Scholarly Document Processing (SDP'2022), ACL, Octobre 2022, p. 210–213 (URL : https://aclanthology.org/2022.sdp-1.4/).

Becker, J., Wahle, J.-P., Ruas, T. et Gipp, B. (2023). *Paraphrase Detection: Human vs. Machine Content*. arXiv (DOI : https://doi.org/10.48550/arXiv.2303.13989).

Summary

Public policies push researchers to steadily publish scientific articles in reputable journals. This pressure leads a minority of unscrupulous people to commit fraud, notably by resorting to tortured phrases to disguise plagiarism. Sometimes this type of content is not detected by the peer review process, leading to its publication. In order to decontaminate the scientific literature, we propose a tortured acronym extraction task to detect fraudulent publications. A benchmark is run using a corpus of 75 tortured scientific articles in open access, with an acronym extraction and classification pipeline. We obtained an F-score of 0.74 for the tortured acronym extraction task, which can be improved by enriching the dataset. We have created a baseline against which anyone wishing to improve this task can refer to.

Caractéristique	Classe		
	Non torturé	Torturé	Suspect
Fautes de frappe	-	-	Tru_-Positive (TP) – 47 acronymes de 29 sources distinctes
Même nombre d'initiales (1)	Pain Dysfunction Syndrome (PDS) – 398 acronymes de 68 sources distinctes	Concealed Markov Display (HMM) – 253 acronymes de 69 sources distinctes	-
Nombre différent d'initiales (2)	Centers for Diseases Control and Prevention (CDC) – 50 acronymes de 31 sources distinctes	Summed Up Direct Models (GLM) – 75 acronymes de 37 sources distinctes	-
Mots composés (3)	Intraocular Pressure (IOP) – 72 acronymes de 35 sources distinctes	Multidrug Safe (MDR) – 11 acronymes de 10 sources distinctes	-
Composés chimiques (4)	3-Aminopropyltriethoxysilane (3-APTES) – 43 acronymes de 5 sources distinctes	Diethylenetriamine Pentaacetic Ruinous (DTPA) – 4 acronymes de 2 sources distinctes	-
Formes réduites (5)	Residual Networks (ResNets) – 5 acronymes de 3 sources distinctes	Lingering Brain Organizations (ResNet) – 7 acronymes de 6 sources distinctes	-
Semi-acronymes	Gabor-HOG (GHOG) – 12 acronymes de 9 sources distinctes	Non-straight ARMA models (NARMA) – 12 acronymes de 9 sources distinctes	-
Nombres	Three-Dimensional (3D) – 2 acronymes de 2 sources distinctes	Fifth-age (5G) – 1 occurrence	-
Synonymes	-	Human PC Interface/Cooperation (HCI) - 1 occurrence	-
Suffixes supplémentaires	-	PC Assisted Determination (CADx) - 1 occurrence	-
Acronymes hallucinés	-	Bolster Vector Machine (BVM) - 1 occurrence	-

TAB. 1 – *Caractéristiques des acronymes par classe. Les comptages effectués ne tiennent pas compte du nombre d'occurrence d'un même acronyme dans une même source. Les sources peuvent comporter des acronymes de caractéristiques différentes. La classe des mots composés est distincte de celle du nombre différent d'initiales mais peut très bien englober celle des composés chimiques, une distinction a été faite car il pourrait être pertinent d'utiliser un lexique pour vérifier si un composé chimique est torturé ou non. Nous nous focalisons sur les caractéristiques (1) à (5), communes aux deux classes, qui représentent au total entre 95,63 et 97,59 % de celles-ci (voir Figure 3).*

Normalisation automatique de variables issues de bases de données en agroécologie

Oussama Mechhour[*,**,***], Sandrine Auzoux[*,**],
Clément Jonquet[****], Mathieu Roche[*,***]

[*]CIRAD, UPR AIDA & UMR TETIS, F-34398 Montpellier, France
prenom.nom@cirad.fr
[**]AIDA, Univ Montpellier, CIRAD, La Réunion, France
[***]TETIS, Univ Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, Montpellier, France
[****]MISTEA, Univ Montpellier, INRAE, Institut Agro, Montpellier, France
prenom.nom@inrae.fr

Résumé. L'objectif de ces travaux est de proposer et évaluer des méthodes de mise en correspondance entre des variables sources [1] et des variables candidates du domaine de l'agroécologie [2] (en anglais). Le but de notre démarche est d'aider l'expert à normaliser les bases de données, ainsi qu'à lier les variables sources aux variables candidates des dictionnaires des modèles utilisés en agroécologie (AEGIS [4]).

1 Introduction

La mesure de similarité entre variables constitue un problème reconnu, en particulier dans notre cas d'étude impliquant des données textuelles. Cette complexité émerge de la diversité des langues, engendrant des problèmes tels que la synonymie et la prise en compte du contexte.

Dans cet article, nous nous concentrons sur la mesure de similarité des variables utilisées dans la culture de la canne à sucre, un domaine très spécifique et peu étudié en fouille de textes. L'objectif principal de ce travail est double : résoudre la problématique de l'hétérogénéité des variables utilisées par les chercheurs, appelée variables sources (chaque chercheur ayant sa propre méthode de nomination et de description de ces variables), grâce à l'aide d'experts pour normaliser les bases de données et lier ces variables aux variables candidates (dictionnaires des modèles utilisés en agroécologie, AEGIS). D'autre part, nous visons le développement d'une interface web permettant l'application des mesures décrites dans l'article (cf. Section 3) et d'autres fonctionnalités (cf. Section 3.4). Des efforts ont été déployés pour établir une similarité entre les variables sources et candidates stockées dans le système d'information AEGIS.

1. Ce sont des variables issues des travaux de recherche dans le domaine de l'agroécologie, et pour cet article, le travail s'est focalisé sur des données spécifiques liées à la culture de la canne à sucre.

2. Ces variables, composées de termes sémantiques issus de connaissances expertes et d'ontologies de référence, ont été spécifiquement définies dans le but de faciliter la comparaison et l'analyse des données, ainsi que d'établir des liens avec des modèles de culture tels que Modèle STICS. [3]

4. AEGIS, développé par le **CIRAD** (Auzoux, 2019), est une plateforme en ligne qui permet de stocker et exploiter des données provenant d'expérimentations en agroécologie menées dans les pays du Sud.

TAB. 1 présente des exemples des noms de ces deux types de variables, sachant que chaque nom d'une variable source ou candidate se compose de deux parties : son nom avant le dernier _ et son unité de mesure après le dernier _. Nos travaux vont consister à mettre en place des mesures de similarité pour la mise en lien des variables d'agroécologie. Pour illustrer l'importance de l'intégration de mesure de similarité des variables, prenons l'exemple du *products matching* (Tracz et al., 2020), qui permet de recommander automatiquement des produits similaires aux préférences des clients de manière efficace en termes de temps et de coût.

Cet article est structuré de la manière suivante. La section 2 présente une revue de littérature des différentes approches appliquées pour résoudre la problématique de la similarité entre chaînes de caractères, une question qui a été étudiée dans plusieurs travaux de recherche, y compris dans le domaine de l'agroécologie. La section 3 décrit les approches proposées (lexicale, contextuelle et combinaison) pour mesurer la similarité entre les variables et le développement d'une interface web pratique pour appliquer ces approches. La section 4 présente les méthodes de prétraitement des données appliquées, les résultats précédents et actuels, ainsi que la discussion montrant les meilleurs résultats pour chaque mesure (lexicale et contextuelle), ainsi que les améliorations des résultats. La section 5 conclut cet article et met en avant les perspectives de notre étude.

2 État de l'art

Dans cette revue de littérature, nous abordons la problématique de la similarité entre chaînes de caractères, qui a été étudiée dans plusieurs travaux de recherche. Ces travaux se divisent généralement en deux approches distinctes : (1) approche ne prenant pas en compte de contexte (cf. Section 2.1), approche prenant en compte un contexte (cf. Section 2.2).

2.1 Approche ne prenant pas en compte de contexte

Dans cette approche, les travaux se concentrent principalement sur des mesures de similarité générales qui ne prennent pas en compte le contexte spécifique de l'agroécologie. Ils utilisent des méthodes telles que TF-IDF (Term Frequency, Inverse Document Frequency) (Jones, 1972), BM-25 (Okapi BM-25) (Robertson et al., 1994), la distance de Levenshtein (Levenshtein, 1966), la distance de Jaccard (Jaccard, 1901) et des modèles de descriptions des variables comme I-ADOPT framework [5] pour évaluer la similarité entre les variables. La distance de Levenshtein permet de calculer le nombre de changements nécessaires entre deux chaînes de caractères. Pour mieux appréhender le fonctionnement de la distance de Levenshtein, examinons l'exemple suivant : considérons les mots *agro-écologie* et *agroécologie*. La distance de Levenshtein entre ces deux mots est de 1, car il suffit de supprimer le tiret pour les transformer l'un en l'autre.

La mesure TF-IDF (Jones, 1972) est une méthode classique pour évaluer l'importance des termes dans un document par rapport à une collection de documents. Elle représente la fréquence du terme dans le document. Elle est calculée en divisant le nombre d'occurrences du terme par le nombre total de termes dans le document. La formule mathématique de TF pour un terme t dans un document d est donnée par :

5. https ://www.rd-alliance.org/group/interoperable-descriptions-observable-property-terminology-wg-i-adopt-wg/wiki/i-adopt

$$\text{TF}(t, d) = \frac{\text{nombre d'occurrences de } t \text{ dans } d}{\text{nombre total de termes dans } d}$$

L'IDF mesure l'importance globale d'un terme dans la collection de documents. Elle est calculée en prenant le logarithme inverse de la proportion du nombre total de documents sur le nombre de documents contenant le terme. La formule mathématique de IDF pour un terme t dans une collection de documents est donnée par :

$$\text{IDF}(t) = \log\left(\frac{\text{nombre total de documents}}{\text{nombre de documents contenant } t}\right)$$

La mesure TF-IDF est obtenue en multipliant la valeur de TF par la valeur de IDF pour chaque terme. Ainsi, les termes qui sont fréquents dans un document particulier tout en étant rares dans l'ensemble de la collection auront une valeur TF-IDF plus élevée. Cependant, notons que ces approches peuvent présenter certaines limites pour capturer la similarité sémantique et de prendre en compte le contexte spécifique de l'agroécologie. Cette approche a été abordée dans un travail précédent (Ngaba, 2022), où la distance de Levenshtein a été utilisée comme mesure de similarité entre les noms des deux types de variables. De plus, le TF-IDF et la mesure cosinus (Manning et al., 2008) ont été utilisés respectivement pour la vectorisation des descriptions et la mesure de similarité entre ces vecteurs. Ces deux techniques ont ensuite été combinées dans une méthode appelée combinaison (cf. Section 3.3). Cependant, il est important de noter que l'utilisation du TF-IDF et Levenshtein dans cette approche ne tient pas compte du contexte spécifique de l'agroécologie, ce qui peut limiter les résultats obtenus.

Le framework I-ADOPT (InteroperAble Descriptions of Observable Property Terminology), faisant partie des modèles de description des variables scientifiques, de la RDA [6] vise à aborder le volet 'I' (interoperability) des principes FAIR [7]. Une variable est la combinaison de tous les composants descriptifs considérés comme nécessaires pour comprendre ce qui a réellement été observé, mesuré, simulé ou calculé. Elle décrit ce qui a été observé, mesuré, simulé ou calculé, indépendamment de L'endroit, de la manière, et du moment où l'acquisition des données a eu lieu. Par exemple, *Concentration of endosulfan sulfate in the flesh of Ostrea edulis expressed per unit wet weight*. La variable la plus simple requiert au moins un ObjectOfInterest et une Property. Cependant, une variable plus complexe impliquerait davantage d'entités ayant le rôle de Matrix et/ou ContextObject(s). Property représente la qualité d'un ObjectOfInterest, agissant comme un élément dépendant et une caractéristique de ObjectOfInterest. Les propriétés peuvent être exprimées en tant que types de quantité (QuantityTypes) ou types de qualité (QualityTypes) lorsqu'elles sont observées, mesurées, calculées ou simulées, par exemple : hauteur, couleur, vitesse, concentration, densité. ObjectOfInterest désigne l'entité qui joue le rôle de la cible d'observation pour laquelle la propriété est observée. Par exemple, dans *Concentration of endosulfan sulfate in the flesh of Ostrea edulis expressed per unit wet weight* 'Endosulfan sulfate' est l'ObjectOfInterest. ContextObject est l'entité qui joue un rôle descriptif pour fournir le contexte de ObjectOfInterest. Par exemple, dans *Concentration of endosulfan sulfate in the flesh of Ostrea edulis expressed per unit wet weight* 'Ostrea edulis' est le ContextObject. Matrix désigne l'entité qui joue le rôle du contexte matériel de

6. https ://www.rd-alliance.org/

7. Les principes FAIR (Findable, Accessible, Interoperable, Reusable) décrivent comment les données doivent être organisées pour être plus facilement accessibles, comprises, échangeables et réutilisables.

ObjectOfInterest. Par exemple, dans *Concentration of endosulfan sulfate in the flesh of Ostrea edulis expressed per unit wet weight* 'Flesh' est la Matrix.

À notre connaissance, aucune recherche dans le domaine de l'agroécologie n'a encore mis en oeuvre la combinaison des approches ne prenant pas en compte de contexte et de celles prenant en compte un contexte (cf. Section 2.2). Par conséquent, notre travail vise à combler cette lacune en explorant la faisabilité et l'efficacité de cette méthode dans le contexte spécifique de l'agroécologie.

2.2 Approche prenant en compte un contexte

Cette approche utilise des techniques qui prennent en considération un contexte, comme FastText (Bojanowski et al., 2016), Word2Vec (Mikolov et al., 2013) et les modèles de langues (Jurafsky et Martin, 2019). Ces derniers jouent un rôle essentiel dans le domaine du traitement automatique du langage naturel. Ils visent à capturer les structures et les relations linguistiques dans un corpus de texte pour générer des prédictions précises. Dans cette sous-section, nous aborderons les différents types de modèles de langues, en commençant par les n-grammes (Jurafsky et Martin, 2019), puis en discutant des modèles basés sur les RNNs (Recurrent Neural Networks) (Yin et al., 2017) et les LSTM (Long Short-Term Memory) (Hochreiter et Schmidhuber, 1997), pour finalement présenter les modèles de langues basés sur les transformers, tels que BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018).

n-grammes : les modèles de langues (Jurafsky et Martin, 2019) basés sur les n-grammes sont parmi les plus simples. Ils reposent sur l'idée d'exploiter les fréquences d'apparition des n-grammes (séquences de mots) dans un corpus pour prédire le mot suivant. Par exemple, un modèle de langue basé sur les trigrammes considère les deux mots précédents pour prédire le prochain mot. Bien que les n-grammes soient faciles à mettre en oeuvre et puissent fournir des résultats acceptables pour des tâches de langage simples, ils présentent des limites majeures. L'une des principales faiblesses des n-grammes est leur incapacité à capturer les dépendances à long terme entre les mots, ce qui limite leur capacité à générer des séquences de mots cohérentes.

RNNs et LSTM : pour remédier aux limites des modèles de langues basés sur les n-grammes, les RNNs ont été introduits. Les RNNs sont conçus pour traiter des séquences de données, ce qui les rend bien adaptés pour modéliser les séquences de mots dans un texte. Cependant, les RNNs traditionnels souffrent du problème du *vanishing gradient* et ont du mal à capturer des dépendances à long terme. Les LSTM sont une extension des RNNs qui permettent de mieux gérer les dépendances à long terme. Grâce à l'utilisation de portes de mémoire, les LSTM peuvent décider de conserver, de modifier ou d'oublier certaines informations, ce qui améliore la capacité du modèle à capturer des dépendances à plus long terme. Cependant, même les LSTM ont leurs limites. Ils peuvent avoir du mal à capturer des relations complexes entre les mots et peuvent souffrir de problèmes de surapprentissage lorsqu'ils sont confrontés à de grands ensembles de données.

Modèles de langues fondés sur les transformers : les modèles de langues basés sur les transformers (Vaswani et al., 2017), tels que BERT (Devlin et al., 2018), ont révolutionné le domaine du TALN (Ranjan et al., 2016) ces dernières années. Les transformers exploitent une

architecture d'attention (Vaswani et al., 2017 ; Niu et al., 2021) pour capturer les relations entre tous les mots d'une séquence, à la fois dans le contexte avant et après. Cette approche bidirectionnelle permet au modèle de comprendre plus efficacement les relations sémantiques complexes dans le texte. BERT a été largement reconnu pour ses performances exceptionnelles dans de nombreuses tâches de TALN (Ranjan et al., 2016), y compris la compréhension de texte, la traduction et le résumé automatique. Son architecture transformer lui permet de capturer de manière plus précise les dépendances à long terme par rapport aux modèles précédents. De plus, BERT peut être pré-entraîné sur de vastes corpus non supervisés, ce qui lui permet d'acquérir une connaissance linguistique riche et de s'adapter à diverses tâches spécifiques. BERT (Devlin et al., 2018) est un modèle de langues pré-entraîné sur de vastes corpus, tels que Wikipedia, qui utilise le mécanisme d'attention (Vaswani et al., 2017 ; Niu et al., 2021) pour comprendre le contexte des mots. Il prend en compte à la fois le contexte à gauche et à droite de chaque mot, ce qui lui permet de capturer les informations contextuelles de manière précise. Cette capacité à saisir le contexte permet à BERT de générer des représentations vectorielles de mots riches en sémantique. Il convient de noter qu'il existe deux types de modèles BERT couramment utilisés : BERT-base (Devlin et al., 2018) et BERT-large (Devlin et al., 2018).

3 Approches proposées

Dans le cadre de la problématique abordée dans l'introduction, qui est très spécifique et peu étudiée en fouille de textes, en plus de la création d'une interface web, différentes méthodes de fouille de texte sont utilisées pour proposer les variables candidates les mieux adaptées aux variables sources. Les méthodes suivantes sont mobilisées : (1) Des mesures lexicales, (2) Des mesures contextuelles et (3) La combinaison de ces deux dernières.

Dans un premier temps, notre démarche consiste à évaluer la similarité entre chaque variable source (au nombre de 84) et l'ensemble des 84 variables candidates, en les classant de la plus similaire à la moins similaire, en se basant uniquement sur les noms et les descriptions des variables. Par la suite, nous avons enrichi notre méthode en incorporant 15 articles en anglais représentatifs du domaine de l'agroécologie, constitués manuellement avec l'aide de 3 experts. Chaque article a une moyenne de 79 399 mots, contribuant ainsi à un corpus final (composé de 15 articles, des noms et des descriptions des variables sources et candidates) totalisant 5 620 198 mots. FIG. 1 illustre la similarité entre les variables sources et candidates en utilisant 15 articles en anglais comme informations supplémentaires. Pour la similarité sans contexte (se basant uniquement sur les noms et les descriptions des variables), le principe reste le même, mais les articles sont exclus. Pour l'évaluation des résultats, nous utilisons un fichier .csv qui contient le nom de chaque variable source et son vrai nom de sa variable candidate. TAB. 1 montre les 4 premières lignes de ce fichier à titre d'exemple, mais en totalité, nous avons 84 variables sources et 84 variables candidates.

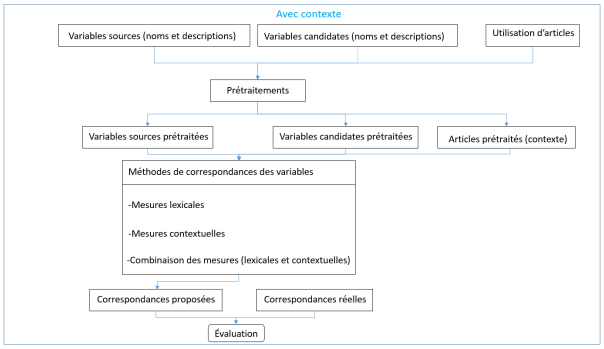

FIG. 1 – *Correspondance des variables avec enrichissement contextuel.*

Variable source	Variable candidate
Yield_CAS_t.ha-1	stem_crop_yield_fm_t.ha-1
Sugar_CAS_%	stem_sugar_fm_content_%
Rec_pds_R_%	plant_ground_cover_%
Rec_globale_plein_%	plant_ground_cover_%

TAB. 1 – *Exemples de noms de variables sources et candidates (en anglais).*

3.1 Mesure lexicale

Le but de l'approche lexicale est de comparer les variables à travers leurs noms. Dans le cadre de cette approche, la distance de Levenshtein (Levenshtein, 1966), la distance de Jaccard (Jaccard, 1901), TF-IDF (Jones, 1972), BM-25 (Robertson et al., 1994), FastText (Bojanowski et al., 2016), Word2Vec (Mikolov et al., 2013) ainsi que des modèles de langues tels que BERT-base (Devlin et al., 2018), BERT-large (Devlin et al., 2018), XLNet (Yang et al., 2019) et RoBERTa (Liu et al., 2019) ont été utilisés.

3.2 Mesure contextuelle

Le but de l'approche contextuelle est de comparer les variables à travers leurs descriptions. Dans cette approche, nous utilisons deux types de méthodes : sans contexte [8], comme TF-IDF (Jones, 1972) et BM-25, et avec contexte [9], en utilisant FastText (Bojanowski et al., 2016), Word2Vec (Mikolov et al., 2013) et des modèles de langues tels que BERT-base (Devlin et al., 2018), BERT-large (Devlin et al., 2018), XLNet (Yang et al., 2019) et RoBERTa (Liu et al., 2019).

8. Les méthodes qui ne prennent pas en considération un contexte se basent principalement sur des mesures de similarité ou de fréquence de termes.

9. Les méthodes qui prennent en compte de contexte utilisent souvent des modèles linguistiques avancés, tels que BERT.

3.3 Combinaison

La méthode que nous proposons permet de combiner les mesures lexicales et contextuelles afin d'améliorer la précision de la similarité des variables. Cette méthode est définie par la formule suivante :

$$\textbf{combinaison} = \alpha \cdot \mathrm{X} + (1 - \alpha) \cdot Y$$

où $\alpha \in]0, 1[$ est un facteur de pondération attribué à X et Y. Il est utilisé pour contrôler l'influence respective de X et Y (où X et Y représentent la mesure cosinus dans le travail actuel, mais dans le travail de (Ngaba, 2022), X représente le cosinus et Y représente la distance de Levenshtein) dans la combinaison finale. Après avoir effectué la vectorisation des noms et des descriptions des variables en utilisant les approches lexicales et contextuelles, nous appliquons la mesure cosinus pour évaluer la similarité entre les variables sources et les variables candidates. L'utilisation de ce paramètre (α) permet un réglage précis du degré d'importance attribué à X et Y dans la combinaison. Dans cette étude, nous avons exploré les valeurs de ce paramètre dans la plage de 0,1 à 0,9, avec un pas de 0,01.

3.4 Interface graphique

Au cours de ce travail, une interface web [10], a été développée pour fournir aux chercheurs un outil pratique. Elle prend en compte 4 fichiers texte : noms des variables sources, noms des variables candidates, descriptions des variables sources et descriptions des variables candidates. Elle leur permet d'appliquer les mesures de similarité entre les variables sources et les variables candidates expliquées dans la section 3.

4 Expérimentations

Après avoir expliqué les approches proposées, nous détaillons ici l'étape de préparation du jeu de données. Ensuite, nous montrerons nos résultats et les comparerons avec les résultats précédents (Ngaba, 2022).

4.1 Préparation du jeu de données

La préparation des données textuelles joue un rôle essentiel. Elle vise à transformer les noms et les descriptions des variables en une forme appropriée pour faciliter leur comparaison et leur similarité. Il existe plusieurs logiciels pour la préparation de données, tels qu'OpenRefine (Petrova-Antonova et Tancheva, 2020) et Trifacta (Petrova-Antonova et Tancheva, 2020). Cependant, pour notre travail, nous avons appliqué nos propres méthodes pour la préparation des variables (noms et descriptions). Voici l'ordre de leur application :

1. `clean_text()` : Cette fonction est utilisée pour nettoyer les variables sources et candidates. Elle élimine les nombres, les parenthèses [11] et leur contenu, et convertit les noms et les descriptions en minuscules. En normalisant les variables, cette étape facilite la comparaison et l'alignement ultérieur.

10. https ://drive.google.com/drive/folders/1NRUoG4LxAlXGnqMYsNz9QRQny1AVyWJl

11. La suppression des parenthèses et de leur contenu a amélioré les performances. Cependant, pour les travaux futurs, nous explorerons comment tirer parti de ces informations.

