Méthodes hybrides pour la reconnaissance d'inférences textuelles en présence d'opérateurs argumentatifs

Mémoire de fin d'étude

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours CMI IASD

Auteur
Gatien HADDAD

Superviseurs
Pascal PONCELET[1]
Christian RETORÉ[1]
Grégoire WINTERSTEIN[2]

Lieu de stage
[1]LIRMM UM5506 - CNRS, Université de Montpellier, France
[2]UQÀM Pavillon Hubert-Aquin, A-3405 400, rue Sainte-Catherine Est, Montréal, QC, Canada

soutenu publiquement le 20 Juin 2024

Résumé

Les avancées en Machine Learning ont révolutionné le traitement automatique des langues, mais les modèles actuels peinent dans les tâches impliquant le raisonnement logique, comme la reconnaissance d'inférences textuelles. Notre travail vise à explorer une méthode hybride combinant l'apprentissage automatique et la linguistique pour traiter les inférences textuelles, en se concentrant sur les opérateurs argumentatifs tels que "presque" et "à peine". Nous analysons ces opérateurs en construisant un corpus annoté pour entraîner un système de prédiction des relations argumentatives. Notre approche vise à améliorer les modèles basés uniquement sur le Machine Learning en intégrant des connaissances linguistiques pour une meilleure compréhension des inférences textuelles.

Abstract

Advancements in machine learning have revolutionized natural language processing, but current models struggle with tasks involving logical reasoning, such as recognizing textual inferences. Our work aims to explore a hybrid method combining machine learning and linguistics to address textual inferences, focusing on French discourse operators such as "presque" (almost) and "à peine" (barely). We analyze these operators using an annotated corpus to train a system for predicting argumentative relations. Our approach aims to enhance models solely based on machine learning by integrating linguistic knowledge for a better understanding of textual inferences.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à l'aboutissement de ce projet et qui m'ont accompagné tout au long de sa réalisation.

Je voudrais dans un premier temps remercier mes maîtres de stage M. Christian RETORÉ, M. Pascal PONCELET et M. Grégoire WINTERSTEIN, pour m'avoir apporté leur aide précieuse sur leur connaissances respectives et m'avoir accompagné tout au long de ces six mois de stage.

Je remercie également tous les membres de l'équipe TEXTE et ADVANSE du LIRMM ainsi que tous les doctorants, chercheurs et stagiaires avec qui j'ai pu échanger, pour la bienveillance dont ils ont fait preuve en m'intégrant au sein du LIRMM et pour leur soutien constant tout au long de ce projet.

Je tiens par la suite à témoigner toute ma reconnaissance auprès des membres de l'équipe SLIC, et plus généralement au département informatique de l'UQÀM, pour leur accueil chaleureux au sein du département, leur bienveillance, ainsi que pour les échanges enrichissants en linguistique et en TALN qui ont grandement contribué à l'avancement de ce travail.

Table des matières

Table des matières **iv**

1 Introduction **1**
1.1 Contexte et motivation 1
1.2 Objectifs de recherche 1

2 Présentation du sujet **2**
2.1 Qu'est-ce que le Traitement Automatique du Langage Naturel ? 2
2.1.1 Définition et intérêt du TALN 2
2.1.2 Méthodes utilisées 2
2.2 Importance des inférences textuelles dans le TALN 4
2.2.1 Définition et intérêt 4
2.2.2 Approches traditionnelles des inférences textuelles 5
2.2.3 Limitations des approches traditionnelles 6
2.2.4 Introduction aux méthodes de *Machine Learning* 6
2.2.5 Approche hybride 7
2.3 Application des méthodes hybrides aux opérateurs argumentatifs 7

3 Approches existantes pour traiter les opérateurs argumentatifs **9**
3.1 Recherches antérieures sur les opérateurs argumentatifs 9
3.2 Lacunes et défis à relever dans le traitement des opérateurs argumentatifs 10
3.3 Exploration des approches hybrides 11

4 Création du corpus **14**
4.1 Extraction des phrases 14
4.1.1 Première possibilité envisagée : CommitmentBank 14
4.1.2 Utilisation de Wikipedia 14
4.2 Filtrage des données 15
4.3 Annotation du corpus 16
4.4 Description des opérateurs argumentatifs ciblés 16

5 Mise en place du système de prédiction **18**
5.1 Choix du modèle et des méthodes de prédiction 18
5.1.1 Sélection du modèle linguistique 18
5.1.2 Méthode de prédiction 18
5.2 Évaluation des performances 19
5.3 Améliorations du modèle 20
5.4 Axes d'étude parallèles 23

6 Résultats et discussion **26**

6.1 Analyse et interprétation des résultats obtenus 26
6.2 Limitations de l'étude et regard critique 28

7 Bilan **30**
7.0.1 Réalisation des objectifs initiaux 30
7.0.2 Connaissances acquises 30
7.0.3 Apport personnel et projet professionnel 30

Bibliographie **32**

1. Introduction

1.1 Contexte et motivation

Dans un monde où l'utilisation des technologies du langage naturel est devenue omniprésente, la compréhension précise et profonde du langage reste cependant un défi majeur. Le traitement automatique du langage naturel (TALN) joue un rôle crucial dans ce domaine, en permettant aux ordinateurs de comprendre du texte en le transformant en représentations utilisables par une machine, et également de pouvoir générer du langage de manière automatique. Au coeur de cette problématique, se trouve la capacité à reconnaître des inférences textuelles (RTE en Anglais, pour "*Recognizing Textual Entailment*"), c'est-à-dire pouvoir déduire des informations à partir des éléments d'un texte.

De plus, nous avons récemment pu constater que l'avènement des grands modèles de langue (LLM en anglais) a révolutionné le domaine du TALN en permettant des avancées significatives dans des tâches telles que la traduction automatique ou la génération de texte. Cependant, malgré ces progrès, les LLM sont encore confrontés à des difficultés lorsqu'il s'agit de comprendre les inférences textuelles et les nuances du langage. Les opérateurs argumentatifs, tels que "presque" et "à peine", sont un exemple de défis particuliers en raison de leur capacité à modifier subtilement le sens d'une phrase selon le contexte.

1.2 Objectifs de recherche

Ce stage propose d'explorer des méthodes hybrides pour aborder les inférences textuelles en présence d'opérateurs argumentatifs. Nous commencerons par définir la question étudiée, puis nous présenterons une étude bibliographique qui guidera notre résolution du problème, où nous présenterons nos méthodologies proposées et les objectifs visés dans le cadre de ce stage de recherche.

2. Présentation du sujet

2.1 Qu'est-ce que le Traitement Automatique du Langage Naturel ?

2.1.1 Définition et intérêt du TALN

Le langage naturel désigne le mode de communication servant à s'exprimer à l'oral ou à l'écrit, en utilisant des règles propres à chaque langue. Ces règles peuvent être syntaxiques, lexicales et sémantiques, et témoignent de la diversité des expressions linguistiques, en allant des conversations entre amis aux discours formels, et permettent d'exprimer une large gamme de nuances.
Permettre aux machines de comprendre le langage naturel est essentiel pour permettre à ces dernières d'interagir de manière plus naturelle avec les utilisateurs. Par exemple, ces systèmes sont utilisés pour développer des *chatbots* pour répondre aux questions des utilisateurs, des outils de traduction automatique, ou encore des outils d'analyse de texte pour extraire des informations pertinentes à partir de grandes quantités de données.
Le traitement automatique du langage naturel (TALN) a alors été introduit pour contribuer à la compréhension du langage naturel par les machines. Ce domaine de recherche se situe à cheval entre l'informatique et la linguistique, et vise à développer des algorithmes capables de traiter automatiquement et de manière intelligente le langage naturel. En somme, son objectif principal est de permettre aux ordinateurs de "comprendre", d'interpréter et d'engendrer du langage de la même manière que nous sommes capables de le faire.

2.1.2 Méthodes utilisées

Pour atteindre cet objectif, les méthodes de TALN englobent un large éventail de techniques qui visent à permettre aux machines de *comprendre*, c'est-à-dire transformer le langage en une représentation interprétable par la machine, de manière automatique. Parmi ces méthodes, il y a dans un premier temps les méthodes symboliques, où les règles linguistiques sont explicitement décrites au programme. Cette approche implique souvent des tâches telles que l'encodage des règles grammaticales et la manipulation de structures syntaxiques et sémantiques. En utilisant des méthodes symboliques, il est possible de construire des systèmes TALN qui fonctionnent selon des règles préétablies, imitant ainsi le processus de réflexion humain lors de la compréhension du langage naturel.
Parmi les méthodes symboliques les plus utilisées, nous pouvons retrouver les suivantes.

2.1.2.1 Analyse syntaxique

L'analyse syntaxique consiste à décomposer une phrase en plusieurs composants syntaxiques, afin de déterminer sa structure grammaticale. Cette méthode utilise des grammaires dites formelles pour définir les règles de formation des phrases. Par exemple, pour la phrase "Un chien regarde un homme dans le parc", une analyse syntaxique pourrait produire la structure suivante :

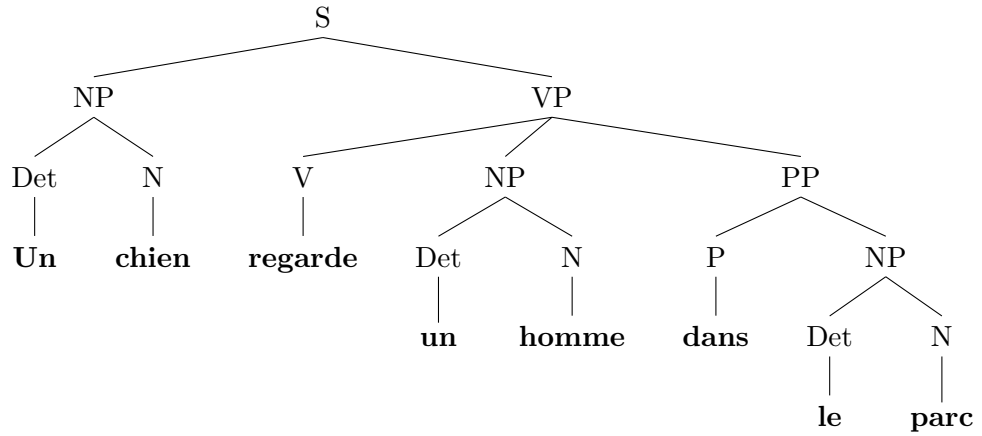


FIGURE 2.1 : Une analyse syntaxique possible d'une phrase

Il peut cependant y avoir des ambiguïtés dans les phrases. Dans l'exemple précédent, le chien peut être en dehors du parc et regarder un homme qui est dedans, ou alors l'inverse, ce qui fait qu'il peut exister plusieurs analyses syntaxiques pour la même phrase. En voici une seconde où le chien se trouve dans le parc.

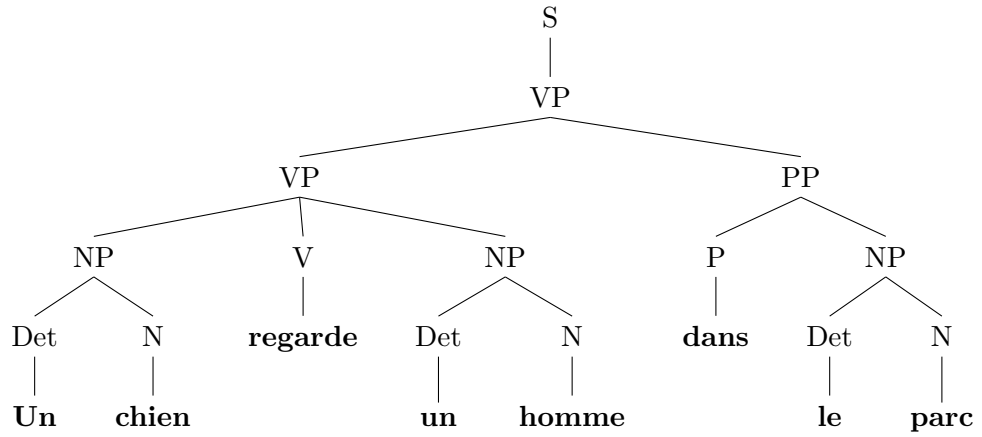


FIGURE 2.2 : Analyse syntaxique où le chien est dans le parc.

Cette structure montre comment la phrase est décomposée en ses différentes parties grammaticales, facilitant ainsi son interprétation et son traitement par des systèmes de TALN.

2.1.2.2 Analyse sémantique

L'analyse sémantique, quant à elle, vise à attribuer un sens aux phrases, en utilisant des représentations formelles de la signification pour comprendre ce que les phrases veulent dire. Cela peut inclure la logique des prédicats, qui est une méthode où l'on convertit les phrases en formules logiques. Par exemple, si l'on prend la phrase "Tous les chiens aboient", on pourrait la convertir en une formule logique équivalente comme $\forall x.Chien(x) \rightarrow Aboie(x)$, soit "Pour tout x, si x est un chien, alors x aboie".
Une autre méthode utilisée est celle des réseaux sémantiques, qui représentent les relations entre concepts. Par exemple, un réseau sémantique pourrait montrer que le mot "chien" est lié à celui de "animal", et que "aboie" est lié au concept de "faire du bruit".

Ces techniques permettent donc aux systèmes de comprendre comment les mots et les phrases peuvent être liés entre eux par leur sens. Cependant, comprendre le sens des mots ne suffit pas toujours pour saisir l'ensemble des informations d'un texte. En effet, il est souvent nécessaire d'aller au-delà des mots et d'inférer des informations qui ne sont pas dites explicitement pour saisir toutes les subtilités d'un texte. C'est ici qu'intervient la notion d'inférences textuelles.

2.2 Importance des inférences textuelles dans le TALN

2.2.1 Définition et intérêt

Les inférences textuelles désignent la capacité à déduire des informations à partir d'un texte. Cette capacité va donc au-delà de la simple reconnaissance des mots, et requiert la compréhension du sens profond et des relations logiques entre les énoncés. Les inférences textuelles permettent de saisir les nuances et les ambiguïtés qui pourraient être présentes dans un texte, ce qui est essentiel pour une compréhension complète du langage. En effet, nous utilisons souvent des inférences textuelles pour tirer des conclusions, interpréter des intentions et anticiper des réponses lors de discussions quotidiennes. Par exemple, si l'on nous dit "Alice regarda le ciel, elle comprit qu'il allait pleuvoir", nous pouvons naturellement en déduire que Alice a vu des nuages.

Le processus est le même pour une machine à qui nous demanderions "Qui a peint La Joconde ?". Pour répondre correctement "Léonard de Vinci", notre modèle doit être capable de comprendre que cette réponse découle logiquement d'une information présente dans une autre phrase qui a été utilisée lors de l'entraînement. Par exemple, si nous lui donnons comme information préalable la phrase "Au Louvre est exposée La Joconde de Léonard de Vinci", notre machine doit être en mesure d'inférer la phrase "Léonard de Vinci a peint La Joconde" à partir du texte initial. En poussant la notion de déduction, nous pourrions dire que notre modèle doit être capable de comprendre que l'artiste associé à l'exposition de La Joconde au Louvre est également celui qui l'a peinte. C'est donc ce processus de déduction et de compréhension du sens profond des énoncés qui caractérise l'inférence textuelle.

Formellement, l'inférence textuelle est une relation dirigée d'un texte (ou prémisse) T vers une hypothèse H. Une hypothèse H est inférée par un texte T, si un humain

qui lirait T déduirait que H est (très probablement) vraie. Comme nous pouvons le remarquer, cette définition semble un peu abstraite, car elle se base sur "ce qu'une personne déduirait". Cette définition est problématique car elle ne repose pas uniquement sur des critères strictement logiques, mais implique également des éléments subjectifs d'interprétation humaine.

Cependant, cela est contrebalancé par plusieurs facteurs. D'une part, la création de jeux de données de qualité supérieure, c'est-à-dire des données d'évaluation qui sont manuellement annotées selon des normes établies, et qui contiennent généralement un large éventail d'exemples couvrant différents cas d'utilisation du domaine étudié [6]. De plus, grâce à la disposition jeux de données de nature variées. Les jeux de données pour l'inférence textuelle tels que GLUE [1] adoptent souvent une définition probabiliste de l'inférence, en évaluant la probabilité qu'une phrase en implique une autre ou son contraire, tandis que des jeux de données comme FraCaS [2] et SICK [3] reposent sur des exemples basés plus strictement sur des règles logiques et des phénomènes linguistiques.

FraCaS, par exemple, fournit des exemples qui nécessitent une compréhension approfondie des structures logiques, comme par exemple « Tous les hommes sont mortels. Socrate est un homme. Donc, Socrate est mortel. », qui nécessite de comprendre la logique de la syllogisme. En revanche, SICK se concentre sur des phénomènes linguistiques comme la synonymie ou l'antonymie, avec des phrases comme "Le chat est sur le tapis" et "Le félin est sur le tapis", où la synonymie entre "chat" et "félin" doit être reconnue.

Ces différents jeux de données permettent de tester les modèles de TALN sous différents angles, et permettent ainsi de pouvoir réaliser une évaluation complète de leurs capacités à faire des inférences textuelles. Les utilisations des inférences textuelles se concentrent sur les réponses à des questions à partir de textes, comme par exemple au travers de la détection de sentiment dans des commentaires (inférer s'ils sont positifs, négatifs, etc) ou encore dans les moteurs de recherche pour extraire les informations pertinentes.

2.2.2 Approches traditionnelles des inférences textuelles

Les méthodes basées sur des règles de logique formelle sont depuis longtemps utilisées pour aborder les inférences textuelles en TALN [4]. Il s'agit d'utiliser des règles de logique et de linguistique, qui sont utilisées pour déduire de nouvelles informations à partir des informations déjà présentes dans le texte. Par exemple, pour la règle du modus ponens, si l'on a la phrase "A implique B" et la phrase "A", alors on peut conclure que la phrase "B" est vraie. Ces méthodes ont été largement utilisées pour modéliser des tâches telles que de l'analyse sémantique des énoncés ou de la déduction formelle dans des domaines spécifiques, tels que des systèmes-experts pour de l'aide à la décision médicale.
Ces méthodes se basent sur des algorithmes de traitement symbolique du langage, qui constituent donc une approche traditionnelle pour aborder les inférences textuelles. En combinant un grand nombre de règles, il est donc possible de construire des systèmes de

[1] GLUE : A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding [20]
[2] Framework for Computational Semantics [5]
[3] Sentences Involving Compositional Knowledge [15]
[4] Jean-Claude Anscombre and Oswald Ducrot - *L'argumentation dans la langue* [1]

TALN capables de réaliser une variété de tâches liées à l'analyse et à la compréhension du langage naturel.

2.2.3 Limitations des approches traditionnelles

Bien que les approches symboliques offrent une base solide pour le TALN en raison de leur fondement dans la logique formelle, elles présentent également des limites. Ces méthodes reposent souvent sur des règles préétablies, ce qui les rend généralement fiables selon la précision de ces règles. Cependant, il est difficile de généraliser ces règles à toutes les situations possibles, car cela nécessiterait de prendre en compte un nombre trop grand de cas particuliers. Par conséquent, les approches symboliques peuvent être limitées dans leur capacité à traiter efficacement la grande variabilité du langage naturel, en particulier dans des domaines ou en constante évolution, comme par exemple des données médicales, ou des discussions sur les réseaux sociaux. De plus, elles nécessitent souvent de grandes bases de connaissances ou des ontologies créées manuellement, ce qui représente un travail considérable en termes de temps, car ces bases doivent être continuellement mises à jour pour rester pertinentes. Cette difficulté à généraliser les règles symboliques constitue l'un des principaux défis auxquels sont confrontées les approches traditionnelles du traitement du langage naturel, et souligne la nécessité de développer des méthodes plus flexibles et adaptatives pour surmonter ces limitations.

2.2.4 Introduction aux méthodes de *Machine Learning*

Une des approches existantes consiste alors à se tourner vers le *Machine Learning* (ML). Ce dernier suscite un intérêt croissant dans de nombreux domaines, ce qui nous intéresse particulièrement pour traiter la complexité du langage naturel.

Le *Machine Learning* est une branche de l'intelligence artificielle qui se focalise sur la capacité des machines à apprendre, seules, à partir de données, sans nécessité de programmation spécifique pour chaque tâche différente. Comme nous l'avons évoqué plus tôt, le ML trouve des applications dans une multitude de domaines, allant de la traduction automatique à la génération de texte, en passant par l'analyse de sentiments sur les réseaux sociaux.

Pour réaliser ce genre de tâche pour du TALN, il est primordial d'avoir un corpus de données qui servira de base d'entraînement pour le modèle d'inférences textuelles. Ce corpus doit contenir une variété d'exemples de texte, accompagnés d'annotations sur les relations sémantiques et logiques entre les phrases. Ces annotations peuvent inclure des informations sur les relations d'inférences, de contradiction ou de neutralité entre les énoncés.

Une fois le corpus annoté créé, différents modèles d'inférences textuelles peuvent être entraînés sur ce corpus, puis doivent être évalués sur un ensemble de données de test pour évaluer leurs performances et leur capacité à généraliser sur de nouveaux exemples. Ce processus d'entraînement et d'évaluation permet d'améliorer progressivement les performances des modèles d'inférences textuelles et de les adapter à des tâches spécifiques.

Ainsi, en exploitant de grandes quantités de données et en utilisant des algorithmes qui visent à imiter le cerveau humain tels que les réseaux de neurones, le ML permet de construire des systèmes de TALN capables d'apprendre à partir d'exemples et de s'adapter à des situations variées.

Cependant, il est assez contraignant de devoir s'entraîner sur autant de données, d'une part, car cela requiert d'avoir de grandes puissances de calcul pour pouvoir entraîner puis faire tourner ces modèles, et d'autre part, car selon la tâche nécessaire, un corpus de cette envergure n'est pas toujours disponible et il faut alors pouvoir extraire ces données en grande quantité, ce qui nécessite également beaucoup de temps.