2. `remove_stopwords()` : Les stopwords sont des mots couramment utilisés dans la langue qui n'apportent pas de signification particulière dans le contexte de l'agroécologie. Cette fonction supprime ces mots fonctionnels, tels que les prépositions et les conjonctions. En éliminant les stopwords, nous nous concentrons sur les termes clés qui sont plus significatifs pour la mesure de similarité entre les variables.
3. `lemmatize()` : La lemmatisation est une technique linguistique qui consiste à ramener les termes à leur forme canonique ou à leur lemme. Elle permet de transformer les noms du pluriel au singulier et les verbes à leur forme infinitive. Par exemple, elle permet de ramener les termes *rédige*, *rédiges* et *rédigé* à leur forme de base *rédiger*. La lemmatisation facilite la mesure de similarité entre les termes similaires, améliorant ainsi la précision de l'alignement des variables.
4. `remove_punctuation()` : Cette fonction supprime la ponctuation des descriptions des variables. En éliminant les caractères spéciaux tels que les points, les virgules et les guillemets, nous évitons les interférences indésirables lors de la mesure de similarité entre les variables.
5. `replace_synonyms()` : Pour faciliter la mesure de similarité entre les variables, cette technique permet de remplacer certains mots par leurs synonymes. Par exemple, le mot *degré* peut être remplacé par *niveau*. En utilisant des termes équivalents, cette étape a amélioré la cohérence et la précision de l'alignement des variables.

Toutes les fonctions ont été appliquées à l'ensemble des descriptions des variables sources et candidates, tandis que seules les trois premières fonctions ont été utilisées pour les noms des variables. TAB. 2 présente trois exemples de noms de variables sources et leurs descriptions, tandis que TAB. 3 présente trois exemples de noms de variables candidates et leurs descriptions. Nous comptons un total de 84 variables sources et 84 variables candidates.

Nom	Description
Yield_CAS_t.ha-1	Cane yield (in fresh machinable stem)
Sugar_CAS_%	Sugar content of fresh stem mass
Rec_globale_plein_%	full weed and service plant coverage

TAB. 2 – *Exemples de noms et de descriptions des variables sources (en anglais).*

Nom	Description
stem_juice_crop_yield_l.ha-1	stem juice yield
stem_nonstalk_crop_yield_fm_t.ha-1	stem crop yield fresh mass
sugar_crop_yield_dm_t.ha-1	sugar solid in stem

TAB. 3 – *Exemples de noms et de descriptions des variables candidates (en anglais).*

4.2 Comparaison des résultats

Notre objectif est de calculer la similarité de chaque variable source avec toutes les autres variables candidates, en les classant de la plus proche à la moins proche. La méthode qui nous intéresse est la combinaison, et nous nous concentrons spécifiquement sur les précisions à des positions inférieures à 10 (p@10). Noter que p@i représente la probabilité que la solution pertinente, c'est-à-dire la variable candidate qui correspond réellement à la variable source, soit parmi les i premières variables candidates proposées.

4.2.1 Méthode sans utiliser d'informations contextuelles

(Ngaba, 2022) a utilisé la distance de Levenshtein entre les noms, TF-IDF pour la vectorisation des descriptions, et le cosinus pour mesurer la similarité entre ces vecteurs. Les résultats obtenus avec α = 0.3 sont présentés dans TAB. 4.

Pour le travail actuel, l'architecture BERT-base avec 2 couches cachées a été utilisée pour la vectorisation des noms et des descriptions des variables, ce qui a conduit à de meilleurs résultats avec α = 0.79 et sans utilisation d'informations contextuelles (15 articles scientifiques). La mesure de similarité du cosinus a été utilisée pour évaluer les similitudes. Les résultats ont été améliorés (TAB. 5).

Précision	Levenshtein (noms des variables)	TF-IDF + cosinus (descriptions des variables)	Combinaison
p@1	15.48%	33.33%	41.67%
p@3	19.05%	42.86%	51.19%
p@5	23.81%	51.19%	64.29%
p@10	42.86%	60.71%	73.81%

TAB. 4 – *Résultats précedents en termes de précision (84 variables).*

Précision	BERT-base + cosinus (noms des variables)	BERT-base + cosinus (descriptions des variables)	Combinaison
p@1	11.90%	33.33%	41.67%
p@3	28.57%	44.05%	60.71%
p@5	36.90%	52.38%	69.05%
p@10	53.57%	57.14%	79.76%

TAB. 5 – *Résultats actuels en termes de précision (84 variables).*

4.2.2 Méthode utilisant le contexte

Pour le travail précédent, un corpus de documents a été utilisé dans l'étude menée par (Ngaba, 2022), dans le but d'améliorer le contexte des descriptions des variables et d'augmenter les résultats. Ce corpus est composé d'articles scientifiques, de chapitres d'ouvrages, de rapports et de thèses, se concentrant principalement sur les thèmes de la canne à sucre, de la fertilisation des sols, des plantes de service et des adventices. Au total, 122 documents, tous rédigés en anglais, ont été utilisés, variant en longueur de 5 à 160 pages. Pour plus d'informations, veuillez consulter le travail (Ngaba, 2022). TAB. 6 présente les résultats obtenus avec α = 0.3.

Pour le travail actuel, 15 articles scientifiques en anglais ont été utilisés. Ces articles ont été prétraités par les fonctions décrites dans la section 4.1, puis regroupés avec les noms et les descriptions des variables candidates et sources, tous prétraités et représentés dans un corpus unique (sous forme d'une liste de chaînes de caractères en Python), également appelé contexte. Ces données ont ensuite été pondérées avec TF-IDF pour constituer le vocabulaire. Pour la vectorisation des noms, des descriptions des variables et la mesure de similarité, l'architecture BERT-base avec 2 couches cachées, TF-IDF (utilisant 15 articles) et le cosinus ont été respectivement employés. Les résultats obtenus (TAB. 7) reposent sur α = 0.25.

Précision	Levenshtein (noms des variables)	TF-IDF + cosinus (descriptions des variables)	Combinaison
p@1	15.48%	33.33%	44.05%
p@3	19.05%	42.86%	55.95%
p@5	23.81%	51.19%	64.29%
p@10	42.86%	60.71%	73.81%

TAB. 6 – *Résultats précedents en termes de précision (84 variables).*

Précision	BERT-base + cosinus (noms des variables)	TF-IDF + cosinus (descriptions des variables)	Combinaison
p@1	11.90%	29.76%	52.38%
p@3	28.57%	42.86%	66.67%
p@5	36.90%	51.19%	71.43%
p@10	53.57%	64.29%	80.95%

TAB. 7 – *Résultats actuels en termes de précision (84 variables).*

4.3 Discussion

Différentes méthodes ont été utilisées pour la vectorisation des noms de variables (cf. Section 3.1). Parmi ces méthodes, le modèle BERT-base avec 2 couches cachées a donné les meilleurs résultats (TAB. 8). De même, pour la vectorisation des descriptions des variables, plusieurs méthodes ont été explorées (cf. Section 3.2). Cependant, les meilleurs résultats ont été obtenus avec l'utilisation de TF-IDF (avec l'utilisation de 15 articles) (TAB. 9). À cet effet, TF-IDF a été entraîné sur un corpus composé des noms, des descriptions des variables et d'un contexte supplémentaire issu de 15 articles.

Précision	BERT-base + cosinus	BERT-large + cosinus	XLNet + cosinus	RoBERTa + cosinus
p@1	11.90%	11.90%	9.52%	7.14%
p@3	28.57%	17.86%	23.81%	11.90%
p@5	36.90%	25.00%	27.38%	19.05%
p@10	53.57%	27.38%	34.52%	28.57%

TAB. 8 – *Les résultats en termes de précision obtenus pour la vectorisation des **noms** des variables (84 variables).*

Précision	BERT-large + cosinus	XLNet + cosinus	RoBERTa + cosinus	TF-IDF + cosinus
p@1	21.43%	11.90%	14.29%	41.67%
p@3	32.14%	21.43%	21.43%	54.76%
p@5	38.10%	25.00%	22.62%	60.71%
p@10	50.00%	30.95%	23.81%	69.05%

TAB. 9 – *Les résultats en termes de précision obtenus pour la vectorisation des **descriptions** des variables (84 variables).*

Plusieurs combinaisons de méthodes ont été utilisées pour la vectorisation des variables. Les meilleurs résultats obtenus avec ces combinaisons sont tout à fait satisfaisants, avec une

précision allant de 52.38% à 80.95%, en considérant respectivement le premier jusqu'aux 10 premiers éléments retournés. En outre, nos résultats ont largement dépassé ceux de (Ngaba, 2022). Pour la méthode de combinaison, les résultats ont augmenté de 9.52%, 4.76% et 5.95% pour p@3, p@5 et p@10 respectivement. Pour la similarité (i) entre les noms, les résultats ont augmenté de 9.52%, 1.09% et 10.71% pour p@3, p@5 et p@10 respectivement, (ii) entre les descriptions, les résultats ont augmenté de 1.19% pour p@3 et p@5.

Notons enfin que les expérimentations réalisées avec la mesure de Jaccard ont montré une augmentation de p@1 (p@1 = 58.33%), p@3 (70.24%) et p@5 (73.81%), mais en réduisant la précision pour p@10 (78.57%). Okapi BM-25, Fastext, GloVe et Word2Vec ont donné des résultats peu satisfaisants qu'il faudra approfondir dans les futurs travaux.

5 Conclusion et perspectives

Dans nos travaux, plusieurs approches ont été testées pour mesurer la similarité entre les variables sources et candidates, notamment l'utilisation du modèle BERT-base qui a donné de meilleurs résultats pour la vectorisation des noms de variables. Pour les descriptions des variables, la méthode TF-IDF avec l'enrichissement contextuel de 15 articles scientifiques en anglais a obtenu les meilleurs résultats. Des travaux complémentaires pourraient consister à explorer l'application de modèles génératifs tels que Chat-GPT pour générer un contexte de variables. Enfin, nous envisageons d'appliquer des modèles de description de variables scientifiques, tels que I-ADOPT de la RDA. Notre approche consistera à utiliser ce framework pour décrire nos variables, puis à évaluer la similarité entre les variables sources et candidates.

Remerciements : Ce travail est financé par l'Agence Nationale de la Recherche (ANR) au titre de France 2030 portant la référence ANR-16-CONV-0004 (#DigitAg) et par le programme de recherche et d'innovation Horizon Europe dans le cadre de la convention 101081973 - IntercropValuES. Ce travail a bénéficié du soutien du Conseil régional de La Réunion, du ministère français de l'Agriculture et de l'Alimentation, de l'Union européenne (programme Feader, subvention AG/974/DAAF/2016-00096 et programme Feder, subvention GURTDI 20151501-0000735).

Références

Auzoux, S. (2019). Aegis, an extended information system to support agroecological transition for sugarcane industries. In *ISSCT*.

Bojanowski, P., E. Grave, A. Joulin, et T. Mikolov (2016). Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*.

Devlin, J., M. W. Chang, K. Lee, et K. Toutanova (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the ACL : Human Language Technologies*, pp. 4171–4186.

Hochreiter, S. et J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 17351780.

Jaccard, P. (1901). Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles 37*, 241–72.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Jurafsky, D. et J. H. Martin (2019). *Speech and Language Processing (3rd Edition)*. Pearson.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10*(8), 707–710.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et V. Stoyanov (2019). Roberta : A robustly optimized bert pretraining approach. *arXiv preprint arXiv :1907.11692*.

Manning, C. D., P. Raghavan, et H. Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Mikolov, T., K. Chen, G. Corrado, et J. Dean (2013). Efficient estimation of word representations in vector space.

Ngaba, B. (2022). Rapport de stage : Couplage d'un modèle de culture avec une plateforme de capitalisation des données issues d'agroécosystèmes à la réunion.

Niu, Z., G. Zhong, et H. Yu (2021). A review on the attention mechanism of deep learning. *Neurocomputing 452*, 4862.

Petrova-Antonova, D. et R. Tancheva (2020). Data cleaning : A case study with openrefine and trifacta wrangler. In *Int. Conf. on the Quality of Inf. and Com. Technology*, pp. 32–40.

Ranjan, N., K. Mundada, K. Phaltane, et S. Ahmad (2016). A survey on techniques in nlp. *International Journal of Computer Applications 134*(8), 69.

Robertson, S., S. Walker, S. Jones, M. Hancock-Beaulieu, et M. Gatford (1994). Okapi at trec-3. pp. 0–.

Tracz, J., P. I. Wójcik, K. Jasinska-Kobus, R. Belluzzo, R. Mroczkowski, et I. Gawlik (2020). BERT-based similarity learning for product matching. In *Proceedings of Workshop on Natural Language Processing in E-Commerce*, pp. 66–75.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, et I. Polosukhin (2017). Attention is all you need. In *Advances in neural information processing systems 30*.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, et Q. V. Le (2019). Xlnet : Generalized autoregressive pretraining for language understanding. In *arXiv preprint arXiv :1906.08237*.

Yin, W., K. Kann, M. Yu, et H. Schütze (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv :1702.01923*.

Summary

The objective of this work is to propose and evaluate methods of matching between source variables and candidate variables from the agroecology domain (in English). The aim of our approach is to support the experts to standardize the databases, as well as to link the source variables to the candidate variables of the dictionaries of the models used in agroecology (i.e. AEGIS).

Approches linguistiques pour la fouille d'articles scientifiques

Iana Atanassova

CRIT, Université de Franche-Comté / Institut Universitaire de France (IUF)
iana.atanassova@univ-fcomte.fr

Résumé :

Dans cette présentation nous aborderons la problématique de la fouille de corpus scientifiques en différentes disciplines et en plein texte. Nous montrerons des résultats de l'annotation sémantique de corpus et l'extraction d'informations, par l'application d'approches fondées sur les ressources linguistiques qui seront mis en perspective avec des méthodes par apprentissage.

Nous nous intéressons en particulier à la notion d'incertitude scientifique et à la possibilité de l'identifier et la catégoriser dans les textes des publications.

Nous aborderons également la problématique du multilinguisme dans les articles à travers une étude de corpus.

Optimiser l'annotation de données : évaluation critique des outils et présentation de Labelit

Mohamed Chetouani

Batvoice Technologies
mchetouani@batvoice.com

Résumé :

L'importance cruciale des données dans le développement de l'intelligence artificielle met en lumière le rôle essentiel de l'annotation, une intervention humaine sur les données. Cette intervention peut prendre diverses formes telles que l'étiquetage, la segmentation, la description, l'évaluation ou la transcription de données. Les progrès significatifs dans le domaine de l'IA sont attribuables à la disponibilité de données de qualité annotées.
L'annotation est considérée comme une étape clé dans la conception de systèmes d'IA, et de nombreux outils d'annotation sont disponibles pour les domaines académiques et non académiques. Cependant, l'utilisation de plusieurs outils spécifiques pour différentes tâches d'annotation sur des données hétérogènes présente des défis.
La présentation examine divers outils d'annotation, se concentrant sur des critères tels que l'ouverture, la disponibilité en ligne, la maintenance et la personnalisation. Cette analyse révèle certaines limitations des outils actuels, telles que la difficulté de définir et de partager des stratégies d'annotation, le manque de transparence dans la gestion des données et les défis liés à la charge de travail des annotateurs.
En réponse à ces défis, la présentation introduit l'outil Labelit, un outil ouvert et en ligne (`https://github.com/voicelab-org/labelit`) initialement développé pour les besoins de Batvoice. Labelit facilite les campagnes d'annotation de tâches multiples sur des données audio, en particulier des appels téléphoniques, en assurant une gestion contrôlée des données et de la charge de travail des annotateurs. L'outil tire parti de technologies logicielles récentes, offrant une adaptabilité efficace à de nouvelles tâches, des changements de stratégie d'annotation et à l'annotation de données hétérogènes. Des cas d'utilisation illustreront les fonctionnalités de Labelit au cours de la présentation.

Indexation semi-supervisée abstractive-extractive de documents

Saber ZAHHAR*, Christophe RODRIGUES**

* zahhar.saber@gmail.com
** Research Center, Léonard de Vinci Pôle Universitaire, Paris La Défense, France
christophe.rodrigues@devinci.fr

Résumé. L'indexation de documents est un enjeu majeur dans la facilitation d'accès à l'information et la veille scientifique. Du fait des coûts associés à l'indexation manuelle de documents par des professionnels, les auteurs et journaux optent généralement pour des méthodes automatiques. Cependant, on observe un monopole des méthodes dites extractives, i.e. générer des index correspondant aux mots présents dans le texte source. Avec la démocratisation des réseaux de neurones et jeux de données variés, il est aujourd'hui possible d'effectuer une génération abstractive, i.e. générer des mots-clés qui ne sont pas présents dans le texte source. Cet article présente une approche dans la génération d'index appartenant à un vocabulaire contrôlé. Nous illustrons l'application de cette méthode à différents types d'articles pour établir les limites actuelles de la situation.

1 Introduction

Les bibliothèques numériques servent de principal conduit pour l'accès, la préservation et l'organisation des connaissances. Cela est particulièrement évident dans le domaine de la publication scientifique, où l'exponentielle et incessante augmentation du nombre de publications entraîne diverses difficultés (Larsen and Von Ins, 2010). L'un des défis concerne le processus d'indexation des publications. Traditionnellement, les bibliothèques physiques utilisaient l'année de publication ou encore les noms d'auteurs pour organiser les documents ainsi que la catégorie pour classer les documents, e.g. "science-fiction" ou "sciences sociales". Ces index ne répondent plus adéquatement au besoin d'une indexation basée sur le contenu.

Les index, ou plus couramment appelés *mots-clés*, sont des mots et expressions qui encapsulent les idées fondamentales d'un document. Ces mots-clés offrent une représentation plus complète et fidèle des travaux publiés, avec le potentiel d'aider les moteurs de recherche à fournir de meilleurs résultats plus rapidement (Gutwin et al., 1999) ou encore à servir pour des tâches avancées comme l'analyse de sentiment (Berend et Vincze, 2012).

Alors que les mots-clés deviennent de plus en plus standard dans le processus de publication, une mise en œuvre informative et cohérente reste un défi considérable. En effet, en raison des coûts associés à l'attribution manuelle mots-clés par des professionnels, les auteurs se voient souvent confier cette tâche sans avoir particulièrement d'expertise ou formation adéquate (Strader, 2009; Kipp, 2011). Par conséquent, les mots-clés générés s'avèrent souvent inadéquats pour la recherche interdisciplinaire (Kwon, 2018). De plus, la majorité des

recherches se concentrent principalement sur l'extraction des mots-clés directement à partir du texte source, appelée méthode *extractive*. Ces méthodes ne contribuent pas à enrichir le document car elles ne font qu'extraire des données déjà présentes, le choix des mots pouvant être subjectif, cela n'apporte aucune aide à l'indexation qui se veut standardisée (Koh et al., 2022). L'état de l'art appelle à de nouvelles approches qui pourraient générer des mots-clés à partir de sources externes au matériel (Hulth, 2003), c'est-à-dire des méthodes *abstractives*.

Cet article est organisé en cinq sections. Dans la Section 2, nous effectuerons une analyse approfondie de l'état de l'art dans le domaine de l'indexation automatique d'articles. Par la suite (Section 3), nous exposerons notre méthodologie de modélisation en détaillant sa mise en œuvre pratique. En Section 4, nous étayerons notre approche au travers d'une série d'expériences. Enfin, dans la Section 5, nous partagerons nos conclusions quant à la situation actuelle et explorerons les voies envisageables pour l'indexation automatique.

2 État de l'art

2.1 Méthodes extractives

La disponibilité limitée de documents annotés de mots-clés a longtemps contraint les chercheurs à adopter des approches non supervisées d'indexation. Ainsi, les travaux initiaux se sont principalement concentrés sur des méthodes statistiques cherchant à capturer l'importance des candidats mots-clés (Tomokiyo and Hurst, 2003). En parallèle des méthodes statistiques s'est développée des approches graphes (Zha, 2002; Mihalcea and Tarau, 2004).

Les algorithmes de classification classiques ont prouvé leur capacité à rivaliser avec les méthodes non supervisées malgré le manque de données d'entraînement et les hypothèses relatives aux variables de ces dernières (Kim et al., 2010; Hasan and Ng, 2014).

Depuis l'avènement et démocratisation de l'apprentissage profond et ses applications, les conditions sont optimales pour pleinement tirer parti des vastes corpus de traitement automatique du langage naturel disponibles récemment. Plus précisément, les modèles de séquence à séquence ont gagné en importance (Sutskever et al., 2014), ouvrant ainsi une nouvelle voie d'exploration dans ce domaine. Dans l'ensemble, ces architectures ont établi un nouveau jalon dans le domaine (Ye and Wang, 2018) et démontrent maintenant des promesses dans le contexte de l'indexation de texte (Boudin et al., 2014).

2.2 Méthodes abstractives

En effet, ces modèles permettent la génération *abstractive* de mots-clés. (Hulth, 2003) a souligné la nécessité de générer, plutôt que d'extraire, des mots-clés. Des travaux ultérieurs ont tenté de développer des méthodes abstractives par l'attribution de mots-clés (Medelyan and Witten, 2006), mais la tendance s'est longtemps contentée des méthodes extractives. L'apprentissage profond peut ouvrir une nouvelle voie pour l'indexation abstractive de texte, mais le chemin ne sera pas facile car les données massives nécessaires et la faible généralisation des modèles séquentiels représentent un défi sérieux dans le développement d'un générateur de mots-clés unifié et polyvalent. Des suggestions récentes plaident en faveur d'une base de connaissances externe, telle que l'adaptation de domaine pour une indexation spécifique à la tâche (Le et Mikolov, 2014).

2.3 Modélisation du langage

Introduit par (Mikolov et al., 2013), Word2Vec est un ensemble de techniques neuronales qui tournent autour de la construction de représentations de mots dans un espace vectoriel. Doc2Vec (Le et Mikolov, 2014) étend les concepts de Word2Vec pour la représentation de documents identifiés par un index (Fig. 1). En entraînant un ensemble de paires index-document, il est possible de capturer la signification et relations des documents et index dans un même espace vectoriel, permettant des opérations comme "Londres - Royaume-Uni + France = Paris".

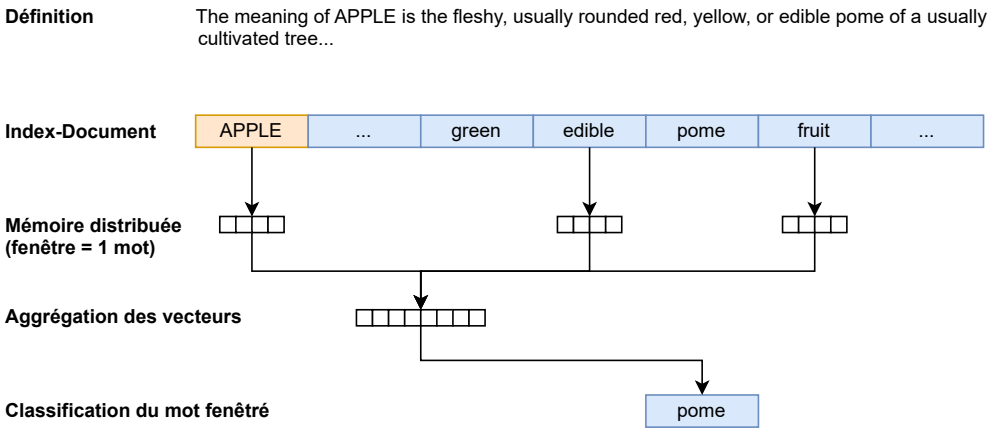

FIG. 1 – *Exemple d'entraînement d'un modèle Doc2Vec.*

2.4 Synthèse de l'état de l'art

Nous observons que les méthodes extractives non-supervisées sont suffisantes pour ne pas requérir les méthodes supervisées, plus coûteuses à mettre en place pour un résultat similaire.

Parallèlement, force est de constater l'absence de méthodes abstractives performantes. De plus, ces méthodes ne sont pas nécessairement utiles dans le contexte d'indexation, qui consiste à générer des mots-clés à partir d'un vocabulaire contrôlé, et non pas juste paraphraser des termes, ce qui souvent mène à une perte d'information à cause de la généralisation des termes, ce qu'est l'opposé souhaité par l'indexation. L'évaluation même des méthodes abstractives reste très controversée en dehors du cadre de l'indexation car il est difficile de mesurer la qualité des mots-clés générés, que ce soit en terme de cohérence avec les concepts du document ou encore de la distance avec les autres mots-clés générés.