2.2.5 Approche hybride

C'est dans un but de tirer parti des avantages respectifs de chaque méthode pour améliorer les performances des systèmes de TALN qu'est apparue l'approche par des méthodes hybrides [5]. En combinant les méthodes traditionnelles basées sur des règles symboliques avec les techniques d'apprentissage autonome du ML, il est possible de construire des systèmes hybrides capables de bénéficier à la fois de la précision des règles symboliques, ainsi que la capacité d'adaptation et l'efficacité du ML.

Cette combinaison des méthodes traditionnelles et du ML peut se faire de plusieurs manières différentes. Par exemple, les règles symboliques peuvent être utilisées pour guider l'apprentissage des modèles de ML en fournissant des informations structurées sur la langue pour implémenter le processus d'inférence. Un exemple serait d'ajouter au corpus des informations syntaxiques, par exemple des annotations grammaticales pour chaque mot, pour aider le modèle de ML à mieux comprendre la structure des phrases et généraliser sur les patrons d'inférences.
D'un autre côté, un modèle de ML peut être utilisé pour identifier les entités nommées dans un texte, et ces entités peuvent ensuite être traitées par un système à base de règles pour effectuer des inférences plus complexes ou des analyses spécifiques, comme la détection de relations entre entités.

2.3 Application des méthodes hybrides aux opérateurs argumentatifs

C'est en ce sens que l'objectif de ce travail est de mettre en place une méthode hybride permettant de traiter efficacement les inférences textuelles. De plus, le but est de se concentrer sur des opérateurs argumentatifs comme "presque" et "à peine".

Les opérateurs argumentatifs sont des mots ou des expressions qui modifient la force ou la direction de l'argument présenté dans une phrase. Ils jouent un rôle crucial dans la structuration du discours en influençant la manière dont une assertion est interprétée par le lecteur ou l'auditeur.

[5] On parle également d'"IA-neuro-symbolique" [10]

Par exemple, pour l'opérateur "presque", il signifie d'un point de vue logique qu'une action n'est **pas** terminée, en engageant le locuteur sur la négation de l'adverbe visé. Cependant, il existe des cas où sa présence vient argumenter le contraire. Considérons l'exemple suivant :
(a) J'ai *presque* fini ma bière, commande-moi en une autre.
(b) J'ai fini ma bière, commande-moi en une autre.
La phrase (b) qui est obtenue en retirant l'opérateur est appelée "préjacent". D'un point de vue logique, l'adverbe *presque* en (a) exprime le fait que la bière n'est **pas** terminée. Cependant, les exemples (a) et (b) apparaissent tous les deux naturels car exprimant la même information. Dans ce cas là, on dit que les deux phrases sont en co-orientation argumentative.

La notion de *classe* argumentative repose sur le caractère distributionnel des énoncés. Cela signifie que pour déterminer si deux énoncés appartiennent à la même classe argumentative, on observe les contextes dans lesquels ils apparaissent et comment ils influencent les discours environnants. Par exemple, si deux énoncés peuvent être utilisés de manière interchangeable dans les mêmes contextes tout en conservant la même orientation argumentative, ils sont considérés comme co-orientés argumentativement.
Il est également possible d'observer l'inverse avec l'exemple suivant :
(a) Il a *presque* été élu, il se représentera l'année prochaine.
(b) # Il a été élu, il se représentera l'année prochaine.
Ici, l'adverbe *presque* en (a) exprime que le candidat à échoué à l'élection. La phrase et son préjacent apparaissent donc en opposition argumentative, car l'adverbe *presque* suggère un échec imminent alors que son préjacent implique un succès complet, rendant ainsi l'énoncé en (b) incohérent dans le même contexte.

Ainsi, le but de ce stage est de développer un système hybride de reconnaissance d'inférences textuelles qui combine les avantages du ML et des méthodes symboliques pour obtenir des performances optimales dans la classification de relation argumentative entre deux phrases, impliquant des opérateurs argumentatifs tels que "presque" et "à peine".

3. Approches existantes pour traiter les opérateurs argumentatifs

3.1 Recherches antérieures sur les opérateurs argumentatifs

Une partie de la recherche dans le domaine de la linguistique se focalise sur la manière dont les informations peuvent être inférées de manière (plus ou moins) fiable à partir d'un texte, ainsi que sur le raisonnement basé sur ces informations, que ce soit en utilisant des règles linguistiques, ou bien des inférences statistiques basées sur les caractéristiques des textes représentés [1]. Pour évaluer les performances de tels modèles de langues, il existe notamment des bechmarks qui fournissent une multitude de tâches de compréhension du langage naturel, tels que GLUE/SuperGLUE [20, 19], ou même FLUE avec un corpus en Français [13]. Ces benchmarks offrent plusieurs outils pour évaluer les performances des modèles sur un ensemble diversifié de tâches en TALN, et permettre de mesurer leur robustesse et leur capacité d'adaptation.

Une grande partie de la recherche s'effectue également autour du *Machine Learning*. En effet, et comme présenté dans l'introduction, les grands modèles de langues ont connu une croissance importante ces dernières années, et ont donc été utilisés pour des tâches de reconnaissances d'inférences textuelles [2].

Les LLM, tels que GPT, reposent principalement sur une architecture appelée transformeur, introduite en 2017 [3]. Cette architecture repose sur des mécanismes d'attention, qui permettent au modèle de se concentrer sur différentes parties du texte de manière dynamique en attribuant des poids différents aux mots en fonction de leur importance dans le contexte global. Les transformeurs utilisent plusieurs couches d'attention pour traiter les informations de manière parallèle, ce qui les rend plus efficaces et capables de gérer de grandes quantités de données textuelles. Grâce à cette architecture, les modèles de langue peuvent capturer des relations complexes entre les mots et les phrases, ce qui est crucial pour des tâches d'inférences textuelles.

En ce qui concerne les opérateurs argumentatifs comme "presque" et "à peine", plusieurs recherches ont montré que les modèles de langue peuvent bien traiter certains de leurs aspects sémantiques. Par exemple, une étude a examiné comment les humains et les modèles de langue traitent le contenu informationnel et l'orientation discursive des énoncés impliquant ces adverbes [4]. Les résultats ont montré que les humains sont sensibles

[1] Ido Dagan et al. - *Recognizing Textual entailment : models and applications* [6]
[2] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Zanzotto - *Recognizing textual entailment* [6]
[3] Vaswani et al. - *Attention is All You Need* [18]
[4] Winterstein et al. - *Informational content vs. discourse orientation* [22]

à la fois au contenu informationnel et à l'orientation discursive, alors que les modèles de langue ont des performances moins transparentes. Les performances des modèles de langue, comme CamemBERT [5], sur les tâches d'inférence ont montré que les modèles capturent bien les patterns d'inférence logique pour "presque", mais ont des résultats mitigés pour "à peine", en partie à cause de la complexité contextuelle de cet adverbe.

3.2 Lacunes et défis à relever dans le traitement des opérateurs argumentatifs

Bien que ces modèles offrent une méthode pour traiter les inférences textuelles, les LLM ont cependant une propension naturelle à générer des hallucinations [6]. Les hallucinations sont des informations qui apparaissent lorsque les modèles génèrent des informations inexactes, voire incorrectes, souvent en raison de biais dans les données d'entraînement ou dans les algorithmes d'apprentissage. Ces hallucinations peuvent donc compromettre la fiabilité et la pertinence des résultats produits pour des systèmes d'inférences textuelles.

De plus, à cause du fait que ce sont des modèles statistiques, ces derniers ont également du mal à prendre en compte les nuances sémantiques d'une variété d'opérateurs présents dans le langage naturel. La raison est que les modèles statistiques se basent sur des schémas où les mots apparaissent ensemble dans les données d'entraînement, plutôt que sur des règles linguistiques explicites. Cela signifie qu'ils peuvent manquer de généralisation dans des contextes nouveaux ou moins fréquents. Un des exemples est notamment le processus d'inversion [7]. Lorsqu'un modèle est entraîné avec une phrase de type "A est B", il ne sera pas forcément en mesure de répondre correctement à la question "B est A".

FIGURE 3.1 : Illustration d'une erreur de LLM sur le processus d'inversion

Dans le cas des inférences textuelles, il a été montré que de nombreux jeux de données contenaient des artefacts d'annotations, c'est-à-dire des biais introduits par des humains lors du processus d'annotation des données, ce qui peut fausser les résultats des modèles. En effet, les modèles entraînés sur ces corpus arrivaient à prédire la relation d'inférence entre le texte et l'hypothèse sans même lire le texte, notamment grâce à des contradictions dans les hypothèses. Par exemple, l'hypothèse "Thomas n'était pas présent au partiel" était considérée par le modèle comme une contradiction en raison de la négation

[5] CamemBERT : a Tasty French Language Model [16]
[6] Yue Zhang et al. - *A survey on hallucination in large language models* [23]
[7] Lukas Berglund et al. - *The Reversal Curse* [2]

dans l'hypothèse, sans même savoir que le texte était "Thomas s'est rendu au partiel" [8].

Il arrive toutefois que certains LLM, entraînés avec de grandes quantités de données performent tout de même bien sur les tâches d'inférences. C'est par exemple le cas de ChatGPT-3.5 et ses 175 milliards de paramètres [17], qui à première vue semble être capable de capturer les nuances nécessaires pour effectuer des inférences textuelles. Toutefois, il est important de noter que ces même modèles sont également influencés par les biais qui ont été évoqués précédemment. Ces biais peuvent se manifester à travers des nuances telles que la négation et des imprécisions, qui peuvent conduire à des erreurs d'inférence et compromettre la performance globale du modèle [9]. Ainsi, bien que certains LLM puissent obtenir des résultats prometteurs sur les inférences, ces résultats n'en restent pas moins sur-estimés, et il reste donc crucial d'essayer de repousser les limites dans le domaine des inférences textuelles.

3.3 Exploration des approches hybrides

C'est dans un but d'améliorer les performances des modèles de ML qu'interviennent les approches hybrides, qui visent à combiner les avantages du ML avec les capacités expressives de la logique symbolique et de la linguistique. Ces dernières visent à offrir un cadre formel pour raisonner sur les relations sémantiques, ce qui permet de capturer les inférences et raisonnements logiques complexes qui échappent souvent aux modèles utilisant l'apprentissage automatique. En intégrant la logique symbolique, il est possible d'enrichir les modèles de ML avec des règles et des contraintes supplémentaires. Par exemple, dans l'une de ces méthodes, plusieurs modules sont dédiés à des aspects spécifiques de la compréhension des textes, comme l'identification des relations temporelles et des entités nommées. Ces modules sont ensuite fusionnés en utilisant un système de vote, qui combine les prédictions des modules en tenant compte de leur fiabilité respective, pour produire une décision finale [10].

Toutefois, il est important de noter que bien qu'il a été montré que des méthodes symboliques de reconnaissance d'inférences textuelles sont un moyen d'améliorer grandement la compréhension du langage naturel par les LLM, cette dernière n'est pas toujours intégrée dans les grands modèles de langue pour améliorer leurs performances.
Plusieurs raisons expliquent cela. D'une part, l'incorporation des inférences textuelles nécessite d'ajouter des algorithmes complexes, ce qui peut entraîner une augmentation significative du temps de calcul [11]. D'autre part, la disponibilité de données annotées pour les inférences textuelles est souvent limitée, ce qui rend difficile l'entraînement de LLM sur ce genre de tâches [12].

[8] Nanjiang Jiang et al. - *Evaluating bert for NLI, case study on the commitmentbank* [11]
[9] Suchin Gururangan et al. - *Annotation Artifacts in Natural Language Inference Data* [8]
[10] Mohamed H. Haggag et al. - *Different Models and Approaches of RTE* [9]
[11] Mohamed H. Haggag et al. - *Different Models and Approaches of RTE* [9]
[12] Jiyi Li - *A Comparative Study on Annotation Quality of LLM via Label Aggregation* [14]

Une autre approche consiste à explorer les *embeddings* générés par des modèles de langage pré-entraînés. Les *embeddings* sont des représentations de *tokens*, qui peuvent être des mots ou des morphèmes (morceaux de mots), sous forme de vecteurs dans l'espace. Cette représentation vectorielle capture les similitudes sémantiques entre les tokens : des mots ayant des significations similaires auront des vecteurs proches dans l'espace. Comme il est également possible de générer des embeddings pour des phrases entières, l'idée serait d'explorer comment les informations sont traitées dans ces modèles, et de déterminer si les embeddings encodent par défaut les propriétés qui nous intéressent [13].

L'un des modèles le plus utilisés pour cette tâche est *BERT*, un modèle de langage pré-entraîné qui fonctionne en utilisant un transformeur dit bidirectionnel. Cette architecture lui permettant de prendre en compte le contexte précédent et suivant lors de l'encodage des mots dans une phrase [14], à l'inverse des modèles comme GPT, qui sont unidirectionnels et ne lisent le texte que de gauche à droite.
BERT est pré-entraîné avec des techniques spécifiques comme le masquage de certains mots dans une phrase pour que le modèle devine les mots manquants, et la prédiction de la phrase suivante dans un texte, ce qui enrichit sa compréhension contextuelle.

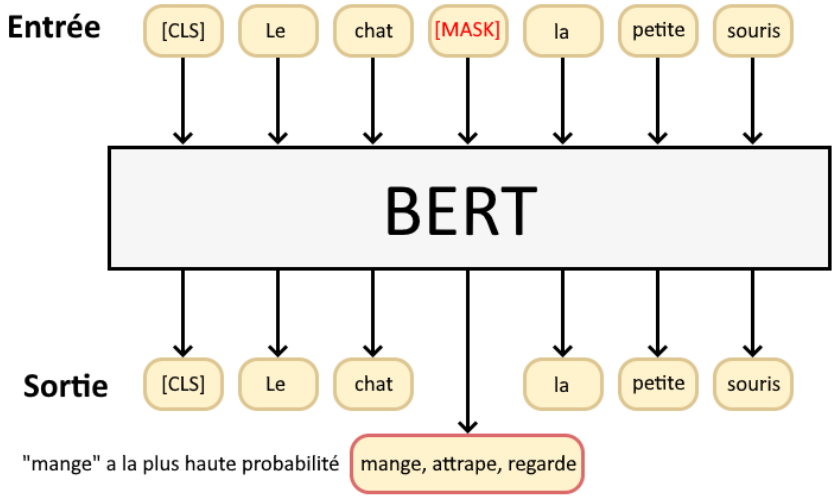


FIGURE 3.2 : Exemple du processus d'entraînement de BERT par masquage de token.

Un aspect clé des modèles de langage pré-entraînés comme BERT est l'utilisation du token Token CLS (pour "Classification token"). Ce token spécial est ajouté au début de chaque texte et permet d'encapsuler l'information pertinente de toute la phrase. La différence avec les embeddings est que ce token a pour but de ne garder que les informations les plus pertinentes et caractéristiques d'une phrase. Lors de l'entraînement, la représentation vectorielle du token CLS est utilisée par le modèle pour effectuer diverses tâches de classification.
Dans le cadre de notre étude des opérateurs argumentatifs, le token CLS peut être uti-

[13] Gabriella Chronis et al. - *A Method for Studying Semantic Construal [...]* [3]
[14] Jacob Devlin et al. - *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding* [7]

lisé en analysant la représentation vectorielle des paires d'énoncés/préjacents. Avec cette méthode, il serait possible d'effectuer des opérations de clustering, une technique qui consiste à regrouper des objets similaires en clusters (ou groupes) qui seraient proches de par leur représentation vectorielle (voir figure 3.3). En utilisant le clustering sur les représentations vectorielles des tokens CLS, il serait possible d'identifier des groupes qui partageraient des caractéristiques argumentatives similaires.

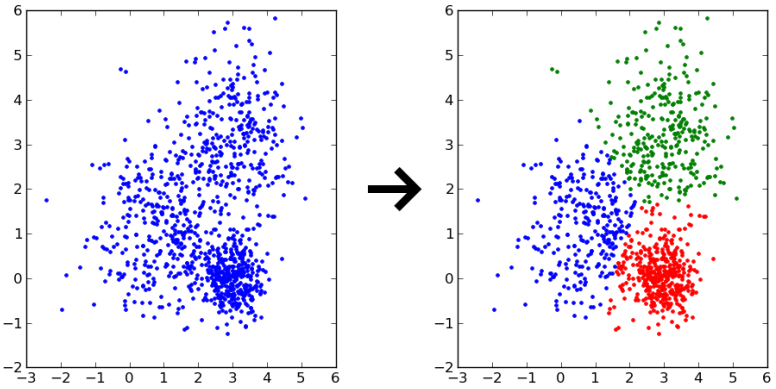


FIGURE 3.3 : Exemple de clustering avec les données initiales (gauche) et les clusters calculés (droite).

Un autre point fort de BERT est qu'il peut également être *fine-tuné* (de l'anglais, *fine-tuning*), c'est-à-dire ajusté après son entraînement initial sur des tâches spécifiques, en utilisant des jeux de données annotés pour adapter ses capacités sur des domaines précis. BERT est généralement utilisé comme modèle de base pour une variété de tâches de traitement du langage naturel, telles que la compréhension, classification et génération de texte.

Cependant, il est important de noter que la grande majorité des modèles de reconnaissance d'inférences textuelles pour le français sont actuellement entraînés sur un corpus de données appelé XNLI, qui contient des données en Anglais [15]. Pour les phases de test, XNLI utilise des traductions manuelles de l'anglais vers le français pour garantir une meilleure qualité des données, mais pour l'entraînement, le corpus dispose de traductions automatiques de l'anglais vers le français. Les traductions automatiques peuvent donc contenir des erreurs qui peuvent affecter la qualité de l'entraînement des modèles. Par conséquent, les performances des modèles de RTE en français pourraient être améliorées en utilisant des données d'entraînement de meilleure qualité, obtenues par des traductions manuelles ou des corpus créés spécifiquement pour le français.

C'est donc afin de surmonter ces défis et d'améliorer la reconnaissance des inférences textuelles, que pour la suite de ce stage, la méthode hybride envisagée vise à développer un système capable de prédire la similarité argumentative des exemples de notre corpus de données. Pour ce faire, nous allons commencer par créer ce corpus, puis entraîner un modèle de *Machine Learning* sur les données de RTE afin d'évaluer ses performances pour ces tâches spécifiques.

[15] Corpus XNLI [4]

4. Création du corpus

4.1 Extraction des phrases

4.1.1 Première possibilité envisagée : CommitmentBank

Une première approche pour constituer un corpus a été de s'orienter vers des données déjà existantes pour pouvoir disposer de phrases déjà annotées. Dans un premier temps, nous avons envisagé d'utiliser le CommitmentBank [11], un corpus de données où chaque "élément" est constitué d'une paire de phrases A et B, et d'une annotation concernant l'implication de A vers B. Ici, l'implication est évaluée entre +3 et -3, et indique le degré d'engagement ("commitment") du locuteur concernant l'information contenue dans la phrase B, dans le contexte de la phrase A. Par exemple, on peut considérer la paire suivante :
A : "Il est parti acheter une voiture à New York"
B : "Il est allé à New York."
Si dans le contexte de A le locuteur considère que "Il est allé à New York" est vrai, alors le score d'implication entre ces deux phrases serait de +3.

Il a donc été question de repartir de ces données et examiner les différents contextes dans lesquels des phrases sont utilisées. Comme les données du CommitmentBank sont en anglais, et que nous nous intéressons spécifiquement aux opérateurs "presque" et "à peine", nous avons comparé ces mots avec leurs équivalents anglais, "*almost*" et "*barely*". Cependant, cette approche n'a pas abouti, car les phrases contenant ces opérateurs dans le CommitmentBank étaient trop peu nombreuses pour constituer un corpus significatif (à peine une vingtaine pour *almost*, et une dizaine pour *barely*). De plus, comme nous savions que ces phrases auraient également nécessité une traduction, nous avons conclu qu'il était préférable de s'orienter vers une autre source de données.

4.1.2 Utilisation de Wikipedia

Pour surmonter les limitations de notre première approche, nous avons décidé d'extraire des données brutes puis de les annoter. Pour l'extraction, nous nous sommes tournés vers Wikipedia, car cette plateforme représente à elle seule une énorme quantité de données textuelles, couvrant ainsi une vaste gamme de sujets. Cette diversité est particulièrement utile pour constituer un corpus contenant des exemples variés de l'utilisation de "presque" et "à peine" pour fournir une représentation la plus *homogène / réaliste* possible.

Pour effectuer l'extraction de phrases contextuelles complètes, nous avons utilisé des techniques de TALN standard pour faire du découpage de phrases, puis récupérer celles qui contenaient les opérateurs souhaités. Cela a été fait en Python dans la fonction *extract_sentences()*, qui prenait en entrée un opérateur puis un texte, et qui renvoyait

une liste de phrases contenant l'opérateur. Comme Wikipedia dispose d'une API proposant diverses fonctionnalités, nous l'avons utilisé pour parcourir les pages de manière aléatoire, et éviter de n'avoir que des articles avec un sujet similaire ou dans le même domaine.
La pipeline principale consistait alors à visiter une page aléatoire, de récupérer le texte et d'en extraire les phrases contenant l'opérateur voulu avec la fonction *extract_sentences()*, puis d'écrire ces phrases ainsi que leur contexte (titre de l'article, phrase avant et après) dans un fichier JSON, en vue d'être traitées ultérieurement.
De plus, nous avons constaté un petit délai (environ 1 seconde) lors d'une requête pour récupérer le texte d'une page. Nous avons alors mis en place du parallélisme, en faisant tourner plusieurs threads en simultanés, pour accélérer le processus d'extraction et également pouvoir travailler avec un plus grand nombre de données.

La pipeline ayant été mise en place pour pouvoir extraire des phrases contenant un opérateur quelconque, nous avons jugé qu'il serait intéressant de ne pas se restreindre à "presque" et "à peine" en vue de potentielles poursuites de travaux. Nous avons alors récupéré des phrases contenant un ensemble ciblé de 6 opérateurs argumentatifs qui sont donc : "presque", "à peine", "juste", "seulement", "seul" ainsi que la négation ("ne ... pas" ou "n'... pas"). Ce choix est basé sur des travaux antérieurs, notamment sur la liste étudiée par Winterstein dans *Argumentative semantics – Meaning with a purpose* [21], qui concerne l'effet de divers modificateurs argumentatifs affectant l'orientation et la force des énoncés, dont nous donnerons plus de détails dans la section 4.4.