Cependant, dans le cadre de l'indexation, la technique Doc2Vec permettrait de générer des mots-clés à partir d'un vocabulaire contrôlé et cela peu importe le nombre de sous-mots qui composent l'index, e.g. "République Française" ou "Union des républiques socialistes soviétiques". Ceci évite la nécessité d'avoir un grand nombre de n-grammes dans l'espace des documents tout en construisant une représentation vectorielle pour documents et index dans le même espace.

3 Approche d'indexation par la modélisation sémantique des index

Notre approche se décompose de la manière suivante :

— Construire un dictionnaire avec des définitions pour chaque terme d'index.

— Entraîner un modèle Doc2Vec sur un corpus de pairs index-définition.

— Pour chaque document à indexer, inférer à partir du modèle et comparer le vecteur du document aux vecteurs des termes d'index entraînés.

3.1 Construction d'un dictionnaire

Le dictionnaire contrôlé doit permettre de placer les termes d'index dans leur contexte à l'aide de définitions concises, claires et exemplifiées. Les mots employés par la définition devraient être d'une distribution similaire au langage utilisé par les documents attendus. Par exemple, indexer des articles scientifiques nécessite un dictionnaire de référence au langage soutenu et technique.

Pour construire ces définitions, nous pouvons utiliser les dictionnaires professionnels mis à disposition. Si indisponibles, nous utiliserons des dictionnaires classiques comme WordNet ou encore des dictionnaires générés par des modèles de langages comme Text-Bison.

3.2 Entraînement du modèle Doc2Vec

Il est à noter que nous avons augmenter le nombre de données d'entraînement en séparant les phrases d'une définition (Fig. 2). Nos paramètres sont `{vector_size:300, min_count:1, window:3, epochs:200}`

3.3 Indexation des textes

A l'aide des librairies `nltk` et `spacy` nous déconstruisons le texte source en une liste de listes, chaque sous-liste correspondant à une phrase. Chaque phrase est une liste de mots/tokens, nous retenons les unigrammes, bigrammes et trigrammes de chaque token.

Ensuite, nous avons la possibilité d'inférer le vecteur du document ou bien d'inférer le vecteur de chaque phrase du document. Ensuite, nous comparons les vecteurs inférés avec les vecteurs des termes d'index entraînés. Les k plus proches termes d'index sont sélectionnés comme candidats mots-clés. Comme métrique de distance nous utiliserons la similarité cosinus :

$$\text{cosim}(u, v) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}$$

Finalement, notre méthode étant principalement abstractive. Nous étudierons l'implémentation d'une méthode extractive basée sur le True Frequency – Inverse Document Frequency (TF-IDF), le TF sera la mesure d'occurrences d'un mot tandis que l'IDF sera mesuré par la fonction $f(\text{mot}) = \log\left(\frac{|\text{documents}|}{1+|\text{mot}\in\text{documents}|}\right) + 1$.

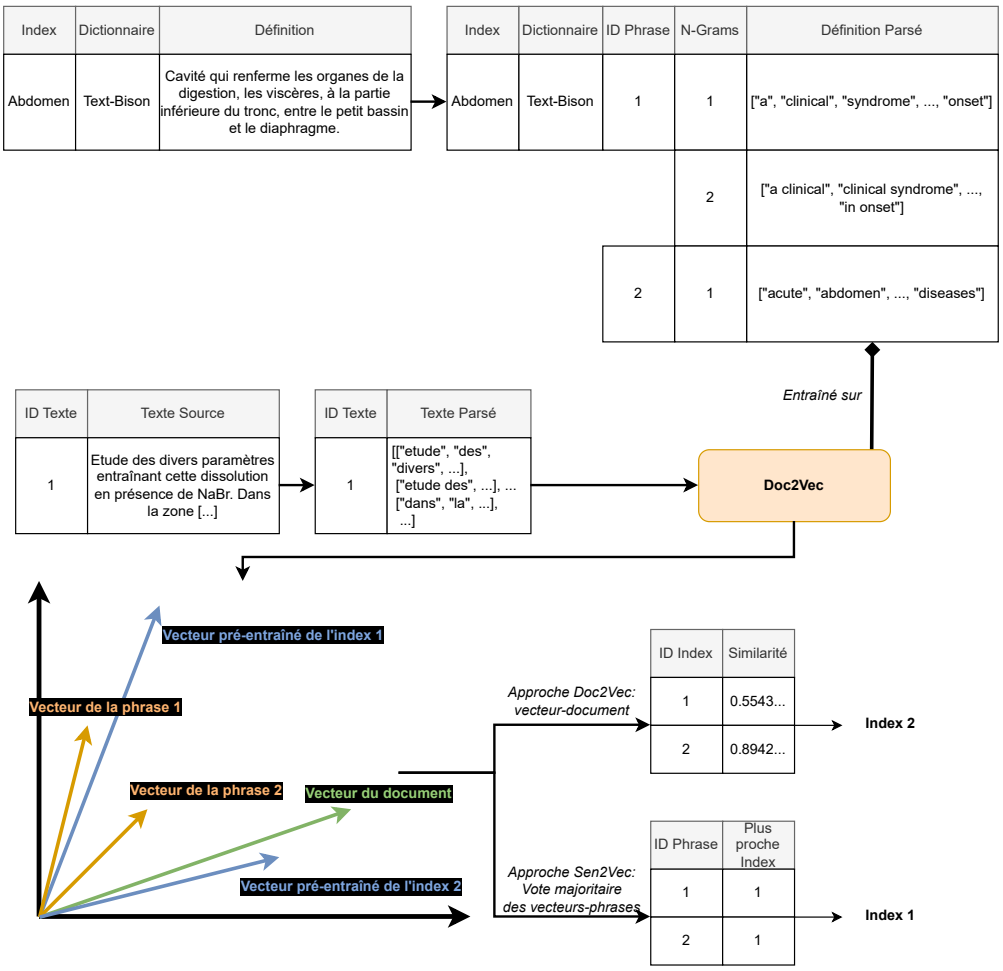

FIG. 2 – *Exemple des deux approches étudiées.*

4 Expérimentations

4.1 Jeux de données

Nous utiliserons les ensembles de données proposées sur HuggingFace par l'équipe de recherche TALN@LS2N [1], les textes proposées par cette équipe regroupe une collection destinées à la génération de mots-clés et possèdent aussi la particularité d'avoir classifier les mots-clés en quatre catégories (Fig. 3) permettant d'évaluer les performances extractives et abstractives des modèles Boudin et Gallina (2021) :

— **inspec** (Hulth, 2003) : 2000 résumés annotés par des indexeurs professionnels à partir du vocabulaire contrôlé de l'Inspec.

— **kpbiomed** (Houbre et al., 2022) : 5.6 million de résumés d'articles médicaux de PubMed annotés par des auteurs à partir du vocabulaire contrôlé médical international du Medical Subject Headings (MeSH). On utiliera uniquement 2000 résumés dans notre étude.

— **semeval-2010** (Boudin et al., 2016a) : 244 articles complets annotés par auteurs et lecteurs de manière non-contrôlée. On notera que les mots-clés sont par défaut sous radical.

— **termith-eval** (Boudin et al., 2016b) : 400 résumés d'articles français de l'Institut de l'Information Scientifique et Technique (Inist) annotés par des indexeurs professionels de manière non-contrôlée.

— **taln-archives** (Boudin, 2013) : 1207 résumés d'articles français issues du groupe TALN annotés par des auteurs de manière non-contrôlée.

FIG. 3 – *Exemple de classification (P)resent, (R)e-ordered, (M)ixed et (U)nseen.*

(Tab 1) permet d'observer que KPBioMed et TermITH-Eval sont de bonnes collections pour la génération abstractive d'index du fait de leur haute proportion de mots-clés absents du texte ; à l'inverse d'Inspec.

Il semblerait que 10 mots-clés soit le nombre de mots-clés attendus pour les articles, à l'exception de taln-archives qui ne nécessite que 5 mots-clés.

1. Disponible à l'adresse : https://huggingface.co/taln-ls2n

dataset	annotation	PRMU	# de mots-clés	# unique de mots-clés	# moy. de mots-clés/document
kpbiomed	professionnel	Present	45421 (28.18%)	7310	3.33
(n=2000)	(contrôlé MeSH)	Reordered	12212 (07.58%)	3195	0.90
		Mixed	39177 (24.31%)	6477	2.87
		Unseen	64459 (39.93%)	7427	4.72
termith-eval	professionnel	Present	1900 (40.60%)	1153	4.76
(n=400)	(non-contrôlé)	Reordered	309 (06.60%)	236	0.77
		Mixed	902 (19.27%)	617	2.26
		Unseen	1570 (33.55%)	907	3.93
semeval-2010	auteur-lecteur	Present	1496 (41.39%)	1257	6.13
(n=244)	(non-contrôlé)	Reordered	283 (11.38%)	276	1.16
		Mixed	966 (21.23%)	903	3.96
		Unseen	869 (24.05%)	800	3.56
taln-archives	auteur	Present	2661 (53.50%)	1586	2.20
(n=1207)	(non-contrôlé)	Reordered	566 (11.38%)	494	0.47
		Mixed	1056 (21.23%)	808	0.87
		Unseen	691 (13.89%)	509	0.57
inspec	professionnel	Present	14963 (77.63%)	13383	7.48
(n=2000)	(contrôlé Inspec)	Reordered	2022 (10.49%)	1969	1.01
		Mixed	1273 (06.60%)	1187	0.64
		Unseen	1016 (05.27%)	923	0.51

TAB. 1 – *Statistiques des jeux de données.*

4.2 Benchmark – Python Keyphrase Extraction

PKE (Boudin, 2016) est un module python unifiant différents modèles d'extraction de mots-clés ;

— **TF-IDF** est une technique de pondération utilisée pour évaluer l'importance d'un terme dans un document par rapport à un corpus plus large. On effectue la prédiction en fonction de la fréquence des mots dans un document particulier et de leur rareté dans l'ensemble du corpus. Dans le module PKE, le dataset SemEval-2010-pre est utilisée comme corpus de référence.

— **TextRank** est un technique d'extraction basée sur le concept PageRank de Google. Il évalue l'importance en fonction de la fréquence et du contexte d'un mot dans un texte. Les éléments les mieux notés sont considérés comme les candidats mots-clés.

— **MultipartiteRank** est une technique d'extraction étendue de TextRank qui permet de traiter des graphes multipartites. Il peut être utilisé pour extraire des mots-clés en utilisant un graphe qui relie les termes.

— **TopicRank** est une technique d'extraction basée sur la détection de sujets dans un document. Il identifie les sujets majeurs en analysant les relations entre les termes et en les groupant en clusters, prédisant ensuite les termes les plus représentatifs de chaque cluster comme les candidats mots-clés.

— **TopicalPageRank** est une technique d'extraction combinant les concepts de PageRank et de modèles de sujets pour évaluer l'importance des termes dans un document en tenant compte à la fois de leur fréquence et de leur pertinence.

— **PositionRank** est un technique d'extraction basée à la fois sur la fréquence et la position d'un terme dans un document. Les candidats mots-clés sont ceux dont la fréquence et importance relative de la position dans le document sont les plus importants.

4.3 Evaluation

Nous demandons aux modèles de générer $k \in \{5, 10\}$ mots-clés. A l'exception de SemEval-2010, nous ne racinons pas les prédictions, car il s'agit d'un exercice d'indexation où le terme exact doit correspondre. Les termes "communisme" et "commune" par exemple ont même radical mais ne possèdent pas le même sens. Si des termes comme "commune" et "communes" sont présents dans le dictionnaire, il s'agit d'une erreur de construction d'index, car ces derniers sont censés être standardisés d'un point de vue linguistique et agissent comme des axes indépendants des différents concepts existants. (Tab. 2) démontre les temps de calculs des modèles [2] :

Modèle	Temps (en s)
MultipartiteRank	1.211
PositionRank	0.873
TextRank	0.869
TfIdf	2.520
TopicRank	1.046
TopicalPageRank	1.677
approche Doc2Vec	0.969
approche Sen2Vec	2.378

TAB. 2 – *Temps de calculs des modèles.*

Nous utilisons le F1-score comme métrique d'évaluation :

$$\text{f1@}k = \frac{TP@k}{TP@k + \frac{1}{2}\left(FP@k + FN@k\right)}$$

avec :

— $TP@k$ le nombre de mots-clés générés corrects sur une prédiction de k mots clés ;

— $FP@k$ le nombre de mots-clés générés incorrects sur une prédiction de k mots clés ;

— $FN@k$ le nombre de mots-clés corrects manqués sur une prédiction de k mots clés ;

On réalisera ces calculs pour chaque sous-classe PRMU.

5 Discussions

Les résultats présentés en (Tab. 3) permettent d'établir plusieurs conclusions.

5.1 Construction du dataset SemEval-2010

On observe que les méthodes extractives parviennent à générer des mots-clés considérés comme absents du texte dans le cadre de SemEval ; cela est incohérent avec la méthode PRMU. Il est possible que la classification PRMU se soit faite sur les mots-clés non-racinés ce qui cause le décalage. Remettant en cause la pertinence du jeu de données SemEval-2010.

2. Calculs réalisés sur une machine HPC Intel Xeon CPU E5649 @ 2.53GHz 50gi RAM.

			$@5_{all}$	$@10_{all}$	$@5_{p}$	$@10_{p}$	$@5_{r}$	$@10_{r}$	$@5_{m}$	$@10_{m}$	$@5_{u}$	$@10_{u}$
subset	**experiment**	**model**										
inspec	pke	MultipartiteRank	15.67	23.17	18.09	25.70						
		PositionRank	15.37	23.29	17.67	25.87						
		TextRank	16.16	**24.71**	18.59	**27.47**						
		TfIdf	11.30	16.40	12.98	18.07						
		TopicRank	14.75	22.09	17.00	24.48						
		TopicalPageRank	16.09	24.37	18.49	27.06						
	text-bison	approche Doc2Vec	14.65	14.67	13.85	13.44	4.23	3.18	0.74	0.68	**0.17**	**0.16**
		approche Sen2Vec	**22.43**	23.40	**22.57**	22.56	**5.13**	**4.35**	**0.81**	**0.88**	0.05	0.04
		custom tfidf	9.65	11.36	11.07	12.49						
	wordnet	approche Doc2Vec	9.00	9.32	9.03	8.93	2.05	1.69	0.28	0.32	0.00	0.00
		approche Sen2Vec	19.07	19.63	20.05	19.67	2.93	2.75	0.43	0.38	0.00	0.02
		custom tfidf	11.03	13.74	12.54	15.06						
kpbiomed	pke	MultipartiteRank	2.78	4.29	5.00	6.55						
		PositionRank	2.43	3.98	4.38	6.06						
		TextRank	1.37	2.10	2.52	3.22						
		TfIdf	3.78	5.68	6.73	8.71						
		TopicRank	2.54	3.86	4.65	5.90						
		TopicalPageRank	1.81	3.02	3.29	4.56						
	mesh	approche Doc2Vec	1.22	1.43	1.16	1.08	0.37	0.36	0.53	0.61	0.19	0.18
		approche Sen2Vec	2.48	2.54	2.83	2.33	0.78	0.59	0.96	0.86	0.19	0.23
		custom tfidf	6.92	7.82	12.63	11.95						
	text-bison	approche Doc2Vec	3.26	3.77	2.51	2.46	1.10	0.99	1.94	1.74	0.62	0.67
		approche Sen2Vec	7.46	7.31	7.97	6.38	**2.09**	**1.60**	**3.26**	**2.68**	**0.88**	**0.85**
		custom tfidf	5.74	6.53	10.53	9.99						
	wordnet	approche Doc2Vec	0.83	1.01	0.97	0.97	0.17	0.18	0.35	0.33	0.06	0.09
		approche Sen2Vec	3.17	3.27	4.22	3.42	0.82	0.75	1.05	0.94	0.04	0.07
		custom tfidf	**7.99**	**8.71**	**14.47**	**13.33**						
semeval-2010	pke	MultipartiteRank	5.72	10.24	7.71	11.95	0.41	0.35	0.39	0.79	1.76	2.52
		PositionRank	3.35	5.87	4.03	6.45	1.10	1.36	1.18	1.48	0.00	0.00
		TextRank	0.74	1.34	0.94	1.22	0.44	0.33	0.19	0.69	0.00	0.00
		TfIdf	7.50	12.54	9.83	13.88	0.85	0.87	1.04	1.45	2.12	3.26
		TopicRank	5.47	8.67	6.91	9.81	0.14	0.07	0.65	0.87	2.32	2.56
		TopicalPageRank	1.40	2.48	1.65	2.32	0.85	0.82	0.42	0.94	0.00	0.00
	text-bison	approche Doc2Vec	6.56	8.21	4.98	5.26	1.58	1.61	4.30	4.20	1.82	2.11
		approche Sen2Vec	**17.17**	**22.12**	**14.65**	**16.18**	**3.99**	**3.91**	**10.53**	**10.77**	**4.12**	**4.86**
		custom tfidf	5.10	7.36	8.10	9.38	0.00	0.22	0.21	0.33	0.75	1.47
	wordnet	approche Doc2Vec	1.60	2.08	1.39	1.57	0.48	0.49	0.83	0.79	0.50	0.52
		approche Sen2Vec	4.50	6.39	4.74	5.26	1.19	1.11	1.51	2.27	0.95	1.60
		custom tfidf	3.58	5.11	5.58	6.65	0.00	0.00	0.00	0.00	0.96	1.14
taln-archives	pke	MultipartiteRank	1.89	2.09	2.35	2.41						
		PositionRank	1.90	2.17	2.35	2.49						
		TextRank	0.96	1.03	1.22	1.19						
		TfIdf	2.16	2.65	2.61	2.97						
		TopicRank	1.82	2.08	2.28	2.41						
		TopicalPageRank	1.13	1.17	1.40	1.34						
	text-bison	approche Doc2Vec	6.22	5.88	5.65	4.77	1.19	1.06	**0.77**	0.75	0.12	0.16
		approche Sen2Vec	**9.74**	**8.68**	9.44	7.71	**1.68**	**1.19**	**0.77**	**0.84**	**0.20**	**0.22**
		custom tfidf	9.50	8.47	**11.31**	**9.38**						
termith-eval	pke	MultipartiteRank	1.69	3.25	2.60	4.51						
		PositionRank	1.61	2.86	2.57	3.96						
		TextRank	0.69	0.92	1.14	1.30						
		TfIdf	2.39	3.42	3.79	4.73						
		TopicRank	1.58	2.98	2.49	4.15						
		TopicalPageRank	0.84	1.12	1.35	1.55						
	text-bison	approche Doc2Vec	8.35	9.85	7.97	7.44	1.91	1.97	3.10	3.56	1.55	1.63
		approche Sen2Vec	**13.31**	**13.29**	**14.46**	12.17	**3.12**	**2.38**	**3.50**	**3.57**	**1.74**	**1.73**
		custom tfidf	9.42	10.81	14.34	**14.37**						

TAB. 3 – *F1-Scores (en %) des modèles.*

5.2 Customisation du TF-IDF

On observe que la technique TfIdf proposée par PKE est moins efficace que notre méthode personnalisée sur KPBiomed, TermITH-Eval, TALN-Archives. Cela s'explique par le fait que les poids utilisés par PKE sont statiques et optimisées pour SemEval-2010 et Inspec.

Ici nous proposons une façon de calculer les poids de façon non-biaisée car nous utilisons uniquement des données externes aux jeux de données.

5.3 Performance des méthodes extractives

Les méthodes extractives du PKE peinent à performer sur les jeux de données avec proportion importante de mots-clés absents comme KPBioMed, TermITH-Eval. Cependant on remarque un très grand bond en performance lorsqu'on passe de $k = 5$ à $k = 10$ mots-clés générés. Il serait intéressant d'étudier l'évolution de la performance en fonction de k.

5.4 Performance des méthodes abstractives

Notre première approche, i.e. inférer le vecteur d'un document entier, renvoie principalement de plus faibles résultats que notre seconde approche, i.e. vote majoritaire pour chaque phrase. Les documents étant des résumés, à l'exception de SemEval-2010, on peut supposer que chaque phrase possède une information importante et indépendante des précédentes, ainsi il semble normal que l'agrégation de tous les vecteurs d'un document renvoie un vecteur peu représentatif de ses thèmes sous-jacents.

De plus, la méthode d'inférence par phrase offre de bons résultats sur l'ensemble des jeux de données et sous-classes PRMU. On notera une sur-performance importante sur SemEval-2010, on doit cela au fait que les documents sont des articles complets et permettent donc un plus large échantillon de phrases pour collecter des prédictions et éviter l'effet de vecteurs aberrants. Effectivement, on observe sur les autres données que la performance stagne ou diminue lorsqu'on augmente le nombre de mots-clés à prédire, cela est dû au plafonnement naturel de l'approche ne pouvant dépasser le nombre de phrases du résumé. Ainsi, nous étudierons les possibilités permettant d'extraire plus de prédictions à l'aide d'une même phrase.

Finalement, on observe des performances très en déça sur la classe Unseen, il serait intéressant d'opter pour un modèle plus complexe comme BERT afin d'observer la différence de modélisation des concepts et index. On notera par ailleurs qu'entre KPBioMed et TermITH-Eval, l'inférence par phrase est plus efficace sur TermITH-Eval. Il est possible, qu'au-delà de la distribution en pourcentage des classes PRMU, le fait qu'il y ait huit fois moins de termes d'index à apprendre permette au modèle une construction plus simple d'un espace vectoriel.

5.5 Construction d'un dictionnaire

Sur l'ensemble des méthodes de notre expérience, l'utilisation de définitions générés par text-bison de Google semble renvoyer les meilleurs résultats. Le large contexte offert par le modèle est un des facteurs à retenir pour cela. Alors que le dictionnaire MeSH, bien que professionnel, offre un cadre d'apprentissage moindre par son décalage entre les lexiques employés par les auteurs et ceux officiellement utilisés par la nomenclature MeSH pour décrire ses index.

5.6 Conclusion

L'indexation est un procédé complexe qui a suscité l'attention des chercheurs. Dans cet article nous avons pu établir une méthode facile à mettre en oeuvre permettant de construire un espace vectoriel partagé par des termes d'index et des mots/phrases. Nous avons aussi pu établir les limites des datasets et modèles présents dans l'état de l'art pour l'indexation.

Références

Berend, G. et V. Vincze (2012). How to evaluate opinionated keyphrase extraction ? In Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis, Jeju, Korea, pp. 99–103. Association for Computational Linguistics.

Boudin, F. (2013). TALN archives : a digital archive of French research articles in natural language processing (TALN archives : une archive numérique francophone des articles de recherche en traitement automatique de la langue) [in French]. In E. Morin et Y. Estève (Eds.), Proceedings of TALN 2013 (Volume 2 : Short Papers).

Boudin, F. (2016). pke : an open source python-based keyphrase extraction toolkit. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : System Demonstrations, Osaka, Japan, pp. 69–73.

Boudin, F. et Y. Gallina (2021). Redefining absent keyphrases and their effect on retrieval effectiveness. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, et Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Online, pp. 4185–4193. Association for Computational Linguistics.

Boudin et al. (2014). Keyphrase generation for scientific document retrieval.

Boudin et al. (2016a). How document pre-processing affects keyphrase extraction performance. In Proceedings of COLING 2016 Workshop on Noisy User-generated Text.

Boudin et al. (2016b). Termith-eval : a french resource for keyphrase extraction.

Gutwin, C., G. Paynter, I. Witten, C. Nevill-Manning, et E. Frank (1999). Improving browsing in digital libraries with keyphrase indexes. Decision Support Systems 27(1), 81–104.

Hasan and Ng (2014). Automatic keyphrase extraction : A survey of the state of the art. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Houbre, M., F. Boudin, et B. Daille (2022). Dataset for biomedical keyphrase generation.

Hulth (2003). Improved automatic keyword extraction given linguistic knowledge. Proceedings of the EMNLP 2003 Conference.

Kim et al. (2010). Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. Proceedings of the 5th International Workshop on Semantic Evaluation.

Kipp, M. (2011). Controlled vocabularies and tags : An analysis of research methods. Proceedings from North American Symposium on Knowledge Organization 3.

Koh, H. Y., J. Ju, H. Zhang, M. Liu, et S. Pan (2022). How far are we from robust long abstractive summarization ? In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 2682–2698. Association for Computational Linguistics.

Kwon, S. (2018). Characteristics of interdisciplinary research in author keywords appearing in korean journals. Malaysian Journal of Library and Information Science 23(2), 77–93.