4.2 Filtrage des données

Après avoir récupéré 5000 phrases pour chaque opérateur, une partie importante de la création du corpus concernait le filtrage des données extraites. En effet, comme les données allaient être annotées manuellement, et en vue de pouvoir étudier précisément l'effet de chaque opérateur, nous avons pris soin de ne garder que les phrases qui ne contenaient qu'un seul opérateur à la fois. Par exemple, une phrase tirée de l'article "Sophisme" "En allant *à peine* plus loin, on soutiendra la thèse que rien n'est vrai, et que tout est relatif.", a été conservée, tandis que "Walt Disney *ne* cantonne *pas* son personnage aux *seuls* films d'animation." (de l'article sur "Mickey Mouse"), contient deux opérateurs : la négation et "seul", et ne sera donc pas gardé pour créer le corpus.

La seconde étape consiste à récupérer le préjacent de la phrase, donc la phrase sans l'opérateur. Dans la majorité des cas, il suffisait de simplement retirer ce dernier pour obtenir une phrase bien formée ; mais dans des cas plus complexe, comme "75% des aliments de la planète proviennent d'à peine 12 espèces végétales cultivées et 5 animales.", il a fallu s'assurer de remplacer "d'à peine 12 espèces" par "de 12 espèces" pour garder une phrase cohérente. Après ces étapes et élimination des doublons potentiels, le corpus, disponible en section 7.0.3 en annexe, contenait à présent des données propres qui étaient disposées à être annotées.

4.3 Annotation du corpus

L'annotation représente l'étape clé de la création du corpus. Chaque phrase a été examinée pour déterminer la relation argumentative entre l'énoncé et le préjacent. Les relations possibles incluent la "co-orientation", dans le cas où les phrases s'accordent mutuellement sur leurs conclusions argumentatives ; "opposition", où les phrases se contredisent argumentativement ; et "neutre", pour les phrases n'ont pas de relation argumentative claire. Pour éviter que l'étape d'annotation ne soit trop fastidieuse, nous avons codé un script qui simplifie et accélère ce processus, en présentant des paires de phrases extraites automatiquement, et en demandant à l'utilisateur de les annoter en indiquant leur similarité argumentative.

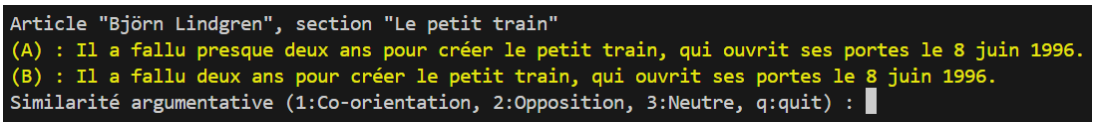


FIGURE 4.1 : Exemple d'annotation d'une phrase avec l'opérateur "presque".

Par exemple, comme illustré dans la figure 4.1, l'utilisateur sélectionne simplement l'option appropriée en fonction de la relation argumentative perçue entre les phrases. Ce système permet de collecter des annotations de manière efficace, pour garantir d'avoir un nombre de données suffisant pour l'analyse ultérieure. Pour commencer, 200 éléments ont été annotés pour "presque", et 200 pour "à peine"

4.4 Description des opérateurs argumentatifs ciblés

Dans cette section, nous décrivons les opérateurs argumentatifs ciblés pour l'annotation, en nous concentrant principalement sur les adverbes *presque* et *à peine*. Ces adverbes jouent un rôle crucial dans la structuration des phrases et dans l'orientation argumentative de ces dernières [1]. Pour mieux comprendre leur fonctionnement, nous allons d'abord nous intéresser aux prédicats sur lesquels portent ces opérateurs, et plus précisément à la gradabilité de ces prédicats.

[1] Kennedy et McNally - *Scale Structure, Degree Modification, and the Semantics of Gradable Predicates* [12]

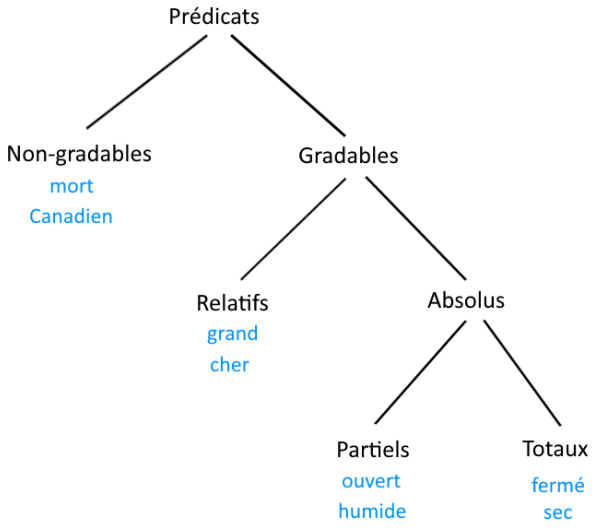


FIGURE 4.2 : Arbre de gradabilité des prédicats.

Les prédicats sont généralement constitués d'un adjectif et expriment une action ou une propriété concernant le sujet de la phrase. En somme, ils correspondent au sens de toute expression qui dénote une propriété ou une relation. Ils peuvent être divisés en deux grandes catégories : non-gradables et gradables. Les prédicats non-gradables, comme "mort" ou "unique", signifient qu'ils ne peuvent pas être comparés entre eux ; il n'est par exemple pas possible de dire qu'une personne est plus "morte" qu'une autre. Par conséquent, ils peuvent être modifiés par des adverbes tels que *presque*, ce qui permet d'indiquer une proximité quant à l'accomplissement d'un état, par exemple dans le cas de "il a été presque élu".

Les prédicats gradables, quant à eux, peuvent être subdivisés en relatifs et absolus. Les prédicats relatifs, tels que "grand", sont évalués par rapport à un standard qui est fixé contextuellement. L'utilisation de *presque* avec ces prédicats n'est généralement pas naturelle ("il est *presque* grand"...). Les prédicats absolus possèdent des limites fixes de gradation et se divisent en deux sous-catégories : partiels et totaux. Les prédicats (gradables) absolus partiels, comme "ouvert", "humide" ou "tordu", témoignent d'une gradation intermédiaire, c'est-à-dire que les prédicats partiels sont jugés vrais dès que le degré de la propriété en question est au-dessus d'un minimum (une porte devient ouverte à l'instant où elle n'est plus complètement fermée), ce qui rend l'utilisation de *presque* assez peu pertinente. Par exemple, dire "presque ouvert" ou "presque tordu" ne semble pas naturel.

En revanche, les prédicats absolus totaux, tels que "fermé" ou "sec" impliquent une gradation complète. Utiliser *presque* avec ces prédicats, comme dans "presque fermé" ou "presque sec", influe sur la gradation vers l'état final. En se basant sur l'arbre de la figure 4.2 qui illustre comment les différents types de prédicats peuvent être classés, nous avons ciblé les adverbes *presque* et *à peine* pour l'annotation car ils illustrent bien les différences dans la gradation des prédicats.

5. Mise en place du système de prédiction

5.1 Choix du modèle et des méthodes de prédiction

Dans le cadre de ce stage, nous avons choisi une méthode de classification afin de pouvoir prédire les relations argumentatives entre les énoncés et étudier les performances d'un modèle pour cette tâche. Le but était donc de développer un modèle capable de distinguer les différentes relations argumentatives que nous avons vues précédemment : "co-orientées" ou "en opposition", qui constitueront les deux classes qui seront annotées. Cette classification vise à évaluer la difficulté de mise en place d'un système de classification, en tenant compte du fait que les relations de ce type pourraient être plus simples à repérer d'un certain point de vue en raison de leur caractère intrinsèquement distributionnel, mais plus complexes par ailleurs en raison de la forte influence du contexte. Comprendre comment les opérateurs argumentatifs comme "presque" et "à peine" influencent les inférences textuelles nous permettra d'améliorer les modèles de prédiction dans des contextes variés.

5.1.1 Sélection du modèle linguistique

Pour notre étude, nous avons choisi d'utiliser le modèle CamemBERT, un modèle de langue pré-entraîné similaire à BERT, mais spécifiquement optimisé pour le français [1]. CamemBERT a été pré-entraîné sur un vaste corpus de textes francophones incluant principalement des sources venant du web, ce qui lui permet de capturer les nuances du français moderne.

Le choix du modèle s'était initialement porté sur FlauBERT [2], mais après différents essais, il est apparu que CamemBERT présentait de meilleurs résultats dans le cadre de notre tâche. En effet, comme nous venons de le voir, CamemBERT ayant été entrainé sur des textes ayant un style d'écriture plus récent (sources web, etc.), à l'inverse de FlauBERT qui a été entraîné sur des textes ayant un style d'écriture plus "formel" (livres, articles de journaux, etc...). CamemBERT semble alors mieux adapté pour traiter et analyser le contenu de Wikipedia, car initialement conçu pour optimiser les performances sur des tâches de TALN en français.

5.1.2 Méthode de prédiction

Pour la prédiction, nous avons dans un premier temps utilisé une approche d'apprentissage automatique classique avant d'utiliser des méthodes symboliques. La pipeline pour l'entraînement du modèle de prédiction comprend les étapes suivantes :

[1] CamemBERT : a Tasty French Language Model [16]
[2] FlauBERT : Unsupervised Language Model Pre-training for French [13]

1. **Prétraitement des données** : Les phrases du corpus ont été formatées pour être compatibles avec CamemBERT. Cela inclut la tokenisation, qui consiste à diviser le texte en tokens, et l'encodage des phrases, qui désigne le processus de conversion des tokens en nombres, car c'est la seule chose que le modèle peut comprendre.

2. **Fine-tuning du modèle** : CamemBERT a ensuite été fine-tuné sur notre corpus. Pour ce faire, nous avons utilisé notre corpus annoté pour entraîner le modèle à reconnaître et classer les relations argumentatives entre les énoncés.

3. **Validation croisée** : Nous avons utilisé la validation croisée (K-fold), qui consiste à diviser l'ensemble de données en sous-ensembles, ou "folds", et à entraîner le modèle sur ces différents folds. Cette méthode permet de vérifier que le modèle fonctionne bien sur toutes les données et pas seulement sur un seul jeu de test.

4. **Entraînement du modèle** : Pour chaque fold, le modèle est initialisé et entraîné sur le sous-ensemble des données d'entraînement. L'entraînement se déroule en plusieurs étapes, pendant lesquelles le modèle apprend à reconnaître des patterns dans les données. Pour éviter le surapprentissage et optimiser le temps d'entraînement, nous avons mis en place un mécanisme d'arrêt anticipé ("*early stopping*"), pour s'assurer que le modèle ne continue pas à s'entraîner inutilement si ses performances sont satisfaisantes.

5. **Évaluation** : Après chaque itération d'entraînement, le modèle est évalué sur l'ensemble de test. Les métriques d'évaluation ainsi que les loss pour chaque folds sont enregistrés, et les modèles les plus performants parmi les folds ont été sauvegardés pour une utilisation future.

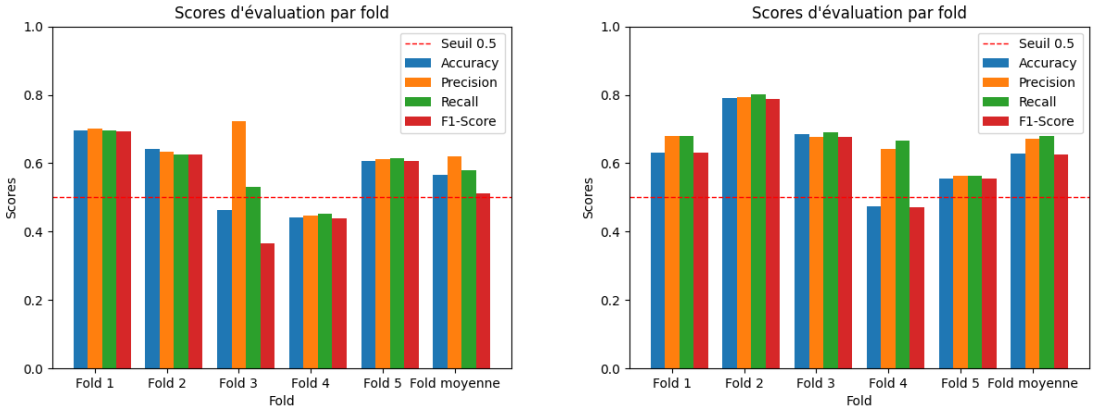
Cette méthode de prédiction nous a permis de tirer parti des capacités avancées de CamemBERT pour classifier les relations argumentatives dans notre corpus. Nous allons maintenant détailler l'évaluation des performances du modèle pour mesurer son efficacité pour notre tâche de reconnaissance d'inférences textuelles.

5.2 Évaluation des performances

Pour évaluer les performances de notre modèle, nous avons dans un premier temps évalué son accuracy, qui mesure le pourcentage de prédictions correctes. Cependant, l'accuracy peut être trompeuse dans des situations où les classes sont déséquilibrées. En effet, parmi les 200 éléments annotés pour l'opérateur "presque", 70% étaient co-orientés, contre 30% de négation. Évaluer le modèle avec l'accuracy n'est donc pas pertinent, dans le sens où le modèle pourrait se contenter de prédire tout le temps "co-orientation" sans réellement apprendre à distinguer les deux classes, et obtenir une accuracy de 70%.

Pour surmonter cette limitation, nous avons utilisé d'autres métriques d'évaluation qui offrent une meilleure perspective des performances du modèle : la précision, le rappel, le F1-score et l'AUC (détail en annexes). Nous pouvons à présent analyser les premiers résultats de l'entraînement.

Comme les métriques pour les différents folds sont assez variables, nous nous intéressons en priorité à la moyenne des résultats. Nous pouvons dans un premier temps



(a) Métriques pour "presque" (b) Métriques pour "à peine"

FIGURE 5.1 : Métriques de l'entraînement sur 5 folds pour les opérateurs

constater que le modèle entraîné sur le corpus de "à peine" performe bien mieux que celui entraîné sur "presque", avec respectivement une AUC à 0.68 et 0.60. L'AUC mesure la capacité du modèle à distinguer entre les classes, donc dans notre cas, une AUC de 0.68 indique que le modèle entraîné sur "à peine" a une meilleure capacité de discrimination entre les classes par rapport à un modèle qui prédirait de manière aléatoire (AUC de 0.5). Ces valeurs signifient que bien que le modèle puisse distinguer les classes, il y a encore une marge d'amélioration, en particulier pour le corpus de "presque".

Les différences de performance entre les modèles peuvent être attribuées à plusieurs facteurs. Premièrement, "à peine" est souvent utilisé dans des contextes où la négation est implicite, ce qui peut rendre les relations argumentatives plus claires pour le modèle. En revanche, "presque" modifie souvent des prédicats relatifs dont la gradation dépend fortement du contexte, ce qui peut introduire une plus grande variabilité dans les données, rendant les inférences plus difficiles à capturer.

5.3 Améliorations du modèle

Pour améliorer les performances de notre modèle, nous avons commencé par visualiser les données. Une première approche nous avons adoptée a été d'utiliser le token CLS (défini en section 3.3) pour effectuer du clustering. Les représentations vectorielles des tokens CLS pour chaque phrase du corpus on été extraites et projetées en 2 dimensions en utilisant un algorithme de réduction de dimension (PCA) pour pouvoir les visualiser.

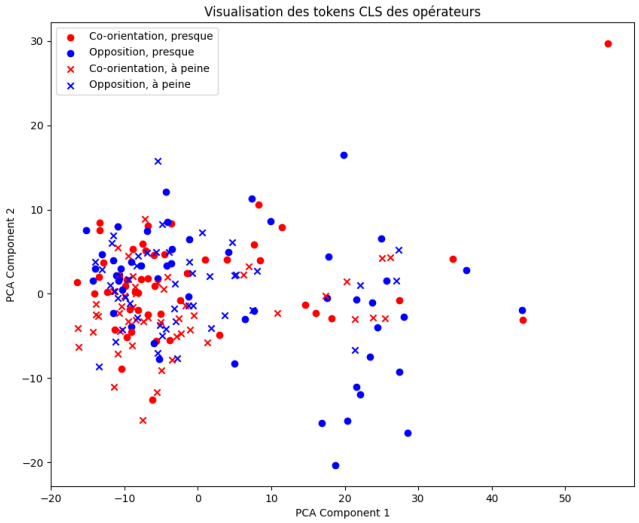


FIGURE 5.2 : Visualisation des CLS selon l'opérateur et la similarité argumentative. Les dimensions ne sont pas interprétables

Dans la figure 5.2, on constate que, même si l'on remarque un regroupements de points plutôt sur la gauche du graphique, la répartition des tokens CLS dans l'espace ne semble pas former de cluster clairs. Cette intuition est confirmée lorsque l'on tente de former des clusters directement à partir des vecteurs des tokens CLS (en 768 dimensions pour BERT/CamemBERT).

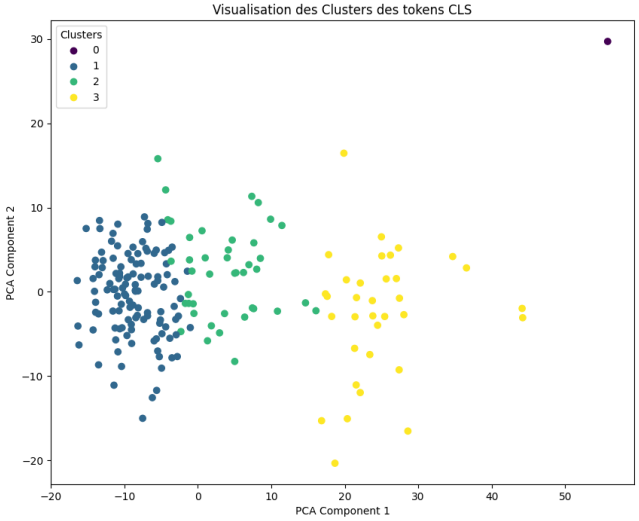


FIGURE 5.3 : Clusters des tokens CLS calculés avec KMeans

Nous pouvons voir sur la figure 5.3 que les clusters calculés ne correspondent pas du tout aux données réelles de la figure 5.2. L'interprétation que nous pouvons faire est que les représentations des tokens CLS ne capturent pas efficacement les distinctions argumentatives entre les opérateurs. De plus, cela suggère que CamemBERT peut avoir des

difficultés à distinguer les subtilités de ces opérateurs dans le cas d'inférences textuelles.

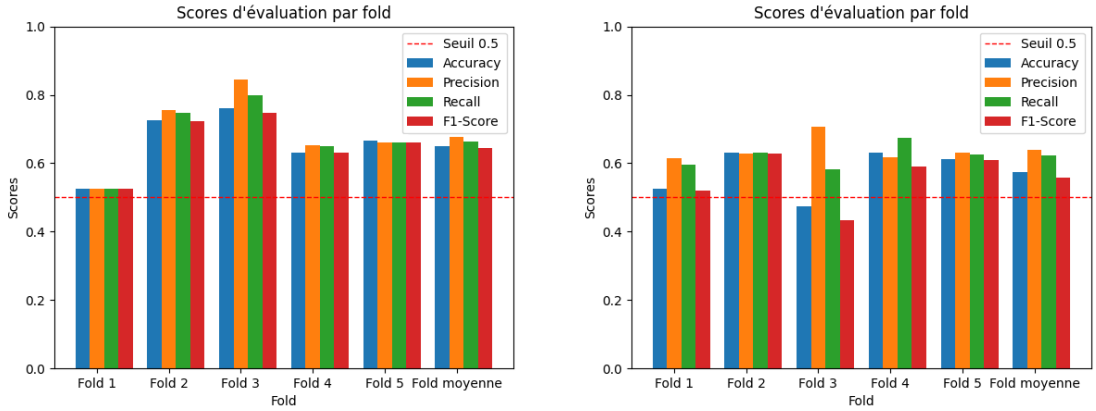
La seconde approche que nous avons utilisée fait maintenant intervenir des méthodes hybrides, en intégrant des méthodes de linguistiques au modèle. Nous avons utilisé Spacy pour l'analyse syntaxique, ce qui nous a permis de fournir au modèle des informations syntaxiques supplémentaires. Spacy est un outil pour le TALN qui peut analyser la structure grammaticale des phrases et identifier les catégories de mots (noms, verbes, adjectifs, etc...).
En se servant de Spacy pour extraire les étiquettes des catégories grammaticales de chaque mot dans les phrases, ces informations ont été ajoutées en tant que caractéristiques supplémentaires lors de l'entraînement du modèle. Par exemple, pour une phrase comme "La Restauration des Stuart en 1660 durera *à peine* trente ans.", après l'analyse grammaticale de Spacy, la phrase ressemblera à "La (DET) Restauration (NOUN) des (ADP) Stuart (PROPN) en (ADP) 1660 (NUM) durera (VERB) *à (ADP) peine (NOUN)* trente (NUM) ans (NOUN) .". Nous nous attendons à ce que ajouter ces informations syntaxiques soit pertinent, car elles devraient permettre au modèle de mieux comprendre la structure et la fonction des différents mots dans une phrase, et donc améliorer sa capacité à capturer les relations sémantiques complexes.

Avant d'entraîner directement le modèle sur ces nouvelles données, nous avons également voulu résoudre un autre problème : le déséquilibre des classes. Comme nous l'avons vu précédemment pour "presque", le nombre d'éléments annotés comme "co-orientation" était *presque* trois fois supérieur au nombre d'annotation en "opposition". Pour remédier à ce déséquilibre, nous avons d'abord utilisé une technique de downsampling, qui consiste à réduire le nombre d'éléments dans la classe majoritaire pour égaler celui de la classe minoritaire, pour obtenir un ensemble de données équilibré. En réduisant le nombre d'annotations "co-orientation" au même niveau que celui des annotations "opposition", nous pouvons entraîner le modèle de manière plus équilibrée, et ainsi éviter qu'il ne devienne biaisé en faveur de la classe de co-orientation.

C'est donc après cela que le modèle a de nouveau été entraîné, en testant individuellement chaque ajout pour évaluer leur impact respectif.