Larsen and Von Ins (2010). The rate of growth in scientific publication and the decline in coverage provided by science citation index. In Scientometrics 84.

Le, Q. et T. Mikolov (2014). Distributed representations of sentences and documents. In E. P. Xing et T. Jebara (Eds.), Proceedings of the 31st International Conference on Machine Learning, Volume 32 of Proceedings of Machine Learning Research, Bejing, China.

Medelyan and Witten (2006). Thesaurus-based automatic keyphrase indexing.

Mihalcea and Tarau (2004). Textrank : Bringing order into text. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing.

Mikolov et al. (2013). Efficient estimation of word representation in vector space.

Strader, C. (2009). Author-assigned keywords versus library of congress subject headings implications for the cataloging of electronic theses and dissertations. Library Resources and Technical Services 53, 243–250.

Sutskever et al. (2014). Sequence to sequence learning with neural networks.

Tomokiyo and Hurst (2003). A language model approach to keyphrase extraction. Proceedings of the ACL 2003 workshop on Multiword expressions : analysis, acquisition and treatment.

Ye and Wang (2018). Semi-supervised learning for neural keyphrase generation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.

Zha (2002). Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.

Summary

Indexing documents is a major issue in facilitating access to information and scientific monitoring. Due to the costs associated with professional manual document indexing, authors and journals generally rely on automated methods. However, there is a monopoly of so-called extractive methods, i.e. generating indexes that are words present in the source text. With the democratization of neural networks and diverse datasets, it is now possible to perform abstractive generation, i.e. generate keywords that are not present in the source text. This article presents an approach in abstractive and extractive generation of indexes belonging to a controlled vocabulary. We illustrate the application of this method to different types of articles to establish the current limitations in solving this challenge.

Evaluation de petits LLM quantifiés sur une tâche de classification de textes en français

Philippe Suignard*

*EDF R&D
7, boulevard Gaspard Monge 91120 Palaiseau
prenom.nom@edf.fr

Résumé. Cet article présente une évaluation de plusieurs "Large Language Models", quantifiés (codés sur 4bits) de taille 7 milliards réalisée sur une tâche de classification de données textuelles (des avis AlloCiné). Les textes sont d'abord transformés en vecteurs, à l'aide d'un LLM, puis servent à entraîner un classifieur. Le meilleur résultat est obtenu avec le modèle NeuralHermès-2.5-Mistral.

1 Introduction

Il n'est pas nécessaire de présenter les "LLM", les "Large Languages Models" popularisés par ChatGPT [1], devenus omniprésents et très performants. Par contre, ces LLM ont l'inconvénient d'être assez volumineux et de nécessiter des infrastructures importantes pour les faire fonctionner. Pour contourner cette difficulté, une technique appelée "quantification" ("quantization" en anglais) a vu le jour pour diminuer la taille des modèles afin de les rendre utilisables sur des infrastructures plus légères (PC grand public).

Une fonctionnalité des LLM, peut-être moins connue ou moins utilisée, permet la vectorisation des textes, c'est-à-dire la possibilité de convertir un texte ou prompt en un vecteur. Ce vecteur (appelé "embeddings" ou plongements en français), peut ensuite être utilisé pour une tâche de classification de texte, comme le décrit la suite de l'article.

De manière générale, les tailles des LLM varient de 7 à 180 milliards de paramètres, mais les modèles 7b ont montré qu'ils offraient des performances très satisfaisantes. Cet article a pour but de comparer entre eux, plusieurs petits LLM de 7 milliards de paramètres sur une tâche de classification, en utilisant la fonction de plongements de documents.

2 Contexte

Le début de l'année 2023 a vu apparaître des LLM de plus en plus performants mais également de plus en plus gros : 7b, 13b, 40b, 80b ou 180b (b pour milliards de paramètres). On peut citer ici Bloom (Workshop et al., 2022), Falcon (Almazrouei et al., 2023), LLama (Touvron et al., 2023), etc. Cette course à la taille était en partie due au fait que sortir un modèle plus gros offrait la garantie d'obtenir de meilleures performances (Touvron et al., 2023). La

1. https://chat.openai.com/

seconde partie de l'année a été un peu différente : on a plutôt assisté à une diminution des tailles de modèles, avec notamment de nombreux modèles 7b, qui obtenaient les mêmes performances que des modèles de taille supérieure 13b, voire 40b, comme le français Mistral [2] par exemple.

En même temps que la diminution du nombre de paramètres, un autre phénomène est apparu appelé la "quantification des modèles" qui vise à coder les poids des réseaux de neurones sur 16 bits, 8 bits, 4 bits voire moins au lieu de les coder sur 32 bits. Tous les poids ne sont d'ailleurs pas quantifiés de la même manière, mais l'objectif final est de disposer d'un modèle presque 10 fois plus petits tout en étant quasiment aussi performant. Les modèles ainsi quantifiés ou "quantified" sont stockés dans le format GGUF pour "Georgi Gerganov Universal Format" (remplaçant de GGML, devenue obsolète). Dès qu'un nouveau modèle est disponible auprès de la communauté, celui-ci est quantifié dans différentes qualités et rendu public, par des personnes comme "TheBloke" [3] sur des sites comme HuggingFace ou autre [4].

Les modèles quantifiés étant moins volumineux que leur version non quantifiée, ils deviennent utilisables sur des infrastructures plus légères comme des PC grand public. Un grand nombre de développeurs se sont alors emparés du phénomène pour développer des outils permettant d'utiliser ces modèles sur de telles infrastructures. On peut citer ici des outils comme **LoLLMS WebUI** [5], **text-generation-web-ui** [6], **koboldcpp** [7], **LM Studio** [8], **GPT4All** [9], **Faraday** [10], **Llama-cpp-python** [11], **candle** [12] ou **ctransformers** [13]. La plupart de ces outils sont construits au-dessus de **Llama.cpp** [14] et offrent des APIs pour différents langages comme Python, Typescript, Go, C#, Java, Rust ou Go.

Dans ces conditions, quel modèle utiliser pour quoi faire ? Même s'il existe plusieurs "leaderboard", comme celui d'**HuggingFace** [15] pour les performances globales ou cet autre concernant les hallucinations [16] ou encore ce très large panorama qui recense plusieurs approches pour évaluer les LLM (Guo et al., 2023), il est difficile de se fier uniquement aux scores affichés, sachant par exemple qu'il existe de grandes différences entre les langues et les tâches. Pour ces différentes raisons, nous présenterons une démarche pour évaluer des modèles 7b quantifiés sur une tâche de classification de textes rédigés en français.

2. https://mistral.ai/
3. https://huggingface.co/TheBloke
4. Explore the LLMs : https://llm.extractum.io/
5. LoLLMs Web UI : https://github.com/ParisNeo/lollms-webui
6. https://github.com/oobabooga/text-generation-webui
7. https://github.com/LostRuins/koboldcpp
8. https://lmstudio.ai/
9. https://gpt4all.io/index.html
10. https://faraday.dev/
11. https://github.com/abetlen/llama-cpp-python
12. https://github.com/huggingface/candle
13. https://github.com/marella/ctransformers
14. https://github.com/ggerganov/llama.cpp
15. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
16. https://github.com/vectara/hallucination-leaderboard

3 Corpus d'évaluation

Le corpus utilisé pour réaliser les tests qui suivent est un corpus en français, constitué d'avis sur des films, issus du site AlloCiné, extraits par Théophile Blard et rendus accessibles avec plusieurs résultats de classification (CamemBERT, RNN, CNN, LSTM et FastText) (Blard, 2020). Les avis sont étiquetés positif ou négatif. Trois corpus sont fournis : un corpus d'entraînement composé de 160 000 avis (corpus-train), un corpus de validation composé de 20 000 avis (corpus-valid) et un corpus de test composé de 20 000 avis (corpus-test). Dans l'article (Blard, 2020), c'est le modèle CamemBERT qui obtient le meilleur score. Les corpus train et valid servent à fine-tuner le modèle qui est ensuite appliqué sur le corpus-test. Il obtient un excellent score avec 97% de F-mesure.

Dans notre évaluation, nous utilisons uniquement les 2 000 premiers avis issus du corpus-test. Ces avis seront vectorisés à l'aide du LLM à tester. Les features du vecteur sont calculées par le LLM puis utilisées pour entraîner un classifieur intégré au logiciel Weka (Hall et al., 2009). Plusieurs classifieurs sont testés. LibLinear (Fan et al., 2008) est celui qui obtient les meilleurs résultats (Cf Annexe). Les entraînements et tests sont réalisés par validation croisée sur 5 plis (apprentissage sur 4/5 du corpus et test sur 1/5).

Les conditions de tests ne sont pas comparables à celles de l'article précédent : réalisé sur beaucoup moins de documents, sans fine-tuning de modèle et effectué sur une machine moins performante : PC Laptop Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz, avec 32,0 Go de RAM. Néanmoins, nous ne cherchons pas à reproduire les conditions du test de l'article, notre objectif est de tester les modèles entre eux.

4 Modèles testés

Les caractéristiques des modèles testés sont présentées dans le tableau 1. En voici une présentation plus détaillée :

— BERT multilingue (Devlin et al., 2018) : le premier "grand" modèle à avoir connu beaucoup de succès. Avec ses 100 millions de paramètres "seulement", il semble petit aujourd'hui ;
— Llama [17] : disponible en version 7b, 13b et 40b, Llama puis Llama 2, développé par Facebook/Meta est un des modèles les plus utilisés et les plus performants ;
— TinyLlama (Peiyuan Zhang et Lu, 2023) : il s'agit d'un modèle délibérément plus petit (1,1 milliard de paramètres et des embeddings de taille plus petits) qui va être entraîné pendant 90 jours sur 16 cartes graphiques A100 ;
— Mistral [18] : le modèle de la société française éponyme qui se revendique comme le meilleur modèle 7b du marché ;
— Zephyr : proposé par HuggingFace (Tunstall et al., 2023), avec un nom de vent, comme Mistral, est basé sur le « direct preference optimization » pour DPO, (Rafailov et al., 2023) une technique qui fait de l'ombre au RLHF (Christiano et al., 2017). Zephyr était devenu le modèle 7b le plus performant avant d'être détrôné par NeuralHermes-2.5 ;
— Open-Hermes 2.5 Mistral : développé par "Nous Research", une société privée spécialisée dans les modèles de langue et de vision basée à New-York ;

17. https://ai.meta.com/llama/
18. https://mistral.ai/

— Yi : il s'agit d'un modèle développé par la société chinoise Yi, dont le modèle 34b fût, un temps, premier sur le leaderboard d'HuggingFace, devançant le modèle Falcon 180b, un modèle 6 fois plus gros ;
— Neuralhermes-2.5-mistral [19] : réalisé par le français Maxime Labonne, un modèle qui semble très prometteur, le modèle Mistral, fine-tuné avec DPO.

Modèle	Nombre de paramètres	Taille embeddings
BERT multilingue	100m	768
TinyLlama	1,1b	2048
Mistral Instruct	7b	4096
Zephyr	7b	4096
Open-Hermes 2.5 Mistral	7b	4096
Yi	7b	4096
Llama	7b	4096
Neuralhermes-2.5-mistral	7b	4096

TAB. 1 – *Liste des modèles évalués.*

5 Résultats obtenus

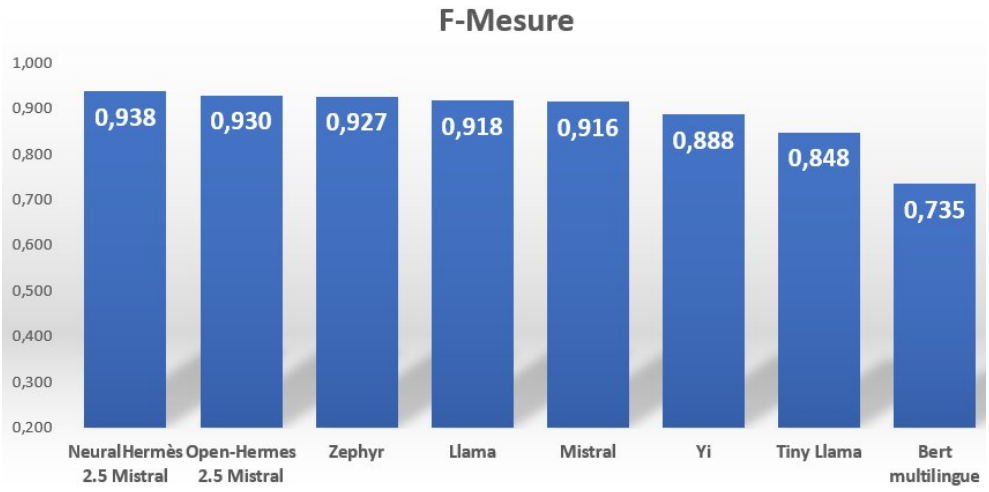

FIG. 1 – *F-Mesure obtenues par les différents modèles.*

Les résultats obtenus avec les différents modèles sont présentés sur la figure 1. Plusieurs constats peuvent être tirés :

19. https://huggingface.co/mlabonne/NeuralHermes-2.5-Mistral-7B

— Les bons résultats obtenus valident l'approche "génération de vecteurs via les LLM, suivie par l'entraînement d'un classifieur", puisque les scores obtenus par tous les LLM sont supérieurs aux résultats obtenus avec Bert ;
— Le meilleur modèle est "NeuralHermès 2.5 Mistral" avec une F-Mesure de 93,8%. Seulement 6,15% des 2000 avis sont mal classés, soit 123 erreurs sur 2000 avis ;
— Les modèles les plus récents obtiennent les meilleurs résultats.

6 Conclusion

Cet article s'est intéressé à l'évaluation de petits modèles LLM 7b quantifiés sur une tâche de classification sur un PC grand public. Pour cette tâche, c'est le modèle "NeuralHermès 2.5 Mistral" qui obtient le meilleur résultat avec une f-mesure de 93,8%. Grâce à la quantification des modèles, ces tests ont pu être réalisés sur un PC grand public, ce qui ouvre des perspectives pour les personnes ou entreprises ne disposant pas d'infrastructures importantes (en terme de serveurs et cartes graphiques).

Le choix avait été fait de comparer entre eux les modèles quantifié en "q4_0" (4bits). Il faudrait tester si les modèles quantifié en "q5_K_M" (d'autres paramètres de quantifications) obtiennent de meilleurs résultats.

D'autres tests comparatifs avec des modèles 13b pourraient être également réalisés.

Références

Almazrouei, E., H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, et G. Penedo (2023). Falcon-40B : an open large language model with state-of-the-art performance.

Blard, T. (2020). French sentiment analysis with bert. `https://github.com/TheophileBlard/french-sentiment-analysis-with-bert`.

Christiano, P. F., J. Leike, T. Brown, M. Martic, S. Legg, et D. Amodei (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems 30*.

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, et C.-J. Lin (2008). Liblinear : A library for large linear classification. *the Journal of machine Learning research 9*, 1871–1874.

Guo, Z., R. Jin, C. Liu, Y. Huang, D. Shi, L. Yu, Y. Liu, J. Li, B. Xiong, D. Xiong, et al. (2023). Evaluating large language models : A comprehensive survey. *arXiv preprint arXiv :2310.19736*.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et I. H. Witten (2009). The weka data mining software : an update. *ACM SIGKDD explorations newsletter 11*(1), 10–18.

Peiyuan Zhang, Guangtao Zeng, T. W. et W. Lu (2023). Tinyllama.

Rafailov, R., A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, et C. Finn (2023). Direct preference optimization : Your language model is secretly a reward model. *arXiv preprint arXiv :2305.18290*.

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. (2023). Llama : Open and efficient foundation language models. *arXiv preprint arXiv :2302.13971*.

Tunstall, L., E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, L. von Werra, C. Fourrier, N. Habib, et al. (2023). Zephyr : Direct distillation of lm alignment. *arXiv preprint arXiv :2310.16944*.

Workshop, B., T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, et al. (2022). Bloom : A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv :2211.05100*.

Annexe : Choix d'un classifieur

Afin d'éviter de tester tous les classifieurs sur tous les LLM, on s'intéresse ici seulement au LLM Mistral_instruct et allons tester les scores obtenus par les différents classifieurs sur ce LLM particulier.

Les documents que l'on souhaite classer (les avis sur des films) sont convertis en vecteurs à l'aide du LLM Mistral_instruct. Chaque avis se trouve ainsi transformé en un vecteur de taille 4096. Les entraînements et tests sont réalisés par validation croisée sur 5 plis (apprentissage sur 4/5 du corpus et test sur 1/5). Les différents classifieurs sont évalués au travers du score de F-Mesure.

Avec une F-Mesure de 0,915, c'est le classifieur LibLinear qui obtient le meilleur score et qui est donc choisi pour évaluer les différents modèles.

Classifieur	F-Mesure
LibLinear	**0,915**
Random Forest	0,892
Régression Logistique	0,848
kNN (k=10)	0,854
J48	0,785
SVM	0,909
NaïveBayes	0,843
LMT	0,906

TAB. 2 – *Liste des classifieurs testés et scores obtenus.*

Summary

This article presents an evaluation of several "Large Language Models", quantified (on 4 bits) of size 7 billion carried out on a textual data classification task (AlloCiné reviews). The

texts are first transformed into vectors, using an LLM, and then used to train a classifier. The best result is obtained with the NeuralHermès-2.5-Mistral model.

Extraction automatique d'entités spatiales imbriquées et de relations spatiales à partir de texte pour la création de graphes de connaissances : Une approche et un jeu de données

Helen Mair Rawsthorne*, Nathalie Abadie*,
Eric Kergosien**, Cécile Duchêne***, Éric Saux****

*LASTIG, Univ Gustave Eiffel, IGN-ENSG, 73 avenue de Paris, F-94165 Saint-Mandé, France
**GERiiCO, Université de Lille, F-59000 Villeneuve d'Ascq, France
***LASTIG, Univ Gustave Eiffel, IGN-ENSG, F-77420 Champs-sur-Marne, France
****IRENav, École navale, F-29240 Brest, France

Résumé. L'extraction automatique d'informations géographiques à partir de texte est essentielle pour exploiter l'ensemble des connaissances spatiales qui n'existent que sous cette forme non structurée. Les éléments clés sont les entités spatiales, leurs types et les relations spatiales entre elles. Structurées en graphe de connaissances géospatial, les connaissances spatiales ambiguës peuvent être désambiguïsées, ce qui facilite considérablement leur accessibilité et réutilisation. Nous présentons une approche pour l'extraction d'entités spatiales imbriquées et de relations spatiales binaires à partir de texte, un jeu de données annoté en français sur le domaine maritime qui peut être utilisé pour entraîner des algorithmes pour les deux tâches d'extraction, ainsi que des résultats de référence pour les deux tâches effectuées individuellement et de bout en bout. Notre approche applique le *Princeton University Relation Extraction system* (PURE), conçu pour l'extraction d'entités génériques plates et de relations binaires génériques, à l'extraction d'entités spatiales imbriquées et de relations binaires spatiales.

1 Introduction

Certaines connaissances spatiales, actuelles ou historiques, n'existent que sous forme de texte. Les guides de voyage, les documents historiques et les publications sur les réseaux sociaux sont quelques exemples de sources de connaissances spatiales non structurées. Les sources textuelles contiennent des connaissances spatiales naturellement hétérogènes : elles peuvent être écrites par différents auteurs, en utilisant un vocabulaire différent, à partir d'un point de vue différent. Elles peuvent par ailleurs couvrir des zones géographiques larges et diverses et contenir des niveaux de détail variés (Ezeani et al., 2023; Jiménez–Badillo et al., 2020; Hu et al., 2019; Kim et al., 2015; Beall, 2010). Pour toutes ces raisons il est difficile d'intégrer dans les modèles de systèmes d'information géographique (SIG) l'information géographique provenant de sources textuelles. L'hypothèse du monde ouvert des technologies du

Web sémantique induit que les graphes de connaissances sont une meilleure solution pour modéliser et stocker les connaissances géographiques extraites de textes hétérogènes, incomplets et imparfaits en langage naturel (Janowicz et al., 2022; Chen et al., 2018; Melo et Martins, 2017; Stadler et al., 2012). Structurées en graphe de connaissances géospatial, les connaissances spatiales ambiguës peuvent être désambiguïsées et liées formellement à des ressources géographiques de référence (telles que DBpedia [1] ou BD TOPO® [2]), ce qui les enrichit de références spatiales directes lorsque c'est possible et facilite considérablement leur accessibilité et réutilisation (Janowicz et al., 2022; Melo et Martins, 2017).

Pendant la population d'un graphe de connaissances, les entités spatiales deviennent des instances de classes ontologiques et les relations spatiales deviennent des propriétés d'objets. Afin de pouvoir attribuer une entité spatiale à sa classe ontologique correspondante, il faut connaître son *type*. Nous faisons l'hypothèse que le nom géographique d'une entité géographique contient souvent un nom commun qui indique son type, tel que *port* dans « Port de Marseille ». Bien que cette hypothèse soit valable pour de nombreuses langues romanes, elle ne s'applique pas à toutes les langues.

L'extraction d'entités spatiales *plates* vise à identifier simplement le nom d'une entité spatiale tel que « Port de Marseille » sans chercher à le définir plus finement, tandis que l'extraction d'entités spatiales *imbriquées* permet de définir plusieurs niveaux d'étiquettes pour la même section de texte. Nous utilisons les étiquettes introduites par Moncla (2015), dont les définitions sont les suivantes :

— geographic feature pour les noms communs qui identifient un *type* d'entité spatiale ;
— name pour les noms propres pures ;
— geographic name pour les noms complets associés à une entité spatiale.

Si nous reprenons notre exemple, l'extraction d'entités spatiales imbriquées viserait donc à identifier « Port de Marseille » comme geographic name, « Port » comme geographic feature et « Marseille » comme name. En extrayant des entités spatiales *imbriquées* au lieu de *plates*, le type de geographic feature de l'entité est déjà connu et une instance de la classe ontologique correspondante peut être créée automatiquement. Dans certains cas, comme dans notre exemple, le name donne une indication de la position géographique de l'entité. Ces deux informations supplémentaires, le type de geographic feature de l'entité et son name, facilitent la tâche de désambiguïsation qui lie l'instance à son entée correspondante dans une ressource géographique de référence (Southall et al., 2011).

En extrayant les relations spatiales entre entités, des assertions de propriétés d'objets peuvent être automatiquement créées entre instances. Ces informations peuvent également être utilisées pour faciliter la désambiguïsation des entités nommées et non nommées, et augmenter la fiabilité des résultats grâce au raisonnement spatial (Paris et al., 2017). Dans le cas où une entité de référence n'existe pas encore, une nouvelle entrée peut être créée dans la ressource géographique, appuyée par les informations relatives aux classes et aux propriétés de l'instance. Le même raisonnement s'applique à la création et à l'enrichissement de gazetiers : en identifiant spécifiquement les types d'entités pendant le processus d'extraction, les entrées de gazetiers peuvent être automatiquement classées ou dotées d'attributs et ainsi être plus facilement désambiguïsées. L'identification et l'extraction des relations spatiales dans lesquelles les entités

1. DBpedia (`https://www.dbpedia.org/`) est une ressource géographique de référence mondiale.
2. La BD TOPO® (`https://geoservices.ign.fr/bdtopo`) est une ressource géographique de référence pour le territoire français.

spatiales prennent part contribue à augmenter le niveau de détail dans les descriptions et les positions géographiques des entrées de gazetiers.

Les textes qui couvrent un environnement international sont susceptibles de contenir des noms géographiques en langues autre que la langue principale du texte. Il est important que cela n'empêche pas le processus d'extraction : les noms géographiques et les types d'entités écrits en autres langues doivent tout de même être identifiés. L'état de l'art en extraction d'information à partir de texte repose sur des modèles de langage à base de réseaux de neurones profonds (Nasar et al., 2021). L'état de l'art en matière d'extraction d'informations à partir de textes repose sur des modèles linguistiques de réseaux neuronaux profonds. Ces modèles peuvent être entraînés dans une ou plusieurs langues et sont appelés respectivement modèles de langage pré-entraînés *monolingues* ou *multilingues*. Une ontologie multilingue peut alors être utilisée pour faciliter la désambiguïsation des entités dont le type est écrit dans d'autres langues (Stadler et al., 2012).