Nous constatons qu'après l'ajout des étiquettes grammaticales, les métriques (en figure 5.4) sont meilleures pour "presque", mais moins bonnes pour "à peine". Pour "presque", les performances du modèle ont significativement augmenté, avec en moyenne 14.4% d'amélioration pour chaque métrique, indiquant que ces informations supplémentaires ont aidé le modèle à mieux capturer les nuances des relations argumentatives pour cet opérateur. En revanche, pour "à peine", les performances ont diminué en moyenne de 7.8%, montrant que les étiquettes grammaticales n'ont pas apporté les mêmes bénéfices.

Pour "presque", les catégories grammaticales ajoutent une couche de compréhension qui semble aller dans le même sens que les inférences textuelles attendues. Cependant pour "à peine", il est possible que, d'une part, ses usages contextuels variés rendent les étiquettes grammaticales moins utiles ou même "perturbantes" pour le modèle. De plus, la complexité de l'opérateur en lui même est à prendre en compte, à savoir que "à peine" est constitué de deux mots/token. Bien que l'ensemble de l'expression fonctionne comme un adverbe, "à peine" est analysé syntaxiquement comme une expression composée d'une



(a) Métriques pour "presque_spacy"

(b) Métriques pour "à_peine_spacy"

FIGURE 5.4 : Métriques de l'entraînement après rajout des catégories grammaticales

préposition, "à", suivie d'un nom, "peine". Cette complexité a par ailleurs été démontrée dans une étude qui a montré que la sensibilité des modèles de langue aux opérateurs argumentatifs peut varier significativement en fonction de la construction sémantique des expressions utilisées, ce qui explique en partie les résultats observés pour "à peine" [3].

En conclusion, l'impact différent sur les performances du modèle en fonction des opérateurs étudiés, après l'ajout des informations grammaticales, suggère qu'une étude plus approfondie de cette approche pourrait être nécessaire, notamment en entraînant un modèle sur d'autres opérateurs, comme "seulement" ou "juste", et en adaptant les caractéristiques qui sont ajoutées en fonction des spécificités de chaque opérateur pour maximiser les performances du modèle.

5.4 Axes d'étude parallèles

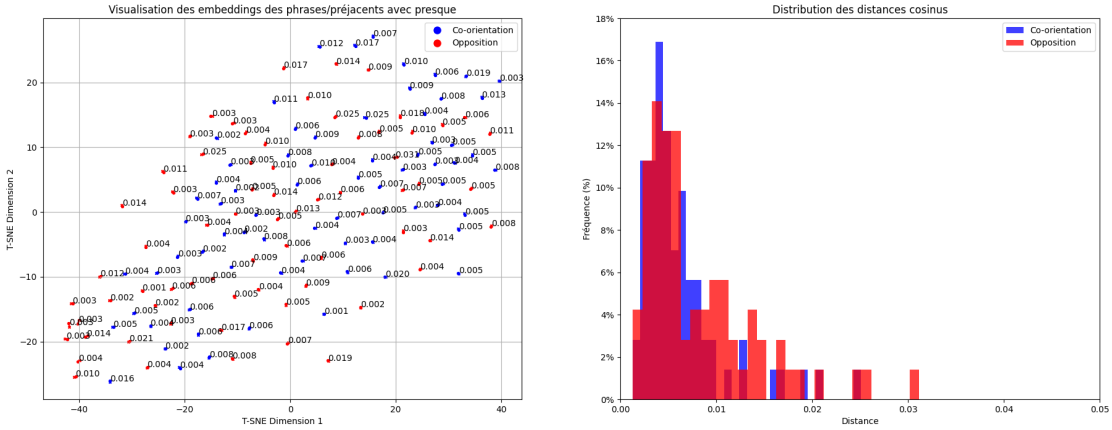
Même si le downsampling représente une solution pour lutter contre le déséquilibre des classes, ce dernier a toutefois l'inconvénient de réduire la quantité totale de données disponibles pour l'entraînement, ce qui peut potentiellement limiter les performances du modèle. Pour surmonter cette limitation, nous prévoyons d'utiliser à l'avenir une technique d'upsampling, en générant artificiellement des exemples supplémentaires pour la classe minoritaire. La procédure vise à utiliser l'API GPT-3.5 d'Openai, en générant des phrases similaires à celles annotées comme "opposition", en vue d'équilibrer le jeu de données. Cette approche sera explorée après le rendu de ce rapport, car elle nécessite des tests supplémentaires pour garantir la qualité des phrases générées.

Afin de mieux comprendre la performance du modèle et les relations entre les phrases et leurs préjacents, nous avons étudié les embeddings des phrases et leurs préjacents. Pour ce faire, nous avons d'abord récupéré les phrases et leurs préjacents, en veillant à

[3] Informational content vs. discourse orientation : experimental and computational perspectives [22]

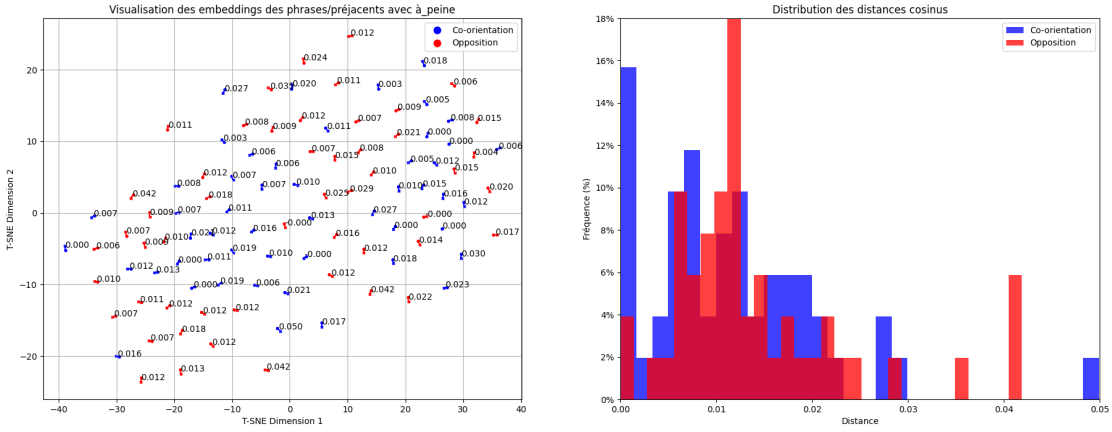
équilibrer les classes "Co-orientation" et "Opposition", puis ces phrases on été converties en embeddings, donc en vecteurs de grande dimension. A partir de ces vecteurs, nous avons calculé la similarité entre chaque paire, à partir de la distance cosinus, pour savoir si les vecteurs des phrases co-orientées étaient alors plus proches dans l'espace que ceux des phrases en opposition.

Pour visualiser ces embeddings, nous avons utilisé T-SNE, qui permet de projeter les vecteurs dans un espace en 2D tout en essayant de préserver au mieux la structure des vecteurs originaux.



(a) Visualisation des embeddings de "presque" (b) Distribution des distances cosinus

FIGURE 5.5 : Visualisation des embeddings dans l'espace pour les phrases avec "presque"



(a) Visualisation des embeddings de "à peine" (b) Distribution des distances cosinus

FIGURE 5.6 : Visualisation des embeddings dans l'espace pour les phrases avec "à peine"

Les visualisations montrent dans un premier temps que les phrases et des préjacents ne forment pas de clusters distincts, ce qui vient appuyer nos observation précédentes avec le token CLS, suggérant que les relations argumentatives ne sont pas non plus capturées par les embeddings. Cependant, nous constatons que les distances cosinus

montrent une distribution qui diffère, dans un premier temps selon l'opérateur, mais aussi selon l'orientation argumentative.

En effet nous pouvons d'abord observer que, pour des longueurs moyenne de phrase similaires, les paires énoncés/préjacents avec "presque" tendent à être plus proches que celles avec "à peine", avec une distance moyenne de respectivement 0.007 contre 0.013 (chiffres en section 7.0.3). Ce premier écart peut sans doute s'expliquer de la même façon que précédemment, à savoir que l'expression "à peine" étant en deux mot, le modèle serait plus sensible a son retrait que pour "presque".

De plus, pour "presque" comme pour "à peine", les paires en co-orientation sont plus proches que celles en opposition. Cette différence, bien que présente, est très faible (0.002) et peut ne pas être significative. Cela indique que le modèle a du mal à capturer les différences sémantiques entre co-orientation et opposition.

En conclusion, bien que notre approche ait montré des limites dans la capture des relations argumentatives, elle ouvre la voie à des explorations futures. Il pourrait être intéressant d'explorer une approche similaire avec des modèles plus sophistiqués ou en intégrant des caractéristiques supplémentaires pour mieux représenter ces nuances sémantiques.

6. Résultats et discussion

6.1 Analyse et interprétation des résultats obtenus

Pour essayer de comprendre pourquoi le modèle avait de telles performances, nous avons voulu effectuer une analyse plus approfondie des données et des prédictions du modèle. En effet, les modèles de machine learning sont souvent considérés comme des "boîtes noires", ce qui rend leur explicabilité difficile, et il est donc crucial d'utiliser des méthodes qui permettent de mieux interpréter les décisions prises par le modèle.

Nous avons pour ce faire utilisé *LIME*, un outil qui nous a permis d'extraire des informations sur les phrases pour lesquelles le modèle s'était trompé. Voici un exemple de sortie renvoyée après l'entraînement :

```
Text: La jeune fille à qui il est venu en aide a à peine 16 ans.
↪  Malgré son histoire difficile, elle chante d'une belle voix,
↪  c'est pourquoi on la surnomme la Goualeuse. [SEP] La jeune fille
↪  à qui il est venu en aide a 16 ans. Malgré son histoire
↪  difficile, elle chante d'une belle voix, c'est pourquoi on la
↪  surnomme la Goualeuse.
Predicted: 1, True: 0
Explanation: [('16', -0.179292454344653), ('ans',
↪  -0.1528796256380854), ('difficile', 0.144247597727433),
↪  ('Goualeuse', 0.137422913123427), ('venu', -0.0899383035114461)]
```

FIGURE 6.1 : Exemple de sortie de LIME avec une prédiction incorrecte du modèle.

Dans cet exemple, le modèle a prédit que les deux phrases étaient en opposition (correspondant à la classe 1), alors qu'elles étaient co-orientées (classe 0). L'explication fournie par LIME montre les mots qui ont le plus influencé la décision du modèle, ainsi que leurs poids respectifs. Les mots tels que "16", "ans", et "venu" ont des poids négatifs, indiquant qu'ils ont influencé le modèle pour faire une prédiction incorrecte, tandis que "difficile" ou "Goualeuse" ont des poids positifs, qui auraient poussé le modèle dans la bonne direction si ces poids étaient plus élevés. À partir de ces prédictions, nous avons une nouvelle fois utilisé Spacy pour analyser plus en détail là où le modèle s'était trompé. En combinant les explications de LIME avec l'analyse syntaxique de Spacy, nous avons voulu d'identifier les types de mots sur lequel le modèle faisait le plus d'erreur.