Dans cet article nous présentons une approche pour l'extraction automatique d'entités imbriquées et de relations binaires de texte, qui peut être appliquée aux entités et aux relations *spatiales* si nécessaire (Rawsthorne et al., 2023). Notre approche est une adaptation du *Princeton University Relation Extraction system* (PURE) existant (Zhong et Chen, 2021), qui a été conçu pour l'extraction d'entités génériques plates et de relations binaires génériques. Elle s'insère dans la méthodologie *ATlantis Ontology and kNowledge base development from Texts and Experts* (ATONTE) pour la construction semi-automatique de graphes de connaissances, géospatiaux ou non, à partir de sources textuelles hétérogènes, des connaissances d'experts et des données de référence (Rawsthorne, 2024). Notre approche d'extraction peut être appliquée à des corpus dans n'importe quelle langue (à condition qu'ils contiennent des noms d'entités imbriqués) couvrant n'importe quel domaine, qu'il soit scientifique ou littéraire, historique ou contemporain, de fiction ou non.

Cet article est une version raccourcie et traduite d'un article en anglais par Rawsthorne et al. (2023). Ce dernier contient notamment une présentation complète de l'état de l'art de l'extraction d'entités et de relations à partir de texte en utilisant des approches en apprentissage profond.

2 Contexte d'application

Les travaux présentés dans cet article ont été réalisés dans le cadre d'un projet qui vise à structurer en graphe de connaissances géospatial les informations géographiques contenues dans les *Instructions nautiques*. Les *Instructions nautiques* sont une série d'ouvrages PDF rédigés en français qui sont produites et publiées par le Service hydrographique et océanographique de la Marine (Shom). Elles décrivent l'environnement maritime côtier et donnent des instructions de navigation côtière.

Les *Instructions nautiques* font partie d'une gamme de produits diffusés par le Shom qui servent à la planification d'itinéraires de navigation maritime. D'autres ouvrages du Shom, plus spécialisés, viennent compléter les connaissances sur l'environnement côtier et la navigation, parmi lesquels on trouve *Feux et signaux de brume*, *Radiosignaux*, *Courants de marée* ainsi que l'*Annuaire des marées*. Ils apportent des renseignements qui sont nécessaires à la préparation

d'un itinéraire adapté et sûr. Le type de navire, l'expérience du navigateur, la temporalité [3], les conditions météorologiques et les conditions océanographiques sont également à prendre en considération lors de la planification. Les *Instructions nautiques* contiennent principalement trois types de renseignements (Shom, 2020) :

1. elles donnent des informations complémentaires à celles qui sont affichées sur les cartes marines comme les caractéristiques physiques (couleur, forme, taille, etc.) d'un amer [4] ;
2. elles recensent les informations absentes des cartes marines telles que le climat typique de la zone décrite ;
3. elles donnent des instructions ou des informations à propos de la navigation telles que les routes conseillées, les conditions d'accès aux ports ou encore les réglementations en place.

Les *Instructions nautiques* sont divisées en plusieurs volumes, un par zone de couverture. Une zone de couverture peut être définie soit comme une section de trait de côte entre deux positions sur la côte, soit comme l'ensemble du trait de côte d'une île ou d'un ensemble d'îles. Chaque volume commence avec un chapitre de renseignements généraux. Le plan général du reste de l'ouvrage suit linéairement le trait de côte, chaque chapitre étant dédié à une section du trait de côte. En lisant un chapitre, le lecteur a l'impression d'être emmené le long de la côte par le rédacteur ; chaque repère, danger et autre particularité de l'environnement est décrit, et chaque mouillage, accès de port et entrée de chenal est détaillé. Les consignes mentionnent également les spécificités de la météorologie, la courantologie et la réglementation locales. Des photographies montrant les amers et les ports notables sont intercalées dans le texte. Elles illustrent également le positionnement relatif des différentes entités géographiques et doivent conforter le lecteur dans la représentation qu'il se fait de son environnement.

3 Plan de l'article et disponibilité du code

Dans la section 4 nous présentons le jeu de données *coAsTaL mAritime NavigaTion InstructionS* (ATLANTIS) [5], un nouveau jeu de données de référence en langue française, annoté manuellement, qui porte sur le domaine maritime (Rawsthorne et al., 2023). Il peut être utilisé pour entraîner des algorithmes pour l'extraction d'entités spatiales imbriquées et de relations spatiales binaires à partir de texte. Nous présentons également dans la section 4 le jeu de données TextMine'24 [6], qui est un sous-ensemble du jeu de données ATLANTIS et contient des annotations d'entités spatiales imbriquées.

Dans la section 5 nous présentons une approche pour l'extraction d'entités spatiales imbriquées et de relations spatiales binaires à partir de texte. Notre approche est une adaptation de PURE (Zhong et Chen, 2021), qui gère uniquement l'extraction d'entités plates génériques et de relations génériques. Nous la rendons applicable à l'extraction d'entités *spatiales* et *imbriquées*, et de relations *spatiales* en utilisant le code mis à disposition par Zhong et Chen (2021) ainsi qu'un format d'annotation modifié.

3. Une temporalité est une condition locale qui est dépendante sur le temps, par exemple l'heure, le mois de l'année ou le saison.

4. Un amer est un « objet remarquable situé à un endroit fixe sur la terre et pouvant être utilisé pour déterminer un emplacement ou une direction. » Traduit du Hydrographic Dictionary Working Group (2019).

5. `https://github.com/umrlastig/atlantis-dataset`

6. `https://www.kaggle.com/competitions/defi-textmine-2024`

Nous présentons dans la section 6 des résultats de référence pour le jeu de données ATLANTIS, obtenus grâce à l'approche décrite dans la section 5, pour les tâches d'extraction d'entités spatiales imbriquées, d'extraction de relations spatiales binaires, et d'extraction combinée d'entités et de relations spatiales de bout en bout. Les entités et relations spatiales extraites de notre corpus pourraient directement enrichir des ressources géographiques de référence ou des gazetiers, qui pourraient être utilisés pour des applications dans des domaines tels que l'hydrographie, la navigation maritime ou les humanités.

4 Préparation du jeu de données

Le le jeu de données ATLANTIS est composé d'extraits de chacun des 15 volumes des *Instructions nautiques* dont nous disposions. Nous avons extrait le texte des PDF en utilisant *pdfminer.six* [7]. Le jeu de données contient 101 400 jetons.

Nous avons annoté notre jeu de données à la main en utilisant le *brat rapid annotation tool* [8]. Il permet la création d'annotations d'étiquettes imbriquées ainsi que la création de liens entre elles, avec un sens et une étiquette propre à chaque lien (Stenetorp et al., 2012). Le schéma d'annotation, décrit ci-dessous, a été concu et validé par l'ensemble des auteur·e·s. Une auteure a réalisé l'annotation du jeu de données et ensuite des extraits ont été vérifiés et validés par tout·e·s les auteur·e·s.

Étant donné que nous souhaitions réaliser l'extraction d'entités spatiales *imbriquées* afin de capturer simultanément le nom complet de l'entité spatiale ainsi que son type et son nom, nous avons mis en œuvre une approche d'étiquetage imbriqué en utilisant les étiquettes définies dans la section 1. Tout jeton peut être annoté de zéro ou d'une étiquette geographic feature ou name. Un jeton ne peut pas être annoté à la fois par une étiquette geographic feature et par une étiquette name. Un jeton ne peut pas être annoté uniquement avec une étiquette name. Un jeton annoté avec une étiquette name doit également être annoté une ou plusieurs fois avec une étiquette geographic name. Tout jeton, déjà étiqueté ou non, peut être annoté zéro ou plusieurs fois avec une étiquette geographic name.

Un très grand nombre de types différents de relations spatiales sont utilisés dans notre corpus. Nous avons donc décidé de nous limiter à l'extraction de ceux qui seraient les plus utiles au cours du processus de désambiguïsation.

Les directions cardinales sont très utilisées dans la navigation parce que les relations spatiales qui les utilisent sont construites en utilisant un cadre de référence absolu, ce qui signifie qu'il n'est pas nécessaire de préciser un point de vue spécifique (Levinson, 1996). Nous avons choisi d'extraire les relations spatiales qui utilisent les directions cardinales en raison de leur utilisation fréquente et de leur absence d'ambiguïté. Cela représente 16 types de relations au total : quatre qui utilisent les directions cardinales (N, E, S, W), quatre qui utilisent les directions intercardinales (NE, SE, SW, NW) et huit qui utilisent les directions intercardinales secondaires (NNE, ENE, ESE, SSE, SSW, WSW, WNW, NNW). Dans notre corpus, ces relations spatiales sont toujours désignées par ces 16 abréviations à une, deux ou trois lettres, par exemple « le port est au NW de la ville » ou « la tour est à l'ESE du château ». Les 16 étiquettes qui correspondent à ces relations spatiales sont au format « est au XYZ de », ou « XYZ » est une des 16 abréviations des directions cardinales.

7. https://github.com/pdfminer/pdfminer.six
8. http://brat.nlplab.org

Nous avons identifié trois autres types de relations spatiales permettant de capturer plus d'informations sur les entités spatiales spécifiques à notre domaine qui ne sont souvent pas nommées dans le corpus ou qui sont susceptibles d'être absentes des ressources géographiques de référence telles que les marques de navigation (bouées, balises, etc.), les rochers ou encore les bancs de sable. Tout d'abord, l'étiquette « *is off the coast of* » est utilisée lorsqu'il est indiqué qu'une entité spatiale est située au large ou dans les eaux côtières d'une autre. Elle est donc fréquemment utilisée pour localiser des entités spatiales isolées. Ce type de relation spatiale, qui est également construit à partir d'un cadre de référence absolu, est toujours désigné par la même expression dans notre corpus : « est au large de ». Deuxièmement, l'étiquette « *is marked by* » est utilisée pour toute entité spatiale qui est marquée ou signalée par une autre entité délibérément placée, souvent une marque de navigation, soit lorsque la première représente un danger pour la navigation, soit lorsqu'elle permet un passage en toute sécurité : « Son musoir est marqué par un feu. » (Shom, 2021). Cette relation indique une proximité entre les deux entités et est exprimée de différentes manières dans notre corpus : « est marqué par » peut être exprimé alternativement par « est signalé par » ou « est indiqué par ». Troisièmement, l'étiquette « *is an element of* » indique une relation topologique qui inclut des entités situées *dans* ou *sur* d'une autre de manière à ce qu'une vue à vol d'oiseau montre l'empreinte spatiale de l'une comme étant à l'intérieur ou en partie à l'intérieur de l'autre. Cette relation est exprimée de différentes manières dans notre corpus, y compris de manière implicite, et inclut rarement le mot « élément ». Par exemple, « l'île porte un phare », « le feu est établi sur le quai » et « les hauts-fonds de la baie » indiquent tous une relation de type « *is an element of* ».

Toutes les annotations de relations doivent relier deux annotations d'entités, soit des étiquettes geographic feature, soit des étiquettes geographic name. Toutes les annotations de relation doivent avoir une direction. Au lieu de dupliquer les étiquettes de relation pour tenir compte de leurs inverses et de créer des annotations de relations orientées qui vont toujours dans le sens du texte (« A $\rightarrow$ *is marked by* $\rightarrow$ B » et « C $\rightarrow$ *marks* $\rightarrow$ D »), nous avons créé une version pour chaque étiquette et nous autorisons les annotations de relations orientées qui vont dans l'une ou l'autre direction (« A $\rightarrow$ *is marked by* $\rightarrow$ B » et « C $\leftarrow$ *is marked by* $\leftarrow$ D »).

La figure 1 montre une phrase des *Instructions nautiques* annotée selon notre schéma d'annotation d'entités spatiales imbriquées et de relations spatiales binaires. L'association de l'étiquetage spécifique du geographic feature « *ras* » (« cap ») au sein du geographic name et des valeurs d'étiquettes multilingues dans une ontologie signifie que cette entité spatiale pourrait être automatiquement instanciée dans la bonne classe, quelle que soit la langue dans laquelle la geographic feature est écrite (dans ce cas l'arabe romanisé). La figure 2 montre des triplets *Resource Description Framework* (RDF) qui pourraient être automatiquement construits à partir de cette phrase selon l'ontologie ATLANTIS (Rawsthorne et al., 2022).

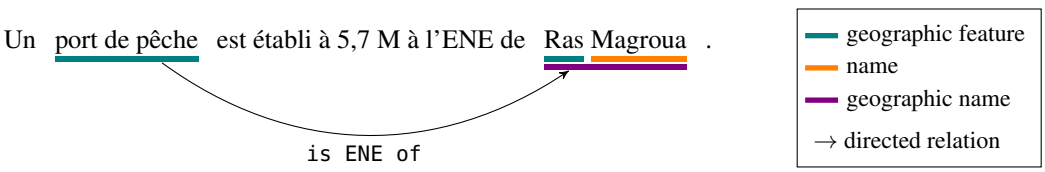

FIG. 1 – *Une phrase des Instructions nautiques annotée selon notre schéma d'annotation d'entités spatiales imbriquées et de relations spatiales binaires (Shom, 2021).*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ent: <http://data.shom.fr/id/spatialentity/> .
@prefix atln: <http://data.shom.fr/def/atlantis#> .

ent:0001 rdf:type atln:FishingPort ; # entit  num ro 1 est un port de p che
    atln:isENEof ent:0002 . # entit  num ro 1 est ENE d'entit  num ro 2

ent:0002 rdf:type atln:Cape ; # entit  num ro 2 est un cap
    rdfs:label "Ras Magroua" . # entit  num ro 2 s'appelle Ras Magroua
```

FIG. 2 – *Triplets RDF construits à partir de l'information annotée dans la figure 1 selon l'ontologie ATLANTIS (Rawsthorne et al., 2022).*

Nous avons divisé notre jeu de données annoté en trois parties : entraînement, développement et test, en veillant à respecter un rapport de 80 : 10 : 10 entre le nombre total de jetons et le nombre d'étiquettes d'entités. Nous avons également cherché à ce que du texte couvrant chaque zone géographique soit présent dans les trois parties. Notre jeu de données de 101 400 jetons contient 16 777 étiquettes d'entités (qui peuvent s'étendre sur plusieurs jetons) et 3 051 étiquettes de relations (qui relient exactement deux étiquettes d'entités dans un sens donné). Au total, 18 030 jetons sont annotés avec au moins une étiquette d'entité, ce qui correspond à près d'un jeton sur cinq. La composition du jeu de données et affiché dans le tableau 1. La distribution des étiquettes peut être consulté sur le dépôt de notre jeu de données [9].

	Entraînement	Développement	Test	Total
Jetons	83 851	8 156	9 393	101 400
Jetons sans étiquette	69 200	6 507	7 663	83 370
Jetons avec étiquette d'entité	14 651	1 649	1 730	18 030
Étiquettes d'entité	13 582	1 476	1 719	16 777
Étiquettes de relation	2 507	222	322	3 051

TAB. 1 – *Nombre de jetons et d'étiquettes dans chaque partie du jeu de données. Une étiquette d'entité peut s'étendre sur un jeton ou plus.*

Nous avons converti notre jeu de données du format *brat* au format *JSON Lines* (JSONL) [10] requis pour PURE à l'aide d'un script Python [11]. La figure 3 montre la même phrase annotée que dans la figure 1 convertie en valeur JSON dans ce format. Dans notre cas, chaque valeur JSON correspond à un paragraphe des *Instructions nautiques* et contient une liste de phrases (dont chacune est une liste de jetons), une liste des annotations d'entités (les numéros des deux jetons frontaliers + l'étiquette), et une liste des annotations de relations (les numéros des deux fois deux jetons frontaliers + l'étiquette). Ce format d'annotation imbriquée, qui permet à tout jeton d'être annoté avec zéro ou plusieurs étiquettes, rend possible l'extraction d'entités imbriquées sans utiliser du *joint labelling* (Rawsthorne et al., 2023; Agrawal et al., 2022).

9. https://github.com/umrlastig/atlantis-dataset
10. Un fichier JSONL contient une valeur JSON valide sur chaque ligne.
11. https://github.com/dwadden/dygiepp/blob/master/scripts/new-dataset/brat_to_input.py

```
{ "doc_key": "d6_phrase_exemple",
  "dataset": "atlantis",
  "sentences": [["Un", "port", "de", "pêche", "est", "établi", "à", "5,7", "M"
      , "à", "l'", "ENE", "de", "Ras", "Magroua", "."]],
  "ner": [[[1, 3, "geogFeat"], [13, 13, "geogFeat"], [14, 14, "name"], [13, 14
      , "geogName"]]],
  "relations": [[[1, 3, 13, 14, "isENEof"]]] }
```

FIG. 3 – *Une ligne d'un fichier JSONL formaté pour PURE. Elle contient le texte et les annotations illustrés dans la figure 1.*

5 Entraînement des modèles

PURE (Zhong et Chen, 2021) entraîne indépendamment deux encodeurs de base à partir de modèles de langage profonds pré-entraînés existants : l'un pour identifier et étiqueter les entités, et l'autre pour identifier les paires reliées d'entités et classer la relation entre elles. Nous appelons le premier modèle le *modèle entité* et le second le *modèle relation*. PURE permet également de réguler la taille de la fenêtre contextuelle W, c'est-à-dire la quantité de contexte inter-phrases qui est mise à la disposition du modèle. Le contexte disponible lors du traitement d'une phrase donnée s'étend de $(W-n)/2$ mots à gauche de la phrase à $(W-n)/2$ mots à droite, où n est le nombre de mots dans la phrase. Une perte d'entropie croisée est utilisée pour les deux modèles.

Pour les encodeurs de base, nous avons utilisé *bert-base-french-europeana-cased* comme modèle *Bidirectional Encoder Representations from Transformers* (BERT) monolingue français et *bert-base-multilingual-cased* comme modèle BERT multilingue. Nous avons utilisé les hyperparamètres par défaut fournis par (Zhong et Chen, 2021) et nous avons fait des expériences avec plusieurs tailles de fenêtres contextuelles, dans les plages des valeurs par défaut, pour le modèle entité et le modèle relation. Pour le modèle entité, nous avons utilisé des fenêtres contextuelles de 0, 50, 100, 150, 200 et 248 ($W = 250$ a dépassé le mémoire GPU disponible) et pour le modèle relation, nous avons utilisé des fenêtres contextuelles de 0, 50 et 100. Nous n'avons apporté aucune autre modification au code publié par Zhong et Chen (2021). Nous avons entraîné et évalué les deux encodeurs de base BERT pour l'extraction d'entités spatiales imbriquées grâce à notre format d'annotation imbriquée, puis nous avons entraîné et évalué séparément les deux mêmes encodeurs de base pour l'extraction de relations spatiales. Pendant l'entraînement, les modèles ont eu accès aux annotations manuelles (*gold*) des entités. Nous avons effectué deux évaluations différentes sur les modèles de relations : l'une avec les annotations d'entités manuelles et l'autre avec les entités prédites. Les relations prédites à partir des annotations d'entités manuelles donnent uniquement une évaluation du processus d'extraction de relations. Les relations prédites à partir des entités prédites donnent une évaluation du processus d'extraction d'entités et de relations de bout en bout. Pour chaque configuration, nous avons entraîné et évalué cinq modèles individuels en utilisant différentes valeurs de graines aléatoires et nous avons calculé la moyenne arithmétique et l'écart-type des scores F1 micro obtenus.

6 Résultats et analyse

Les scores F1 pour les trois tâches avec différentes tailles de fenêtres contextuelles sont affichés dans le tableau 2. La précision globale, le rappel et les scores F1 par étiquette peut être consulté sur le dépôt de notre jeu de données [12].

W	Entités		Relations		E. + R. (de bout en bout)	
	Mono.	Multi.	Mono.	Multi.	Mono.	Multi.
0	91.1 ± 0.3	91.9 ± 0.2	**64.2** ± 2.2	**63.2** ± 1.0	**63.9** ± 2.2	**63.2** ± 1.2
50	92.1 ± 0.2	92.3 ± 0.3	64.2 ± 1.4	63.0 ± 1.7	63.8 ± 1.4	63.1 ± 1.7
100	91.9 ± 0.2	92.3 ± 0.2	63.7 ± 0.7	62.9 ± 0.7	63.6 ± 0.7	62.9 ± 0.8
150	91.9 ± 0.2	92.2 ± 0.2	-	-	-	-
200	92.0 ± 0.2	**92.3** ± 0.2	-	-	-	-
248	**92.2** ± 0.2	92.3 ± 0.2	-	-	-	-

TAB. 2 – *Score F1 micro moyen avec écart-type sur cinq exécutions pour différentes tailles de fenêtres contextuelles pour : l'extraction d'entités, l'extraction de relations à partir d'annotations manuelles (gold) d'entités, et l'extraction d'entités et de relations de bout en bout [e2e] à partir des meilleures annotations d'entités prédites ($W = 248$ pour le modèle monolingue et $W = 200$ pour le modèle multilingue). Pour chaque tâche, le score F1 le plus élevé parmi toutes les tailles de fenêtres contextuelles pour chaque encodeur de base est en* ***gras****, et le score F1 global le plus élevé parmi toutes les tailles de fenêtres contextuelles et les deux encodeurs de base est souligné.*

Pour l'extraction d'entités, nos expériences montrent que la mise à disposition du contexte inter-phrases pendant l'entraînement et la prédiction améliore les scores F1 micro pour les deux modèles, et que le modèle multilingue BERT surpasse légèrement le modèle monolingue français BERT pour toutes les tailles de fenêtre contextuelle, avec son score F1 micro moyen le plus élevé étant de 92,3 lorsque $W = 200$ (tableau 2). Nous attribuons ce contraste dans les résultats par rapport à ceux de la littérature (Rawsthorne et al., 2023) à une caractéristique de notre jeu de données : bien que la langue principale du texte soit le français, il contient des mots provenant d'un grand nombre d'autres langues. Les mots en question sont principalement des geographic feature qui font partie de geographic name, ce qui signifie qu'ils doivent être identifiés et correctement étiquetés par le modèle entité. Dans ces cas, le modèle monolingue perd son avantage par rapport au modèle multilingue, car ce dernier est capable de comprendre le sens sémantique d'une plus grande proportion des mots de l'ensemble de données.

7 Conclusion et perspectives

Nous avons discuté et souligné l'importance d'une extraction fiable d'entités spatiales imbriquées et de relations spatiales pour la construction de graphes de connaissances géospatiales ou de gazetiers à partir de textes et pour la désambiguïsation d'entités spatiales. Nous avons

12. https://github.com/umrlastig/atlantis-dataset

présenté un nouveau jeu de données annoté en langue française pour ces deux tâches d'extraction, spécifique au domaine maritime. Nous avons fourni des résultats de référence pour notre propre jeu de données et avons ainsi démontré que PURE (Zhong et Chen, 2021), une approche existante pour l'extraction générique d'entités et de relations binaires à partir de textes, peut être utilisée pour extraire des entités *imbriquées*. Ceci a été réalisé en entraînant un encodeur BERT avec des annotations imbriquées, sans utiliser du *joint labelling*. Nous avons également montré que PURE est une approche de base adaptée à l'extraction d'entités *spatiales* et de relations *spatiales* spécifiques à un domaine. Nos résultats révèlent que le modèle multilingue BERT est légèrement plus performant que le modèle monolingue français BERT pour l'extraction d'entités, avec un score F1 micro moyen de 92,3, tandis que pour l'extraction de relations et l'extraction d'entités et de relations de bout en bout, le modèle monolingue français BERT est légèrement plus performant, avec des scores F1 micro moyens de 64,2 et 63,9 respectivement. Nos résultats montrent que la mise à disposition d'informations contextuelles inter-phrases lors de l'entraînement et de la prédiction favorise l'extraction d'entités mais entrave l'extraction de relations.

Une première perspective concerne la réduction du temps nécessaire pour annoter le jeu de données d'entraînement. Pour ce faire, on pourrait utiliser un outil d'annotation qui propose soit un étiquetage assisté par apprentissage machine, soit une annotation automatisée en fonction d'un ensemble de mots clés défini par l'utilisateur. Il faudrait évaluer et comparer les annotations produites à l'aide des différents outils afin de vérifier que la qualité ne diminue pas par rapport aux annotations que nous avons réalisées de manière entièrement manuelle. Dans le but d'augmenter la fiabilité du jeu de données annoté, on pourrait adopter une approche d'annotation à plusieurs et calculer l'accord inter-annotateurs. On pourrait aussi produire automatiquement des données synthétiques sous la forme d'un texte annoté par le biais d'expressions régulières. Il serait également possible de combiner un apprentissage non supervisé avec un apprentissage supervisé en utilisant un plus petit jeu de données annoté manuellement lors de l'étape d'extraction. Si les résultats sont identiques ou meilleurs que ceux obtenus avec notre approche actuelle, l'apprentissage non supervisé pourrait être intégré à notre méthodologie et le volume de données annotées manuellement nécessaire pourrait être réduit. Une autre perspective concerne l'élargissement de l'approche d'extraction d'information à partir de texte afin de permettre la prise en compte de plusieurs types d'entités simultanément et la gestion de relations n-aires (ternaires ou plus). Ceci permettrait d'extraire des relations impliquant plus de deux entités telles que la relation spatiale *entre*, comme par exemple dans la phrase « la bouée est entre l'île Est et l'île Ouest ».

Références

Agrawal, A., S. Tripathi, M. Vardhan, V. Sihag, G. Choudhary, et N. Dragoni (2022). BERT-Based Transfer-Learning Approach for Nested Named-Entity Recognition Using Joint Labeling. *Applied Sciences 12*(3), 976.

Beall, J. (2010). Geographical research and the problem of variant place names in digitized books and other full-text resources. *Library Collections, Acquisitions, & Technical Services 34*(2-3), 74–82.

Chen, H., M. Vasardani, S. Winter, et M. Tomko (2018). A Graph Database Model for Knowledge Extracted from Place Descriptions. *International Journal of Geo-Information 7*(6), 221.

Ezeani, I., P. Rayson, et I. Gregory (2023). Extracting Imprecise Geographical and Temporal References from Journey Narratives. In *Proceedings of Text2Story — Sixth Workshop on Narrative Extraction From Texts*, Volume 3370, Dublin, Ireland. CEUR Workshop Proceedings.

Hu, Y., C. Deng, et Z. Zhou (2019). A Semantic and Sentiment Analysis on Online Neighborhood Reviews for Understanding the Perceptions of People toward Their Living Environments. *Annals of the American Association of Geographers 109*(4), 1052–1073.

Hydrographic Dictionary Working Group (2019). S-32 IHO Hydrographic Dictionary.

Janowicz, K., P. Hitzler, W. Li, D. Rehberger, M. Schildhauer, R. Zhu, C. Shimizu, C. K. Fisher, L. Cai, G. Mai, J. Zalewski, L. Zhou, S. Stephen, S. Gonzalez, B. Mecum, A. Lopez-Carr, A. Schroeder, D. Smith, D. Wright, S. Wang, Y. Tian, Z. Liu, M. Shi, A. D'Onofrio, Z. Gu, et K. Currier (2022). Know, Know Where, KnowWhereGraph : A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Magazine 43*(1), 30–39.

Jiménez–Badillo, D., P. Murrieta–Flores, B. Martins, I. Gregory, M. Favila-Vázquez, et R. Liceras-Garrido (2020). Developing Geographically Oriented NLP Approaches to Sixteenth–Century Historical Documents : Digging into Early Colonial Mexico. *Digital Humanities Quarterly 14*(4).

Kim, J., M. Vasardani, et S. Winter (2015). Harvesting large corpora for generating place graphs. Volume 12, Santa Fe, NM, USA.

Levinson, S. C. (1996). Language and Space. *Annual Review of Anthropology 25*, 353–382.

Melo, F. et B. Martins (2017). Automated Geocoding of Textual Documents : A Survey of Current Approaches. *Transactions in GIS 21*(1), 3–38.

Moncla, L. (2015). *Automatic reconstruction of itineraries from descriptive texts*. PhD, Université de Pau et des Pays de l'Adour, Pau.

Nasar, Z., S. W. Jaffry, et M. K. Malik (2021). Named Entity Recognition and Relation Extraction : State-of-the-Art. *ACM Computing Surveys 54*(1), 20 :1–20 :39.

Paris, P.-H., N. Abadie, et C. Brando (2017). Linking Spatial Named Entities to the Web of Data for Geographical Analysis of Historical Texts. *Journal of Map & Geography Libraries 13*(1), 82–110.

Rawsthorne, H. M. (2024). *Creation of Geospatial Knowledge Graphs From Heterogeneous Sources*. PhD, Université Gustave Eiffel, Champs-sur-Marne.

Rawsthorne, H. M., N. Abadie, E. Kergosien, C. Duchêne, et Saux (2022). ATLANTIS : Une ontologie pour représenter les Instructions nautiques. In *Journées Francophones d'Ingénierie des Connaissances (IC) Plate-Forme Intelligence Artificielle (PFIA 2022)*, Saint-Étienne, France, pp. 154–163.

Rawsthorne, H. M., N. Abadie, E. Kergosien, C. Duchêne, et Saux (2023). Automatic Nested Spatial Entity and Spatial Relation Extraction From Text for Knowledge Graph Creation : A Baseline Approach and a Benchmark Dataset. In *7th ACM SIGSPATIAL International*

Workshop on Geospatial Humanities (GeoHumanities '23), November 13, 2023, Hamburg, Germany, Hamburg, Germany. Association for Computing Machinery.

Shom (2020). Guide de rédaction des Instructions Nautiques du Shom. Procédure spécifique, Shom.

Shom (2021). *Instructions nautiques. D6 : Mer Méditerranée, côtes d'Afrique et du Levant [Version à jour au 13 octobre 2021].* Brest, France.

Southall, H., R. Mostern, et M. L. Berman (2011). On historical gazetteers. *International Journal of Humanities and Arts Computing 5*(2), 127–145.

Stadler, C., J. Lehmann, K. Höffner, et S. Auer (2012). LinkedGeoData : A core for a web of spatial open data. *Semantic Web 3*(4), 333–354.

Stenetorp, P., S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, et J. Tsujii (2012). brat : a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, pp. 102–107. Association for Computational Linguistics.

Zhong, Z. et D. Chen (2021). A Frustratingly Easy Approach for Entity and Relation Extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, Online, pp. 50–61. Association for Computational Linguistics.

Summary

Automatically extracting geographic information from text is the key to harnessing the vast amount of spatial knowledge that only exists in this unstructured form. The fundamental elements of spatial knowledge include spatial entities, their types and the spatial relations between them. Structuring the spatial knowledge contained within text as a geospatial knowledge graph, and disambiguating the spatial entities, significantly facilitates its reuse. We present a baseline approach for nested spatial entity and binary spatial relation extraction from text, a new annotated French-language benchmark dataset on the maritime domain that can be used to train algorithms for both extraction tasks, and benchmark results for the two tasks carried out individually and end-to-end. Our approach involves applying the Princeton University Relation Extraction system (PURE), made for flat, generic entity extraction and generic binary relation extraction, to the extraction of nested, spatial entities and spatial binary relations.

CIAD System for Geographical Entity Detection at TextMine'24

Pauline Armary*,** Cheikh-Brahim El-Vaigh, * Ouassila Labbani Narsis* Christophe Nicolle*

* CIAD, UMR 7533, *Université de Bourgogne, UB*,
64 rue de Sully 21000 Dijon
firstname.lastname@u-bourgogne.fr
**Anabasis-Assets
https://www.anabasis-assets.com/fr/

Abstract. This paper outlines CIAD's approach to Named Entity Recognition (NER) in a corpus of nautical instruction. Employing a conventional entity detection approach, our system tackles the NER task through token classification. We handled this classification task using two approaches: 1. NER is performed using different BERT models (BERT-base, Tiny-BERT, and CamemBERT); 2. We also used a GCN-based token classification where a graph is built connecting words in the same context. Our results suggest that within these token classification setups, our models are bounded to an accuracy of around 92% on the test data regardless of the complexity of the used BERT model.

1 Introduction

Named Entity Recognition (NER) is a fundamental task in natural language processing that involves the identification and classification of named entities within textual data (Liu et al., 2022). Named entities are specific entities such as persons, organizations, locations, dates, and more, which carry significance in the given context. NER plays a crucial role in information extraction, and the process typically involves analyzing and annotating text to locate and classify named entities (Liu et al., 2022; Wang et al., 2022).

NER has undergone significant advancements, with various approaches and techniques employed for identifying and classifying named entities in text. Initially, rule-based NER systems were used relying on predefined linguistic rules and patterns to identify named entities in text (Collins and Singer, 1999). Then, machine learning algorithms, such as Conditional Random Fields (CRF) and Hidden Markov Models (HMM), have been applied to learn patterns and features from labeled data (Patil et al., 2020; Morwal et al., 2012). The advent of deep learning models, especially Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Transformers has ushered in a new era of state-of-the-art performance for NER tasks (Peters et al., 2018; Chiu and Nichols, 2016). Finally, with the success of transformers despite their huge training cost, fine-tuning pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) and GPT (Generative Pre-trained Transformer) has become a common practice.

In the context of the initiative TextMine 2024 [1], focusing on the identification of geographical entities through named entity processing from a French nautical instruction corpus (Rawsthorne et al., 2024), the application of NER becomes crucial for extracting entities from the text. TextMine 2024 is a EGC 2024 [2] evaluation Lab. However, the inherent challenges of the geographical entities context make the NER task more intricate, given that the texts are of a small size and noisy. Moreover, the target entities are not fine-grained types but rather ambiguous. For example, in Tab. 1 which shows the statistics of the different entity types, the annotation *name* refers to a proper noun being it a Person, a Place, or an Organization, while in the standard NER these three categories are separated. Moreover, while NER is often seen as a multi-class classification task, in this evaluation the NER is both a multi-class and a multi-label classification problem, and for example, in Tab. 1, we have 2123 tokens that are both *name* and *geogName*.

This paper proposes two approaches for the NER task in Sec. 2. We first studied several pre-trained BERT models (Devlin et al., 2018) that we fine-tuned on the TextMine corpus and compared their accuracy (see Sec. 2.1). We also devised a graph convolutional network (GCN) (Kipf and Welling, 2017) based approach that considers the tokens in the corpus as nodes and connects these nodes based on their co-occurrence in the same context. Moreover, labels are also seen as nodes and a token is connected to its labels from the train dataset. We then tackle NER as a node classification task using the standard GCN architecture (see Sec. 2.2). The code of our system is on GitHub [3].

Type	geogFeat	geogFeat geogName	geogName	name	name geogName	tokens without label
annotations in the corpus	4167	1469	4490	2118	2123	32668

TAB. 1 – *The different annotations present in the corpus.*

2 CIAD's Systems

2.1 Fine-tuning BERT models

Fine-tuning a BERT model for a specific task is nowadays the state-of-the-art option for several NLP tasks, especially for NER. A BERT (Devlin et al., 2018) model is a Large Language Model (LLM) trained with an Encoder architecture on a very large corpus of documents. As with any LLM, the principle is to transform each word of the corpus into a vector within a large-dimensional space. This vector is reduced using a neural network architecture by optimizing the result to a specific task while reducing the number of dimensions of the vector space.

As part of the Transformer family, BERT uses an Encoder architecture trained on a masked-word prediction task. The Encoder Neural Network is trained to predict what is the correct word which was masked within a sentence. As such, the model uses the overall context surrounding the word to create a vector representing its meaning. For NER task, the model is fine-tuned as a Token Classification or Sequence classification task, with a corpus associating a label of each specific kind of entity (Person, Organization, Place, etc) to each word.

[1] https://textmine.sciencesconf.org/resource/page/id/8
[2] https://iutdijon.u-bourgogne.fr/egc2024/
[3] https://github.com/elvaigh/textmine-ciad

TAB. 2 – *Comparison of the different models.*

	1	2	3	4	5
Model	Bert-base NER	CamemBert NER	GPT2	TinyBert	GCN
Accuracy	0.92771	0.92679	0.91931	0.91508	0.77936

For predicting the label of the corpus, we used a pre-trained BERT model on Sequence Classification for NER, which was pre-trained for recognizing different kinds of entities within the text. We fine-tuned the model to recognize the specific label of our corpus ("geoFeat", "geoName", "name"). We choose a Sequence Classifier as the data is presented as an ordered list of tokens. We fined-tuned the following models: BERT-Base-NER (BERT trained for NER in English), CamemBERT-NER (French BERT trained for NER in French), TinyBERT (refinement of the BERT model), and a NER model using GTP2 within a prompt-based learning setup.

2.2 Using GCN for NER

The Graph Convolutional Network (GCN) architecture was first proposed by (Kipf and Welling, 2017) for node classification. In the context of Natural Language Processing, the text is first transformed into a graph representing the connections between the words that appear in the same context (a window of size 2). For NER, the words are also associated with their labels, each label and token representing a node in the graph. This architecture is often presented as more resilient to multi-label classification, as a node can easily be associated with multiple labels. Within the GCN, an embedding of each node is learned and a softmax layer is used to perform classification based on these embeddings. We used the GCN model to our classification task for geographical entities.

3 Experiments

Experimental validation was conducted on the TextMine 2024 French corpus to assess the quality of our system. We evaluate our system using the accuracy metric. The description of the TextMine corpus can be found in the challenge website. [4]

We discuss hereunder the performance of NER classifiers that we described in Sec. 2.1 and Sec. 2.2. We gathered in Tab. 2 the accuracy of the different classifiers on the test dataset (a.k.a "public score" on the Kaggle page of the challenge [5]). One can see that BERT-based classifiers are the best, reaching an accuracy of 92.7%. Meanwhile, the difference between the smallest model (TinyBERT column 4 in Tab 2) and the largest model (BERT-base column 1 in Tab 2) is rather small (1.2%). This conclusion can be explained by the fact that, for the fine-tuning task, only a subset of the parameters is used (unfrozen parameters), therefore most the parameters of the model are kept as they are. We also show the accuracy of the GPT2 model using prompt learning (column 3 in Tab 2). We notice that this model shows a comparable performance to the BERT-based ones. Finally, the GCN NER classifier did not perform very well on the test dataset and its accuracy is only 77.9% (column 5 in Tab 2). The GCN classifier showed

[4] https://textmine.sciencesconf.org/resource/page/id/8

[5] https://www.kaggle.com/competitions/defi-textmine-2024/

promising results on the training dataset, as most of the nodes are known and the underlying graph creation is straightforward. However, we noticed that this GCN tends to over-fit on the train data and could not generalize well. We believe this is due to the way the graph is created (connecting nodes that appear in the same context), as with the provided TextMine dataset, each word has only limited context. This approach is not suitable for small datasets as it will require different contexts for each word.

4 Conclusion

We built an entity processing system based on a BERT model for the NER task, exploiting the existing pre-trained models. Our system was evaluated on the TextMine 2024 French dataset. The proposed model achieved an accuracy of 92.7% being only 5% points away from the best model. We also proposed a GCN model that did not perform well due to the small size of the data. These results open new perspectives for taking into account BERT embedding in a GCN model for NER.

References

Chiu, J. P. and E. Nichols (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics 4*, 357–370.

Collins, M. and Y. Singer (1999). Unsupervised models for named entity classification. In *1999 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora*.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kipf, T. N. and M. Welling (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Liu, X., H. Chen, and W. Xia (2022). Overview of named entity recognition. *Journal of Contemporary Educational Research 6*(5), 65–68.

Morwal, S., N. Jahan, and D. Chopra (2012). Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing (IJNLC) Vol 1*.

Patil, N., A. Patil, and B. Pawar (2020). Named entity recognition using conditional random fields. *Procedia Computer Science 167*, 1181–1188.

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). Deep contextualized word representations. In M. Walker, H. Ji, and A. Stent (Eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 2227–2237.

Rawsthorne, H., N. Abadie, A. Guille, P. Cuxac, V. Lemaire, and C. Lopez (2024). Reconnaissance d'entités géographiques dans un corpus des instructions nautiques (2024) défi textmine'24). *Conférence Extraction et Gestion des Connaissances 2024 (EGC'24)*.

Wang, Y., H. Tong, Z. Zhu, and Y. Li (2022). Nested named entity recognition: a survey. *ACM Transactions on Knowledge Discovery from Data (TKDD) 16*(6), 1–29.

Défi TextMine 2024 : "Reconnaissance d'entités géographiques dans un corpus des Instructions nautiques" - soumission équipe CRIT

Nicolas Gutehrlé*

*Université de Franche-Comté, CRIT
F-25000 Besançon, France
nicolas.gutehrle@univ-fcomte.fr,
https://nicolasgutehrle.github.io/

Résumé. Cet article présente notre participation au défi TextMine 2024. Nous proposons une méthode reposant sur un modèle CRF linéaire, ainsi qu'une méthode hybride combinant un modèle CRF linéaire et un modèle XGBoost. Nous entraînons les modèles sur des caractéristiques morphologiques et syntaxiques extraites de chaque mot. Notre modèle CRF linéaire obtient une micro F-mesure de 0.942 sur le jeu de test privé. Ces résultats montrent que des modèles simples tels que CRF linéaire entraînés sur des caractéristiques simples telles que la forme des mots sont efficaces pour la tâche de reconnaissance d'entités spatiales nommées et non nommées.

1 Introduction

Le défi TextMine 2024 consiste en la reconnaissance d'entités spatiales nommées et non nommées à plusieurs niveaux dans les Instructions nautiques, une série d'ouvrages publiée par le Shom (Service hydrographique et océanographique de la Marine). Le corpus mis à disposition est constitué d'extraits de 15 volumes des Instructions nautiques décrivant des environnements maritimes côtiers. La tâche de reconnaissance des entités nommées (REN) a suscité un vif intérêt ces dernières années, et a vu l'application de nombreuses approches, en particulier les approches par règles (Nadeau et Sekine, 2007; Sekine et Nobata, 2004; Kim et Woodland, 2000; Farmakiotou et al., 2000; Appelt et al., 1993), par apprentissage automatique (*machine-learning*) (Asahara et Matsumoto, 2003; Borthwick et al., 1998; Sekine, 1998; Bikel et al., 1998; McCallum et Li, 2003; Ritter et al., 2011; Rocktäschel et al., 2012), et plus récemment par apprentissage profond (*deep-learning*) (Yadav et Bethard, 2019; Li et al., 2020)

Dans cet article, nous présentons notre participation au défi TextMine 2024. Nous abordons la reconnaissance d'entités spatiales nommées et non nommées comme une tâche d'annotation de séquence. Nous entraînons pour cela trois modèles CRF linéaires (Lafferty et al., 2001), chacun entraîné sur des ensembles de caractéristiques différents. De plus, nous proposons un modèle hybride, combinant un modèle CRF linéaire pour la détection des mentions d'entités nommées, et un modèle XGBoost (Chen et Guestrin, 2016) pour la classification des mentions détectées. Nous entraînons les modèles sur des caractéristiques (*features*) morphologiques et

syntaxiques extraites de chaque mot dans le jeu de données d'entraînement mis à disposition. Les résultats du défi sont disponibles sur Kaggle [1], tandis que notre code et nos expérimentations sont disponibles sur GitHub [2]. Le reste de l'article est structuré comme suit : nous présentons les données dans la section 2, puis notre méthodologie dans la section 3. Nous présentons l'évaluation de notre méthodologie en section 4 avant de présenter notre conclusion en section 5.

2 Données

Un jeu de données d'entraînement et de test ont été mis à disposition des participants, au format CSV. Le jeu d'entraînement est divisé en deux colonnes, une pour les tokens et une pour les classes correspondantes. Le jeu de test ne contient que les tokens. Le jeu de test mis à disposition, dit jeu de test public, représente environ 60 % des données totales de test. Afin d'entraîner nos modèles et identifier les meilleures combinaisons d'hyper-paramètres possibles, nous avons divisé le jeu d'entraînement en un jeu d'entraînement et un jeu de développement. Le Tableau 1 présente la distribution des classes dans les jeux de données d'entraînement et de développement.

Classe	Entraînement	Développement
aucun	22 859	9 804
geogFeat	1 899	805
geogName name	1 491	626
geogFeat geogName	995	468
geogName	655	255
Total	27 899	11 958

TAB. 1 – *Distribution des classes dans les jeux de données d'entraînement et de développement.*

Plusieurs classes sont composées de multiples étiquettes : par exemple, la classe *geogFeat geogName* est composée des étiquettes *geogFeat* et *geogName*, qui sont toutes deux des classes à part entière dans le jeu de données. Le Tableau 2 présente la distribution des classes uniques dans les jeux d'entraînement et de développement.

Classe	Entraînement	Développement
name	1 491	626
geogFeat	2 894	1 273
geogName	3 141	1 349

TAB. 2 – *Distribution des étiquettes uniques dans les jeux de données d'entraînement et de développement.*

1. https://www.kaggle.com/competitions/defi-textmine-2024/submissions
2. https://github.com/nicolasgutehrle/DefiTextmineCRIT

3 Méthodologie et implémentation

Nous abordons ce défi comme une tâche d'annotation de séquence. Afin d'entraîner les modèles, nous extrayons des caractéristiques morphologiques et syntaxiques pour chaque token t dans le jeu de données. Nous extrayons ces mêmes caractéristiques des token qui précédent et suivent directement le token t. Ces caractéristiques sont préfixées par *prev_* et *next_* respectivement. Le Tableau 3 présente l'ensemble des caractéristiques extraites pour un token t. La caractéristique $shape$ représente les lettres majuscules par un "X", les valeurs numériques par un "d" et les ponctuations par un ".". Tous les autres caractères sont représentés par un "x". Par exemple, l'entité nommée "Nadji" aura la forme "Xxxxx". Nous employons le modèle *fr_core_news_lg* du framework *spaCy* [3] pour obtenir les parties du discours ainsi que les rôles syntaxiques des tokens. Nous employons également la liste de mots vides de ce modèle pour déterminer si un token est un mot vide.

	Caractéristique	Description
1	$Token$	Mot
2	$lower$	Mot en minuscule
3	$isdigit$	Vrai si le mot est une valeur numérique, sinon Faux
4	$isupper$	Vrai si le mot est écrit en lettres capitales, sinon Faux
5	$ispunct$	Vrai si le mot est une ponctuation, sinon Faux
6	$isstop$	Vrai si le mot est un mot vide, sinon Faux
7	len	Nombre de caractères composant le mot
8	$shape$	Forme du mot
9	pos	Partie du discours du mot
10	dep	Rôle syntaxique du mot

TAB. 3 – *Caractéristiques extraites pour chaque token.*

Nous entraînons trois modèles CRF linéaires, chacun avec un ensemble différent de caractéristiques : $baseModel$ entraîné avec les caractéristiques 1 à 8, $posModel$, entraîné avec les caractéristiques 1 à 9 et $posDepModel$, entraîné avec les caractéristiques 1 à 10. Notre objectif est de comparer l'apport des parties du discours et des dépendances syntaxiques pour la reconnaissance d'entités spatiales nommées et non nommées. Nous utilisons l'implémentation du modèle CRF linéaire proposée par le package *sklearn-crfsuite* [4]. Pour chaque modèle, nous recherchons la combinaison optimale de valeurs d'hyper-paramètres parmi les valeurs présentées dans le Tableau 4. Nous évaluons chaque modèle sur le jeu de développement. Les valeurs optimales de chaque hyper-paramètre pour chaque modèle sont présentées dans le Tableau 5. Enfin, nous entraînons chaque modèle CRF linéaire sur l'ensemble des données, c'est-à-dire sur les jeux d'entraînement et de développement combinés, avec les meilleurs valeurs d'hyper-paramètres identifiées.

Comme indiqué en section 2, la classe d'un token peut être une combinaison de plusieurs étiquettes. Afin d'exploiter la nature composite des classes, nous proposons un modèle hy-

3. https://spacy.io/
4. https://sklearn-crfsuite.readthedocs.io/en/latest/

c1	0.5, 0.1, 0.01, 0.001, 0.0001
c2	0.5, 0.1, 0.01, 0.001, 0.0001

TAB. 4 – *Valeurs testées pour les hyper-paramètres c1 et c2 du modèle CRF linéaire.*

Modèle	c1	c2
$baseModel$	0.1	0.001
$posModel$	0.1	0.01
$posDepModel$	0.1	0.01

TAB. 5 – *Valeurs optimales des hyper-paramètres pour les modèles $baseModel$, $posModel$ et $posDepModel$.*

bride, qui combine un modèle CRF linéaire pour la détection d'entités nommées et un modèle XGBoost pour la classification de ces entités.

Afin d'entraîner le modèle CRF linéaire de ce modèle hybride, nous modifions les jeux d'entraînement et de développement de telle sorte que toutes les classes autre que "aucun" sont remplacées par l'étiquette "NER". Nous proposons ainsi une version binaire de jeu de données. La distribution des classes dans les jeux d'entraînement et de développement binaires est présentée dans le Tableau 6.

Classe	Entraînement	Développement
aucun	22 859	9 804
NER	5 040	2 154
Total	27 899	11 958

TAB. 6 – *Distribution des classes binaires dans les jeux de données d'entraînement et de développement.*

Nous entraînons plusieurs modèles CRF linéaires afin de trouver la combinaison optimale de valeurs d'hyper-paramètres parmi les valeurs présentées dans le Tableau 7. La valeur optimale de chaque hyper-paramètre y est notée en gras. Chaque modèle est évalué sur le jeu de développement. Nous entraînons enfin le modèle CRF linéaire binaire final sur l'ensemble des données avec les meilleurs valeurs d'hyper-paramètres identifiées.

Pour entraîner le modèle XGBoost, nous modifions les jeux de données d'entraînement et de développement de telle sorte qu'un token est associé à plusieurs classes. Ainsi, nous entraînons ce modèle selon une tâche classification multi-étiquettes (*multi-label*), et non pas une tâche de classification multi-classes. Nous employons l'implémentation du modèle XGBoost du package *xgboost*. Nous entraînons plusieurs modèles XGBoost afin de trouver la combinaison optimale de valeurs d'hyper-paramètres parmi les valeurs présentées dans le Tableau 8. La valeur optimale de chaque hyper-paramètre y est notée en gras. Chaque modèle est évalué sur le jeu de développement. Nous entraînons enfin le modèle XGBoost final sur l'ensemble des données avec les meilleurs valeurs d'hyper-paramètres identifiées.

c1	0.5, 0.1, 0.01, 0.001, **0.0001**
c2	0.5, **0.1**, 0.01, 0.001, 0.0001

TAB. 7 – *Valeurs testées pour les hyper-paramètres c1 et c2 du modèle CRF linéaire. Les valeurs en gras sont les valeurs optimales identifiées.*

n_estimators	50, 100, **150**, 200, 350, 500
max_depth	1, 2, 3, 4, 5, 6, 7, 8, **9**, 10
learning_rate	0.0001, 0.001, 0.01, 0.1, **1.0**
subsample	.25, .5, .75, **1**

TAB. 8 – *Valeurs testées pour les hyper-paramètres n_estimators, max_depth, learning_rate et subsample du modèle XGBoost. Les valeurs en gras sont les valeurs optimales identifiées.*

4 Résultats et Discussion

Les modèles ont été évalués en terme de micro F-mesure sur un jeu de test public et un jeu de test privé. Le jeu de test public représente environ 60 % du jeu de test privé. Le Tableau 9 présente les performances des quatre modèles sur les jeux de test public et privé.

Modèle	Micro F-mesure (public)	Micro F-mesure (privé)
$baseModel$	**0.964**	0.942
$posModel$	0.959	**0.944**
$posDepModel$	0.959	0.943
$hybrid$	0.915	0.895

TAB. 9 – *Evaluation des quatre modèles selon la micro F-mesure sur les jeux de test public et privé.*

Sur le jeu de test public, le modèle $baseModel$ atteint le meilleur score, qui est de 0.964. Les modèles $posModel$ et $posDepModel$ obtiennent tous deux un score de 0.959. Enfin, le modèle $hybrid$ obtient le score le plus bas, qui est de 0.915. Sur le jeu de test privé, le modèle $posModel$ atteint le meilleur score, qui est de 0.944. Les modèles $baseModel$ et $posDepModel$ obtiennent respectivement un score de 0.942 et 0.943. Enfin, le modèle $hybrid$ obtient le score le plus bas, qui est de 0.895.

Les scores des modèles $baseModel$, $posModel$ et $posDepModel$ sont similaires. L'ajout des parties du discours et des rôles syntaxiques comme caractéristiques d'entraînement ne semblent pas affecter les performances du modèle. De simples caractéristiques d'entraînements, telles que celles que nous avons extraites, semblent donc adéquates pour identifier les entités spatiales nommées et non nommées. Cependant, les données mises à disposition rassemblent les tokens dans un seul grand document, sans les séparer en phrases. Cela peut affecter la qualité de l'analyse en partie du discours ainsi que l'analyse en dépendances syntaxiques. Le modèle $hybrid$ obtient les scores les plus bas sur les deux évaluations. Cette baisse des scores peut s'expliquer par l'incapacité du modèle XGBoost à prendre en compte le

contexte de l'élément à catégoriser. La conversion de la tâche de classification multiclasses en une tâche de classification multi-étiquettes peut également expliquer ces résultats inférieurs.

Le Tableau 10 présente les dix plus importantes caractéristiques identifiées par le modèle $baseModel$ contribuant positivement à l'identification de chaque classe. Chaque caractéristique est représentée par sa valeur et son poids. La caractéristique *lower*, c'est-à-dire la forme du mot en minuscule, est la plus fréquemment employée pour chaque classe. Le modèle se concentre sur le mot lui-même pour les classes contenant l'étiquette *geogFeat*, comme en témoigne l'emploi des caractéristiques *Token* et *lower*. Il en est de même pour les classes contenant l'étiquette *geogName* et *name*. Pour ces classes là, le modèle se concentre également sur les mots précédents et suivants, comme en témoigne l'emploi des caractéristiques *prev_Token, prev_lower, next_Token, next_lower*. Le modèle semble apprendre du vocabulaire maritime ("mouillage", "rade", "lagon", etc.) pour identifier les classe *geogFeat* et *geogFeat geogName*. De même, il semble apprendre des noms propres désignant des lieux ("Islande", "Djibouti", etc.) pour identifier les classes *geogFeat geogName* et *geogName name*.

aucun	geogName	geogName name	geogFeat	geogFeat geogName
isdigit, 8.512	*Token* :», 6.085	*Token* :nouveau, 7.609	*lower* :mouillages, 11.987	*lower* :île, 10.255
lower :entre, 7.967	*lower* :», 6.085	*lower* :nouveau, 7.609	*lower* :amers, 9.929	*lower* :pointe, 8.422
Token :petit, 7.822	*Token* :principal, 4.995	*prev_lower* :côte, 7.492	*lower* :rade, 9.815	*lower* :ras, 7.632
lower :ouest, 7.780	*lower* :principal, 4.995	*lower* :djibouti, 7.405	*lower* :lagon, 9.778	*lower* :cap, 6.950
Token :grande, 6.443	*prev_lower* :capitainerie, 4.939	*lower* :blancpignon, 7.383	*lower* :alignement, 9.282	*lower* :îles, 6.755
lower :petite, 6.408	*next_Token* :médian, 4.922	*lower* :grande, 7.100	*Token* :appontement 9.074	*lower* :aéroport, 6.554
lower :jusqu', 5.881	*next_lower* :médian, 4.922	*lower* :espagnole, 6.454	*lower* :pontons, 9.057	*Token* :Island, 6.535
shape :X, 5.880	*shape* :xxxxxxxxx, 4.187	*Token* :ancien, 6.296	*Token* :baie, 8.515	*lower* :island, 6.535
lower :devant, 5.782	*prev_lower* :phare, 4.123	*lower* :ancien, 6.296	*lower* :échelles, 8.266	*Token* :Pass, 6.496
Token :Est, 5.746	*next_Token* :sur, 3.932	*shape* :Xxxxx.Xxxxxxx, 6.258	*lower* :îles, 8.249	*prev_Token* :ancienne, 6.251

TAB. 10 – *Dix plus importantes caractéristiques identifiées par le modèle $baseModel$ contribuant positivement à l'identification de chaque classe. Chaque caractéristique est représentée par sa valeur et son poids.*

5 Conclusion

Dans cet article, nous avons présenté notre participation au Défi TextMine 2024 : "Reconnaissance d'entités géographiques dans un corpus des Instructions nautiques". Nous avons proposé deux méthodes pour la tâche de reconnaissance d'entités spatiales nommées et non nommées. Notre première méthode repose sur un modèle CRF linéaire, tandis que notre seconde méthode est hybride, et combine un modèle CRF linéaire avec un modèle XGBoost. Notre modèle CRF linéaire, entraîné sur des caractéristiques morphologiques simples telles que la forme des mots, obtient une micro F-mesure de 0.964 sur le jeu de test public, et une micro F-mesure de 0.942 sur le jeu de test privé. Ces résultats montrent que des modèles simples tels que CRF linéaire entraînés sur des caractéristiques morphologiques également simples sont efficaces pour la tâche de reconnaissance d'entités spatiales nommées et non nommées. Dans des travaux futurs, nous comptons expérimenter avec d'autres caractéristiques d'entraînement d'ordre morphologique, syntaxique ou sémantique. Nous avons également l'intention d'approfondir notre étude sur la nature composite des entités nommées et non-nommées en continuant nos expérimentations sur la classification multi-étiquettes.

Références

Appelt, D. E., J. R. Hobbs, J. Bear, D. Israel, et M. Tyson (1993). Fastus : A finite-state processor for information extraction from real-world text. In *IJCAI*, Volume 93, pp. 1172–1178.

Asahara, M. et Y. Matsumoto (2003). Japanese named entity extraction with redundant morphological analysis. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pp. 8–15.

Bikel, D. M., S. Miller, R. Schwartz, et R. Weischedel (1998). Nymble : a high-performance learning name-finder. *arXiv preprint cmp-lg/9803003*.

Borthwick, A., J. Sterling, E. Agichtein, et R. Grishman (1998). Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7), Fairfax, Virginia, April 29-May 1, 1998*.

Chen, T. et C. Guestrin (2016). Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Farmakiotou, D., V. Karkaletsis, J. Koutsias, G. Sigletos, C. D. Spyropoulos, et P. Stamatopoulos (2000). Rule-based named entity recognition for greek financial texts. In *Proceedings of the Workshop on Computational lexicography and Multimedia Dictionaries (COMLEX 2000)*, pp. 75–78.

Kim, J.-H. et P. C. Woodland (2000). A rule-based named entity recognition system for speech input. In *Sixth International Conference on Spoken Language Processing*.

Lafferty, J., A. McCallum, et F. C. Pereira (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data.

Li, J., A. Sun, J. Han, et C. Li (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering 34*(1), 50–70.

McCallum, A. et W. Li (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons.

Nadeau, D. et S. Sekine (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*(1), 3–26.

Ritter, A., S. Clark, O. Etzioni, et al. (2011). Named entity recognition in tweets : an experimental study. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pp. 1524–1534.

Rocktäschel, T., M. Weidlich, et U. Leser (2012). Chemspot : a hybrid system for chemical named entity recognition. *Bioinformatics 28*(12), 1633–1640.

Sekine, S. (1998). Nyu : Description of the japanese ne system used for met-2. *http ://www. muc. saic. com/*.

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

Yadav, V. et S. Bethard (2019). A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv :1910.11470*.

Summary

This article presents our participation to the TextMine 2024 challenge. We propose a method based on a linear CRF model, as well as a hybrid method combining a linear CRF model and an XGBoost model. We train the models on morphological and syntactic features extracted from each word. Our linear CRF model obtains a micro F-measure of 0.942 on the private test set. These results show that simple models such as linear CRF trained on simple features such as word shape are effective for the Named and Unnamed Spatial Entity Recognition task.

OctopusMind @ Défi TextMine'24 Reconnaissance d'entités géographiques dans un corpus d'instructions nautiques

Oussama Ahmia*
Danrun Cao* **
Nicolas Béchet**
Pierre-François Marteau**

*OctopusMind
Nantes, France
o.ahmia@octopusmind.info, d.cao@octopusmind.info
https://www.octopusmind.info/

**Univ. Bretagne Sud, CNRS, IRISA
Vannes, France
danrun.cao@univ-ubs.fr ; nicolas.bechet@univ-ubs.fr ; pierre-rancois.marteau@univ-ubs.fr
https://www.irisa.fr/en/equipes/expression

Résumé. Le défi Textmine'24 concerne la reconnaissance d'entités nommées et non nommées au sein des instructions nautiques, en utilisant un jeu de données extrait d'une série d'ouvrages annotés publiés par le Shom. Cet article présente les résultats obtenus en appliquant des méthodes de l'état de l'art dans le domaine de la reconnaissance des entités nommées en considérant le problème de reconnaissance comme une tâche multi-classe plutôt que multi-label. Cette approche est mise en production chez OctopusMind, un acteur français spécialiste en intelligence économique.

1 Introduction

Le défi Textmine'24 (Rawsthorne et al., 2024) cible la "Reconnaissance d'entités géographiques dans un corpus d'instructions nautiques". Proposé par l'IGN (Institut Géographique National) et le Shom (Service hydrographique et océanographique de la Marine), ce défi concerne plus précisément l'identification d'entités spatiales nommées et non nommées dans les instructions nautiques. Le corpus annoté à partir d'ouvrages publiés par le Shom, offre des détails sur les environnements maritimes côtiers mondiaux, fournissant des informations cruciales pour la navigation sécurisée et l'accès aux ports. Ce corpus est constitué d'extraits annotés issus de 15 volumes d'instructions nautiques, totalisant 66 030 tokens et 18 537 labels.

Notre participation à ce défi nous offre l'opportunité d'évaluer des méthodes actuellement utilisées en production chez OctopusMind pour des tâches similaires. Dans cet article, nous appliquons certaines approches de l'état de l'art dans le domaine de l'extraction des entités nommées, notamment les modèles à base de CRF (Conditional Random Fields (Lafferty et al.,

2001)) et de Transformers (Vaswani et al., 2017), en changeant de perspective sur la problématique en considérant ainsi le problème de reconnaissance comme une tâche multi-classe plutôt que multi-label.

2 Jeu de données

Le jeu de données fourni aux participants comprend un total de 66 030 tokens et 18 537 labels. Il est préalablement tokenisé et annoté au niveau des tokens, puis divisé en deux parties distinctes :

train, le jeu d'entraînement annoté contenant 39 857 tokens et **test**, le jeu d'évaluation (public) non annoté contenant 26 173 tokens, ce qui représente 41% du jeu de **test** total (public + privé).

Les étiquettes (labels) utilisées sont les suivantes :

name : pour les noms propres purs, par exemple : "Port de **Kpémé**"
geogFeat : pour les noms communs qui identifient une caractéristique géographique, par exemple : le **récif** qui porte l'îlot Pagode, **geogName** : pour le nom associé à une caractéristique géographique, par exemple : "De l' Est du **phare de Nadji** jusqu'à ..."

Chaque token peut recevoir entre 0 et 2 labels, la problématique peut donc être considérée comme multi-label, ce qui fait 5 combinaisons possibles.

3 Méthode

Nous avons choisi de traiter le problème de reconnaissance comme une tâche multi-classe plutôt que multi-label. Cela signifie que si un token possède deux labels, nous considérons cette combinaison comme une classe distincte, ce qui nous donne un total de 5 classes : **geogFeat geogName**, **geogFeat**, **geogName**, **geogName name**, **aucun**. Cette approche a déjà fait ses preuves sur des tâches similaires (Ahmia et al., 2017) et permet de simplifier la tâche.

Le jeu de données est présenté sous la forme d'une seule séquence englobant l'ensemble des tokens. Par conséquent, nous avons organisé les tokens par phrases en utilisant le critère de fin de phrase défini par le token ".". Nous avons ainsi obtenu 1702 phrases. Le tableau 1, présente des statistiques sur le nombre de tokens par phrase.

Moyenne	23.4
Maximum	302
Minimum	2
Median	17.5

TAB. 1 – *Statistiques sur le nombre de tokens par phrase.*

Les algorithmes que nous avons utilisés sont présentés ci-après.

3.1 CRF

Les champs aléatoires conditionnels ou **CRF** constituent une famille de modèles discriminants probabilistes et peuvent être considérés comme des graphes non dirigés, les noeuds de ce graphe sont séparés en deux ensembles X et Y, les observations et les sorties (les étiquettes). Les arcs entres les noeuds représentent la probabilité conditionnelle de transition $P(Y|X)$. Les **CRF** sont utilisés principalement pour la modélisation de séquences, en particulier dans le traitement du langage pour la détection d'entités nommées.

Nous avons utilisé différents types de caractéristiques pour représenter chaque token : **text** la forme du token, **lemma** le lemme, **lower** le token en minuscule, **shape** la forme du mot (majuscules, ponctuation, chiffres) exemple '4.5.3'='d.d.d', **is_alpha** si le token est formé que de lettres, **is_digit** si le token est un nombre, **is_lower** si le token est en minuscule, **is_upper** si le token est en majuscule, **is_title** si le premier caractère du token est en majuscule, **is_punct** si le token est une ponctuation et **pos** qui est la catégorie morphosyntaxique du token.

Nous avons aussi utilisé des caractéristiques à longue portée qui modélisent le contexte d'occurrence des tokens (les variables voisines d'un token).

L'optimisation des poids du CRF est réalisée avec la méthode **Limited-memory Broyden-Fletcher-Goldfarb-Shanno** (L-BFGS, Zhu et al. (1997)), le **CRF** est entraîné avec l'outil **CRFsuite** et l'extraction de caractéristiques est produite avec **Spacy**.

3.2 CamemBERT

CamemBERT (Martin et al., 2019) est un modèle de langage basé sur l'architecture BERT (Bidirectional Encoder Representations from Transformers) pré-entraîné sur des corpus français et sur plusieurs tâches différentes (classification, séquence à séquence, ...). Nous adaptons ce modèle pour une tâche d'extraction d'entités nommées en rajoutant une couche linéaire dense en sortie du réseau CamemBERT qui permettra de prédire les classes de sorties pour chaque Token. Un apprentissage final (*fine tuning*) est ensuite pratiqué en exploitant les données **TextMine'24**. Afin d'éviter le sur-apprentissage du modèle, on effectue une stratification (Sechidis et al., 2011) des donnée afin de créer des ensembles d'entraînement et de validation homogènes en terme de représentation des classes (en ne considérant qu'une seule occurrence d'une classe dans une séquence). Pour l'entraînement du réseau de neurones on utilise l'outil **Transformers** [1].

On notera aussi que le jeu de données fourni est préalablement tokenisé en mots, tandis que CamemBERT utilise une tokenisation en sous-mots (sub-word tokenization). Par conséquent, nous avons mis en place une correspondance entre les tokens d'origine et les tokens en sous-mots, ce qui nous permet de faire le lien entre les étiquettes fournies par CamemBERT et les tokens d'origine.

4 Résultats

L'évaluation du défi **TextMine'24** est obtenue via une mesure de F-score sur les données de test publiques. Le tableau 2, présente les scores des différentes versions de CRF testées : "**CRF base**" où seul le texte du token est utilisé comme caractéristique, "**CRF context 3 text**"

1. https ://huggingface.co/docs/transformers/index

	F-score
CRF base	90.1%
CRF context 3 text	93.9%
CRF context 5 all features	95.4%
CRF context 3 all features	95.7%
CamemBERT 16 ep	97.4%
CamemBERT stratified 30 ep	97.6%
CamemBERT alldata 60 ep	98.0%
CamemBERT alldata 30 ep	98.1%

TAB. 2 – *Résultats des modèles en termes de F-score obtenus sur les données de test publiques*

un CRF utilisant des caractéristiques à longue portée (contexte local de taille 3 du token), "**CRF context 5 all features**" et "**CRF context 3 all features**" qui utilise toutes les caractéristiques mentionnées dans la Section 3.1 avec respectivement des contextes de tailles 5 et 3.

On remarque que l'utilisation de caractéristiques à longue portée améliore le score du CRF jusqu'à atteindre une certaine limite, sachant que l'augmentation du contexte augmente la dimensionnalité du problème pouvant ainsi dégrader les performances.

Nous avons aussi testé différentes approches d'entraînement pour le réseau CamemBERT en faisant varier le nombre d'époques ainsi que l'entraînement sur la totalité ou une partie des données. Le gain en termes de score est cependant marginal avec une légère amélioration de la performance pour le modèle entraîné sur la totalité des données. Au-delà de 30 époques, nous n'observons plus d'amélioration de performance. On notera aussi que le modèle CamemBERT obtient de meilleurs scores comparativement aux CRF (**+0.03**), néanmoins, le temps nécessaire pour générer un modèle CamemBERT est bien plus conséquent que pour générer un modèle CRF (30 minutes vs quelques secondes pour des tests effectués sur une machine munie d'une GeForce RTX 2080 Ti et d'un AMD Ryzen 9 3950X).

Le code permettant de générer le modèle obtenant le meilleur score est accessible via le lien suivant : `https://github.com/oussamaahmia/textmine24`

5 Conclusion

Cet article présente l'approche que nous avons développée dans le cadre de notre participation au défi TextMine 2024, dont l'objectif cible la reconnaissance des entités nommées et non nommées au sein d'un corpus d'instructions nautiques du Shom. L'originalité de notre approche consiste à aborder cette problématique comme un problème multi-classe plutôt qu'un problème multi-label. Nous avons comparé les performances de quelques architectures CRF à celles d'un réseau CamemBERT affiné spécifiquement pour cette tâche. Le modèle CamemBERT a obtenu la première place du classement du défi TextMine privé/public [2].

En perspective, compte tenu de l'écart faible de performance entre CRF et modèles camenBERT, le principe du rasoir d'Occam nous incite à approfondir les expérimentations sur les architectures CRF en testant d'autres combinaisons de caractéristiques. De plus, une analyse

2. `https://www.kaggle.com/competitions/defi-textmine-2024/leaderboard`

approfondie des poids attribués aux différentes caractéristiques pourrait apporter des explications supplémentaires sur les résultats obtenus par ces architectures CRF.

Références

Ahmia, O., N. Béchet, et P.-F. Marteau (2017). Apprentissage de structures séquentielles pour l'extraction d'entités et de relations dans des textes d'appels d'offres. In *EGC*, pp. 351–356.

Lafferty, J., A. McCallum, et F. C. Pereira (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data.

Martin, L., B. Muller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. de La Clergerie, D. Seddah, et B. Sagot (2019). Camembert : a tasty french language model. *arXiv preprint arXiv :1911.03894*.

Rawsthorne, H., A. G. N. Abadie, V. L. P. Cuxac, et C. Lopez (2024). Reconnaissance d'entités géographiques dans un corpus des instructions nautiques. actes de l'atelier textmine'24. *Conférence Extraction et Gestion des Connaissances 2024 (EGC'24)*.

Sechidis, K., G. Tsoumakas, et I. Vlahavas (2011). On the stratification of multi-label data. In *Machine Learning and Knowledge Discovery in Databases : European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III 22*, pp. 145–158. Springer.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, et I. Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems 30*.

Zhu, C., R. H. Byrd, P. Lu, et J. Nocedal (1997). Algorithm 778 : L-bfgs-b : Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS) 23*(4), 550–560.

Summary

The Textmine'24 challenge concerns the recognition of named and unnamed entities within nautical instructions, using a dataset extracted from a series of annotated documents published by Shom. This article presents the results obtained by applying state-of-the-art methods in named entity recognition by considering the recognition problem as a multi-class rather than multi-label task. This approach is currently deployed at OctopusMind, a French player specializing in business intelligence.

Détection d'entités nommées géographiques par réseau de neurones récurrents

Lucas Anki*, Léo Gaillard*, Justine Revol*

*Inist-CNRS (UAR 76), 2 rue Jean Zay, 54500 Vandoeuvre-lès-Nancy, France
[prénom].[nom]@inist.fr

Résumé. Cet article présente notre méthodologie pour répondre au défi TextMine 2024 (voir Guille (2023)). L'objectif est la **Reconnaissance d'entités géographiques dans un corpus des instructions nautiques**. Nous utilisons pour cela un réseau de neurones récurrents que nous construisons et entraînons en utilisant le framework **Flair** (Akbik et al. (2019)).

1 Introduction

Le groupe de travail TextMine a pour but de réunir des chercheurs et chercheuses sur la thématique large de la fouille de texte. Le défi TextMine naît d'une problématique spécifique à la fouille de texte (ici la détection d'entités nommées géographiques), la solution est apportée sous forme d'un défi, en mettant à disposition des jeux de données inédits à la communauté scientifique. C'est dans ce cadre que nous participons à l'édition 2024 : le problème est posé par l'Institut national de l'information géographique et forestière (IGN) et le Service hydrographique et océanographique de la Marine (Shom). L'objectif est la **reconnaissance d'entités géographiques dans un corpus des instructions nautiques**. Pour répondre à ce défi, les participants et participantes ont à leur disposition un corpus constitué d'extraits de 15 volumes des instructions nautiques annotés selon trois labels distincts : **name**, pour les noms propres purs ; **geogFeat**, pour les noms communs qui identifient une caractéristique géographique ainsi que **geogName**, pour les noms associés à une caractéristique géographique. Le jeu de donnée est constitué de 66030 tokens et de 18537 labels et chaque token peut avoir jusqu'à deux labels. L'ensemble de ces données est disponibles sur la plateforme Kaggle à ce lien [1]. Pour évaluer les scores, le modèle doit prédire les entités nommées sur le fichier *test.csv*, qui est non annoté, et les participants et participantes sont départagés en fonction de la micro f-mesure résultante.

2 Etat de l'art

La Reconnaissance d'entités nommées, une composante importante de la recherche d'information, a connu une évolution significative au fil des années. Depuis la sortie de *BERT* en 2019, les transformers fine-tunés pour de la détection d'entités nommées ont des performances plus que compétitives.

1. https://www.kaggle.com/competitions/defi-textmine-2024/data

Dans le cadre du défi TextMine, nous avons choisi d'utiliser un réseau de neurones récurrents (RNN) grâce à la bibliothèque **Flair**, ceci pour plusieurs raisons : encore aujourd'hui, les RNN sont toujours beaucoup utilisés pour faire de la détection d'entités nommées comme par exemple dans l'article de Smirnova et Mayr (2023) pour de la reconnaissance d'entités nommées sur des remerciements scientifiques ou encore dans l'article de Suignard et al. (2023) en réponse au défi TextMine 2023 (pour ne citer qu'eux). Dans ces deux cas, le framework **Flair** est utilisé et donne de bons résultats. De plus, en comparaison des transformers les RNN sont moins lourd pour une performance du même ordre sur cette tâche. On peut également noter que le framework **Flair** est encore maintenu : la dernière contribution apportée date de décembre 2023 contre décembre 2019 pour le projet **BERT-NER** [2].

En comparant avec d'autres framework fréquemment utilisés pour faire de la détection d'entités nommées, cet article de Vychegzhanin et Kotelnikov (2019) montre que **Flair** obtient souvent des meilleurs f-mesure.

L'ensemble de ces raisons ainsi que nos expériences passées nous ont montré que **Flair** est un framework adapté pour répondre à ce défi, pour ses performances ainsi que pour sa fiabilité. Le framework **Flair** est entièrement disponible sur GitHub, à ce lien [3].

3 Les données d'entraînement

3.1 Métriques sur les labels

Le jeu de données d'entrainement compte 39857 tokens, et chaque token peut avoir entre 0 et 2 étiquettes. Le tableau 1 la répartition des différents labels possibles. Nous pouvons observer que les classes sont déséquilibrées : il y a 41 fois plus de tokens dont le label est **aucun** que de token dont le label est **geogName**. Nous avons choisi de ne pas considérer la classe **aucun** à l'entraînement et d'attribuer ce label aux tokens n'en portant pas d'autres. En plus de régler le problème de déséquilibre entre les classes, ce procédé est naturel au vu de la signification logique du label.

Labels	nombres d'occurence	fréquence d'occurence
aucun	32663	0.82
geogFeat	2704	0.07
geogName name	2117	0.05
geogFeat geogName	1463	0.04
geogName	910	0.02
geogFeat name	0	0
name	0	0

TAB. 1 – *Répartition des différentes étiquettes possibles dans le jeu d'entrainement train_2.csv.*

Après analyse des données, seules quatre étiquettes se retrouvent dans les données d'entraînement sur les six possibles au départ. Le label **geogFeat name** n'est pas possible à trouver,

2. https://github.com/kamalkraj/BERT-NER
3. https://github.com/flairNLP/flair

étant donné que **geogFeat** fait référence à des noms communs et **name** à des noms propres. Il est possible que le label **name** se retrouve seul dans les données d'évaluation, mais nous avons choisi de ne pas le détecter pour éviter le bruit et se concentrer uniquement sur des entités nommées géographiques : seul les **name geogName** seront détectés.

3.2 Formatage des données

Pour une tâche de détection d'entités nommées et sous version 0.10 (dernière version documentée en date de *juin 2023*), l'utilisation du framework **Flair** nécessite une structure particulière (voir Akbik et al. (2019))

Les règles de formatage sont les suivantes :

1. Trois fichiers textes sont constitués : *train.txt test.txt* et *dev.txt*.
2. Une ligne est constitué d'un unique token, et de son label, séparés par un espace ou une tabulation.
3. Une ligne vide sépare chaque phrase.

Le fichier *train.txt* contient les données d'entraînement. Après chaque époque, nous calculons la f-mesure sur un ensemble de données *dev.txt* et enfin à la fin de l'entraînement nous calculons la f-mesure ou toute autre métrique de qualité sur le corpus *test.txt*. Nous avons choisi une répartition assez usuelle des données à notre disposition : 0.7 / 0.15 / 0.15. La répartition entre les différents fichiers se fait aléatoirement : chaque phrase a une probabilité de 0.7 d'être dans le fichier *train.txt*, de 0.15 d'être dans le fichier *dev.txt* et le reste dans les données constituent le fichier *test.txt*.

4 Entraînement du modèle et utilisation de Flair

En adéquation avec la feuille de route du **CNRS** pour la science ouverte, nous mettons à disposition le code utilisé pour l'entraînement à ce lien [4].

4.1 L'embedding

Nous avons choisi d'utiliser un embedding contextuel de dimension 300, suggéré dans cet article Akbik et al. (2018) par **Flair** pour la majorité des cas d'usages : on utilise la fonction *StackedEmbeddings* de **Flair** pour combiner [5] un embedding **FastText** (Bojanowski et al. (2016)), un embedding contextuel prenant comme contexte le token juste après et un embedding contextuel prenant comme contexte le token juste avant. Chacun de ces embedding a été entraîné sur des données issues du Wikipédia français.

Dans notre optique de modèle léger, nous avons privilégié cet embedding contextuel plutôt qu'un embedding type transformers. En effet les embedding contextuels suggérés par **Flair** ne sont pas significativement différents pour des cas d'usages similaires d'après l'article de Brunner et al. (2020).

4. `https://github.com/Luc-Ank/textmine-2023`
5. Ces trois Embedding sont nommés dans le code respectivement *FlairEmbeddings('fr')*, *FlairEmbeddings('fr-forward')* et *FlairEmbeddings('fr-backward')*

4.2 Structure du modèle

Le réseau de neurone récurrent utilisé par notre modèle comporte trois couche :

— Une couche de reprojection de l'embedding. Dans notre cas la dimension n'est pas réduite et permet un simple ajustement des poids par un modèle linéaire lors de l'apprentissage.

— Deux couches type LSTM, chacune ayant un *word dropout* de 0.05.

4.3 Paramètres de l'entrainement

Pour information, les paramètres utilisés lors de l'entraînement du modèle sont écrits dans le tableau 2. Les paramètres non précisés sont les paramètres par défaut de **Flair** en version **0.10**. Avec le framework **Flair**, si une époque n'est pas une amélioration significative (sur un jeu distinct de l'entrainement), les poids du réseau sont inchangés. Nous souhaitions un grand *learning rate*, un grand nombre d'*époque* et une grande *patience* pour permettre le plus d'occurences possibles et éviter au maximum toute convergence de la loss vers un minimum local.

Paramètre	Valeur	Explicitation
learning_rate	0.2	qu'on abrègera *lr*
patience	10	réduction du *lr* après 10 mauvaises époques
mini_batch_size	12	petit batch size avec des LSTM
min_learning_rate	0.001	valeur par défaut
anneal_factor	0.8	facteur de réduction du *lr*

TAB. 2 – *Exemple de données d'entrainements pour utiliser **Flair**.*

5 Résultats

Afin d'optimiser les premiers résultats, nous avons utilisé la possibilité de faire de nombreuses soumissions à **Kaggle** pour augmenter artificiellement les données d'entraînement : nous n'avons alors plus besoin de fichier *test.txt*, la f-mesure obtenue lors de la soumission faisant office de résultat. Ainsi, les données présentes dans ce fichier peuvent être injectées dans les données d'entraînement : 0.85 pour le fichier *train.txt* et 0.15 pour le fichier *dev.txt*

La figure 1 montre l'évolution [6] de la loss calculée après chaque époque, calculée sur les données issues du fichier *dev.txt*. L'allure de la courbe ne montre ni un sur-apprentissage ni un sous-apprentissage.

La meilleure micro f-mesure obtenue sur le fichier *test.tsv* est de **0.97925**. Nous détaillons les métriques pour chaque label dans le tableau 3. Cependant, ces métriques sont calculées sur le jeu de donnée *dev.txt* étant donné que nous avons pris le parti d'utiliser le jeu de donnée d'évaluation du concours TextMine 2024, *test.tsv* comme unique évaluation de la qualité du modèle..

6. Pour modéliser des données discrètes par une courbe, nous avons utilisé une interpolation ce qui explique que la loss semble remonter légèrement lors de certaines époques, ce qui n'est pas possible avec la méthode employée.

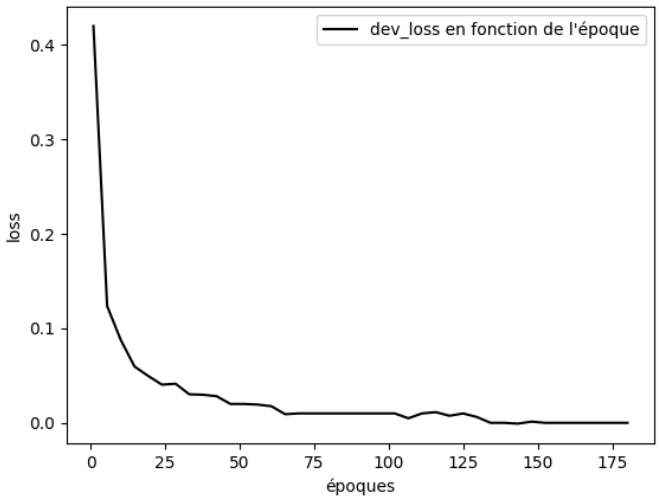

FIG. 1 – *Evolution de la loss lors de l'entrainement du modèle pour le fichier 'dev.txt' en fonction des époques.*

Labels	precision	recall	f1-score
geogFeat	0.8986	0.9538	0.9254
geogName name	0.9737	1.0000	0.9867
geogFeat geogName	1.0000	0.8333	0.9091
geogName	1.0000	0.8667	0.9286

TAB. 3 – *Métriques obtenues sur le fichier "dev.txt" pour chacun des labels à la fin de l'entrainement*

6 Conclusion et perspective

En conclusion, les résultats de notre modèle sont plus que satisfaisant et ils peuvent être aiséments reproduits ou modifiés en utilisant le code à disposition sur **Github** et les données sur **Kaggle**, même sur un ordinateur basique.

Nous n'avons pas beaucoup exploré la question de la dimension de l'embedding : en partant du principe qu'il y a moins de vocabulaire dans des corpus d'instructions nautique, ou tout autre corpus spécialisé, que dans Wikipédia il pourrait être intéressant de réduire la dimension sur laquelle l'embedding est projeté. Il peut être également intéressant de comparer ces résultats avec ceux que nous aurions pu obtenir en utilisant un LLM pour faire l'embedding, voir en utilisant la structure totale d'un modèle type transformers pour réaliser ce défi.

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 *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59.

Akbik, A., D. Blythe, et R. Vollgraf (2018). Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Bojanowski, P., E. Grave, A. Joulin, et T. Mikolov (2016). Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*.

Brunner, A., N. D. T. Tu, L. Weimer, et F. Jannidis (2020). To bert or not to bert - comparing contextual embeddings in a deep learning architecture for the automatic recognition of four types of speech, thought and writing representation. In *SwissText/KONVENS*.

Guille, A. (2023). Défi textmine 2024.

Smirnova, N. et P. Mayr (2023). Embedding models for supervised automatic extraction and classification of named entities in scientific acknowledgements. *Scientometrics*, 1–25.

Suignard, P., L. Hassani, et M. Bothua (EasyChair, 2023). Participation d'edf rd au défi textmine 2023 : Reconnaissance d'entités d'intérêts dans les signatures d'e-mails. EasyChair Preprint no. 10098.

Vychegzhanin, S. et E. Kotelnikov (2019). Comparison of named entity recognition tools applied to news articles. pp. 72–77.

Summary

This article presents the methodology we used for the TextMine 2024 challenge. The objective is the **Recognition of geographical entities in a corpus of nautical instructions**. We built and trained a RNN with the **Flair** framework.

TETIS @ Challenge TextMine 2024 : "Reconnaissance d'entités géographiques dans un corpus des Instructions nautiques"

Rémy Decoupes[*,**] Roberto Interdonato[*,***]
Rodrique Kafando[*,**] Mathieu Roche[*,***] Mehtab Alam Syed [*,***] Maguelonne Teisseire[*,**] Sarah Valentin[*,***]

[*]TETIS, Univ. Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, Montpellier 34090, France
[**]INRAE, F-34398 Montpellier, France
[***]CIRAD, F-34398 Montpellier, France

1 Introduction

Notre équipe, issue de l'unité TETIS (Territoires, environnement, télédétection et information spatiale), s'intéresse à différents tâches du domaine du Traitement Automatique du Langage Naturel (TALN) en se concentrant sur les problématiques associées à l'information spatiale. La reconnaissance des entités spatiales dans les données textuelles représente une étape importante dans la majorité des chaînes de traitements. Aussi, les données relatives à l'information spatiale nécessitent une attention particulière. Par exemple, nos travaux proposent (i) des méthodes d'extraction d'information spatiale à partir de données non standards (Zenasni et al., 2018), (ii) des méthodes et outils de *geocoding* et désambiguïsation d'entités spatiales (Syed et al., 2023; Kafando et al., 2023), (iii) des algorithmes d'augmentation de données fondés sur l'information géographique pour l'entraînement de modèles de type Transformers (Decoupes et al., 2023).

La participation de notre équipe TETIS à ce challenge est dans la continuité de ces travaux. Nous résumons, dans la section suivante, notre approche et mettons à disposition nos codes [1] et modèle [2].

2 Approche

L'approche a consisté à comparer différents modèles pré-entraînés ajustés sur les données du challenge. Afin de commenter les différents résultats, nous présentons la distribution du jeu de données.

1. Dépôt logiciel : `https://github.com/tetis-nlp/tetis-challenge_textmine_2024`
2. HuggingFace hub : `https://huggingface.co/rdecoupes/tetis-textmine-2024-camembert-large-based`

2.1 Données d'apprentissage

Le jeu de données d'apprentissage est extrêmement déséquilibré (Figure 1) : les exemples associés à des labels géographiques (geogFeat, geogName, name GeogName et GeogFeat GeogName) ne représentent que 18% (n=7194/39857) des données labellisées.

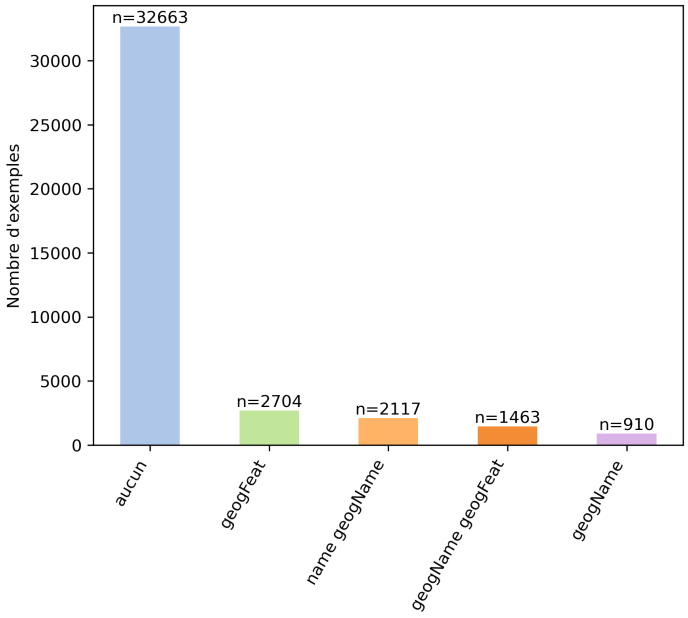

FIG. 1 – *Distribution du nombre d'exemples par classe dans le jeu de données d'entraînement*

La classe *geogFeat* se distingue des autres par l'absence de noms propres. En effet, les mots ou groupes de mots annotés avec ce label désignent une caractéristique géographique sans nom propre comme *phare* ou *port*.

2.2 Méthodes et résultats

Dans un premier temps, nous avons ré-entraîné un modèle en langue française proposé par la librairie spaCy (pipeline *fr core news lg* [3]). Le micro F1-score sur le jeu de données de validation (20 % du jeu de données mis à disposition, en utilisant l'échelle de la phrase) et le jeu de données calculé par la plateforme Kaggle est de 0.950 et de 0.927, respectivement. Ceci a constitué notre base de référence.

Notre deuxième approche a visé à ré-ajuster des modèles de type Transformers pré-entraînés (*fine-tuning* en anglais). Nous avons comparé trois modèles (Figure 2) : deux modèles en langue française (CamemBERT-base et CamemBERT-large) et un modèle multi-langues (XLM-RoBERTa). Sans optimisation des hyperparamètres, CamemBERT-large offre, sur le jeu de

3. https://github.com/explosion/spacy-models/releases/tag/fr_core_news_lg-3.7.0

données calculé par la plateforme Kaggle, les meilleurs résultats. Nous avons ensuite fait varier certains hyperparamètres tels que les tailles de batch d'entraînement ($batch\ size$), le taux d'apprentissage ($learning\ rate$) et le nombre d'epochs ($epochs$) sur plusieurs entraînements différents. La meilleure combinaison d'hyperparamètres sur le jeu de données de validation est : $batch\ size = 16$, $learning\ rate = 1e-05$, $epochs = 10$, alors que pour les données calculées par la plateforme Kaggle, la meilleure combinaison est : $batch\ size = 8$, $learning\ rate = 2e-05$, $epochs = 10$. Nous avons, cependant, constaté une grande variabilité entre deux entraînements ayant des hyperparamètres identiques (Figure 2).

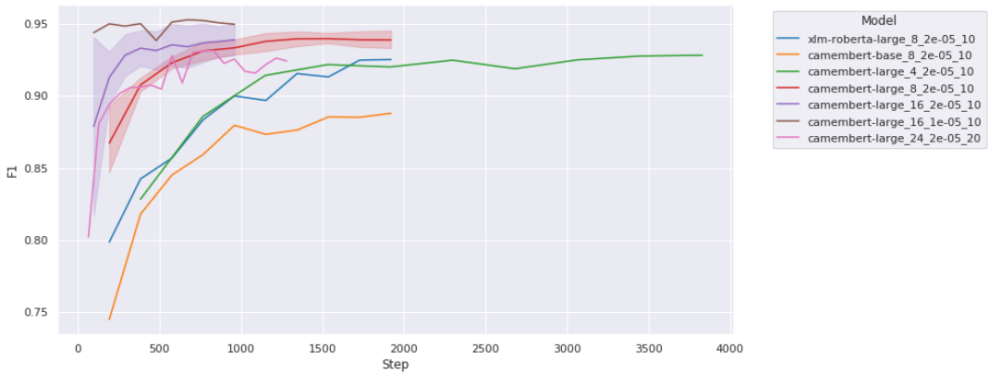

FIG. 2 – *Comparaison des F1-score pendant l'entraînement des 3 modèles avec différents hyperparamètres. Les traces sous forme d'enveloppe correspondent aux écarts types des différents entraînements ayant des hyperparamètres identiques. La règle de nommage est : le nom du modèle pré-entraîné, le* batch size*, le* learning rate *et le nombre* epochs

Sur les données de validation, i.e. 20% du jeu de données mis à disposition, les performances du modèle issu du modèle spaCy sont très inégales : la classe *aucun* obtient le meilleur F1-score (0.978), tandis que le F1-score des autres classes ne dépassent pas 0.877 (Tableau 1). Ces performances modérées s'expliquent par une confusion des instances géographiques avec la classe *aucun*, ce qui diminue leur rappel (Figure 3). La classe *geogFeat*, qui obtient le score le plus bas, souffre également d'un manque de précision.

Les modèles de type transformers, bien que moins performant pour la classe *aucun*, obtiennent de meilleurs résultats pour les classes géographiques. Cependant ils rencontrent également des difficultés avec la classe *geogFeat* comme illustré par le Tableau 1. Cette classe est complexe à distinguer de *aucun* (Figure 4). Les modèles détectent plus efficacement les classes contenant un nom propre i.e. *geogName*, *name geogName* et *geogFeat_geogName*.

TAB. 1 – *Comparaison des F1-scores par classe en fonction du modèle pré-entrainé*

Modèle	**moyenne pondérée *micro***	**aucun**	**geogFeat**	**name geogName**	**geogFeat geogName**	**geogName**
spaCy	0.950	0.978	0.779	0.877	0.823	0.804
xlm-roberta-large	0.926	0.927	0.914	0.932	0.931	0.914
camembert-large	0.950	0.947	0.915	0.969	0.980	0.971

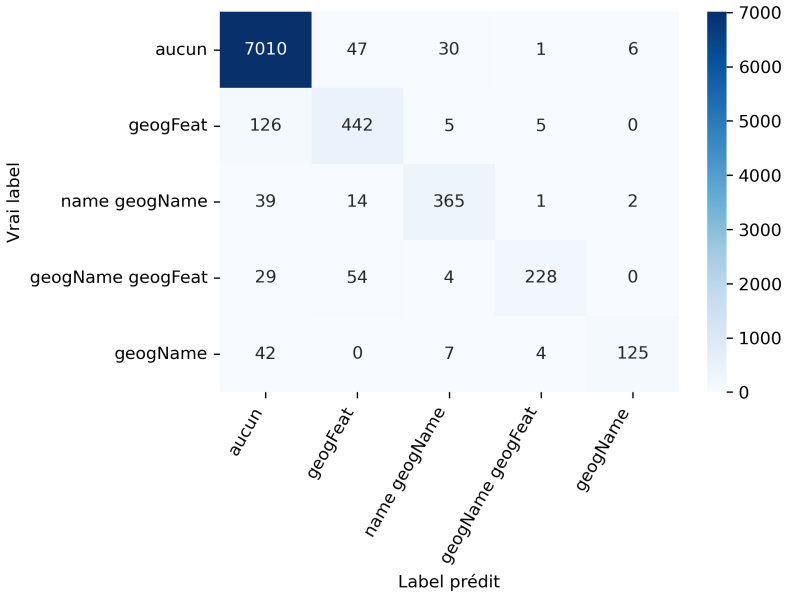

FIG. 3 – *Matrice de confusion obtenue à partir du modèle spaCy*

Références

Decoupes, R., M. Roche, et M. Teisseire (2023). GeoNLPlify : A spatial data augmentation enhancing text classification for crisis monitoring. *Intelligent Data Analysis*, 1–25.

Kafando, R., R. Decoupes, M. Roche, et M. Teisseire (2023). SNEToolkit : Spatial named entities disambiguation toolkit. *SoftwareX 23*, 101480.

Syed, M. A., E. Arsevska, M. Roche, et M. Teisseire (2023). GeospatRE : extraction and geocoding of spatial relation entities in textual documents. *Cartography and Geographic Information Science*, 1–16.

Zenasni, S., E. Kergosien, M. Roche, et M. Teisseire (2018). Spatial information extraction from short messages. *Expert Systems with Applications 95*, 351–367.

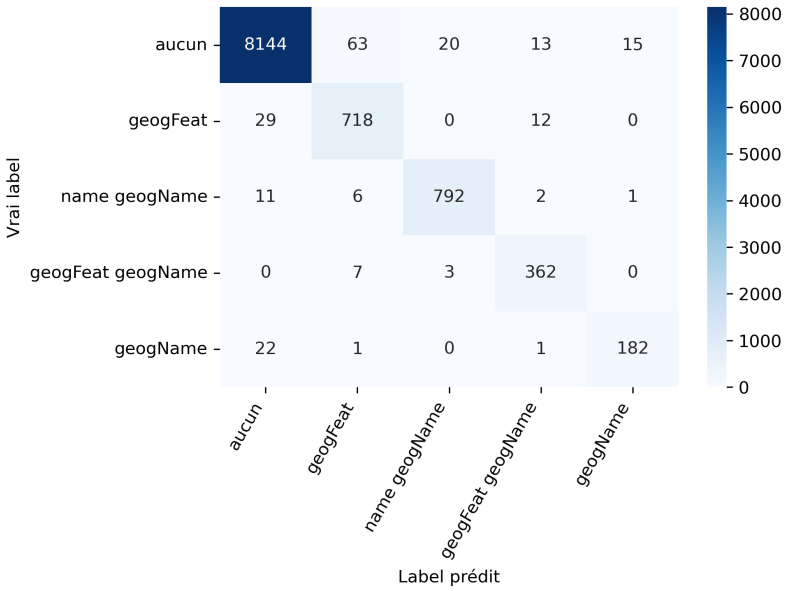

FIG. 4 – *Matrice de confusion obtenue à partir du modèle camembert-large*

Index

S

Z