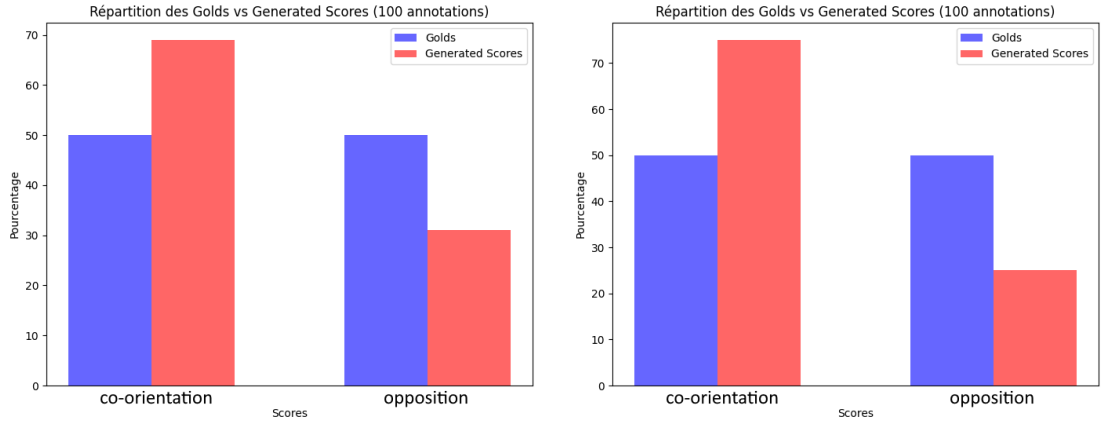
Après analyse, notre première observation se trouve dans le fait que, pour les deux opérateurs, le modèle à beaucoup plus tendance à prédire des faux positif, à savoir "opposition", au lieu de "co-orientation" : dans 96% des cas pour "presque", et 66% pour "à peine". Cela pourrait s'expliquer par le fait que le modèle ne capture pas bien les nuances contextuelles propres à ces opérateurs, même s'il a été noté que "presque" est souvent cohérent dans les contextes où il est utilisé, car il marque une proximité à une

limite ou un seuil. Par exemple, "J'ai presque fini" indique clairement que l'action est sur le point d'être complétée, ce qui implique que la plupart du temps les énoncés soient en co-orientation. Cependant, "à peine" est plus ambigu et dépend fortement du contexte, car il peut soit atténuer soit renforcer l'affirmation, rendant la tâche de prédiction plus complexe pour le modèle.

Cependant, il est difficile de juger des performances de notre modèle étant donné que nous n'avons pas de baseline à laquelle nous comparer. En effet, aucune tâche de classification similaire n'a été étudié pour ces opérateurs argumentatif. Naturellement, ce résultats semblent peu satisfaisants en comparaison avec d'autre modèles de classification ou de reconnaissance d'inférences textuelles, qui peuvent atteindre des scores de précision ou d'accuracy dépassant les 95% ; ainsi que de notre point de vue humain, où cette tâche nous semble plutôt facile, ce qui a par exemple été constaté lors de l'étape d'annotation. Cependant, nous pouvons obtenir une baseline en effectuant la tâche avec un LLM ayant généralement de bonnes performances dans ce domaine.

Notre choix s'est porté sur GPT-3.5, l'API d'OpenAI, pour générer des annotations pour nos éléments déjà labelisés avec "co-orientation" ou "opposition". Pour ce faire, nous avons détaillé à GPT le processus de d'annotation pour ces opérateurs (procédure en annexe en section 7.0.3), puis nous lui avons passé successivement des éléments du corpus pour qu'il les annote. Pour éviter les biais, nous lui avons fourni 50 éléments co-orientés, et 50 en opposition, et nous avons par la suite comparé ses prédictions avec celles de notre modèle.

La première constatation que nous faisons est que le modèle semble biaisé sur la proportion de données co-orientées/en opposition.



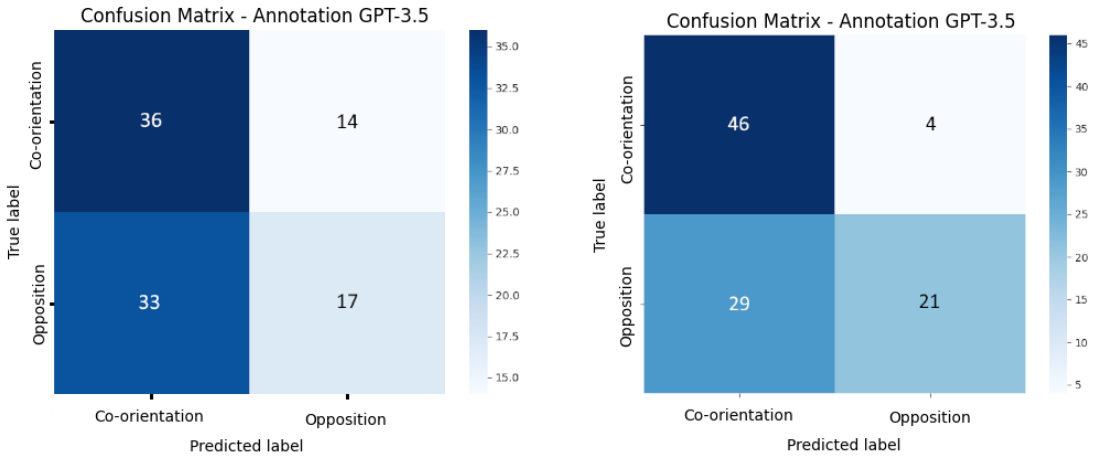
(a) Annotations pour "presque" (b) Annotations pour "à peine"

FIGURE 6.2 : Annotations de 100 éléments du corpus avec GPT-3.5. "Gold" (en bleu) correspond au vrai label, tandis que "generated score" (en rouge) est celui généré.

Comme nous pouvons le voir sur la figure 6.2, le modèle tend à prédire 70% des cas comme de la co-orientation, que ça soit pour les exemple avec "presque" comme "à peine". Même si ce biais pouvait s'avérer fondé lors de rencontre avec "presque" dans divers contexte, ce n'est pas le cas pour "à peine", et encore moins étant donné le fait

que ici, le nombre d'exemples à annoter est équilibré entre co-orientation et opposition.

De plus, si l'on s'intéresse aux annotations générées pour chaque élément à la figure 6.3, nous pouvons constater que le modèle tend à prédire les phrases comme co-orientées la plupart du temps, en négligeant les cas où elles seraient en opposition.



(a) Annotations pour "presque" (b) Annotations pour "à peine"

FIGURE 6.3 : Matrice de confusion des annotations de GPT.

En termes d'accuracy, GPT-3.5 performe à 67% de classification correcte pour "à peine", et de 53% pour "presque". Ces résultats nous permettant d'obtenir une baseline qui semble représentative des capacités actuelles, et témoignent ainsi d'un manque de performance pour cette tâche pour des grands modèles de langues tels que GPT, soulignant la nécessité de méthodes plus sophistiquées pour une reconnaissance précise des inférences textuelles.

6.2 Limitations de l'étude et regard critique

Une des principales limitations de cette étude concerne la taille du corpus. En effet, la création et l'annotation manuelle des données sont des tâches chronophages, ce qui a limité le nombre total de phrases annotées. Avec un corpus de petite taille, les modèles de machine learning ont moins de données sur lesquelles s'entraîner, ce qui peut affecter leurs performances. Il est probable qu'avec un corpus plus grand, les résultats auraient été quelque peu meilleurs, car cela permettrait au modèle de capturer plus de nuances dans les relations argumentatives.

Une autre limitation importante concerne le processus d'annotation lui-même. Les données de notre corpus ont été annotées par une seule personne, ce qui pourrait introduire des biais et des erreurs. Il serait préférable d'utiliser une méthode comme celle utilisée pour des corpus comme le CommitmentBank, où les données sont annotées par plusieurs personnes, souvent des linguistes professionnels, et où une moyenne des résultats est calculée pour garantir une plus grande objectivité des annotations.

Enfin, il est important de noter que les modèles de machine learning, sont souvent considérés comme des "boîtes noires", car il est difficile très d'expliquer comment ils

font leurs prédictions, ce qui complique l'interprétation des résultats. Si nous avions utilisé des règles purement logiques ou linguistiques, nous aurions peut-être pu tirer des conclusions générales sur les opérateurs argumentatifs "presque" et "à peine", ce qui n'est pas le cas pour notre modèle de ML.

Ces limitations sont importantes à considérer, car elles affectent directement nos résultats. Nous reconnaissons leur influence sur cette étude, et ces contraintes devraient être prises en compte en priorité pour les travaux futurs dans ce domaine.

7. Bilan

7.0.1 Réalisation des objectifs initiaux

Après trois mois au sein de l'équipe TEXTE et ADVANSE du LIRMM, ainsi que deux mois et demi dans l'équipe SLIC de l'UQAM, j'ai pu travailler sur l'ensemble des objectifs initiaux de mon sujet de stage, ainsi qu'explorer des pistes alternatives.

Nous avons commencé par utiliser les approches existantes pour la reconnaissance des inférences textuelles, en nous concentrant particulièrement sur les opérateurs argumentatifs "presque" et "à peine", puis nous avons constitué un corpus annoté en utilisant des données issues de Wikipedia, après avoir constaté que les données du CommitmentBank étaient insuffisantes. En parallèle, nous avons intégré une approche hybride en utilisant des techniques d'analyse syntaxique pour enrichir les caractéristiques d'entrée du modèle.

Pour finir, nous avons également exploré d'autres hypothèses, notamment au travers de l'analyse des embeddings et des distances entre les phrases, en passant par de la génération d'annotations à l'aide de LLM.

7.0.2 Connaissances acquises

Au cours de ce stage, j'ai acquis de nombreuses connaissances dans le domaine du TALN et du machine learning. Travailler sur la reconnaissance des inférences textuelles m'a permis de me familiariser avec les LLM comme BERT et CamemBERT, ainsi que les techniques de fine-tuning spécifiques à ces modèles. J'ai également pu apprendre à utiliser des outils d'analyse syntaxiques tels que Spacy pour intégrer ces analyses dans un pipeline de ML pour enrichir les informations données au modèle.

De plus et en parallèle des aspects techniques, ce stage m'a également permis d'appréhender des réflexions linguistiques, en particulier sur les opérateurs argumentatifs "presque" et "à peine". J'ai pu explorer comment ces opérateurs influencent la structure discursive et les inférences textuelles, mais aussi approfondir mes connaissances en linguistique sur la compréhension des relations sémantiques, ainsi que des différents mécanismes qui contribuent à la construction du sens dans le langage. Ces réflexions ont notamment eu leur rôle à jouer lors de l'annotation du corpus et de l'affinage du modèle, en tenant compte des nuances propres à chaque opérateur argumentatif.

7.0.3 Apport personnel et projet professionnel

Ce stage a été très enrichissant sur le plan personnel et professionnel. Il m'a permis de renforcer mes compétences en TALN et en ML, ainsi que de développer une meilleure compréhension des approches hybrides.

Sur le plan professionnel, ce stage a confirmé mon intérêt pour la recherche en informatique et m'a motivé à poursuivre dans cette voie, notamment par le biais d'une thèse.

De plus, durant ces deux mois et demi au sein de l'UQAM à Montréal, j'ai particulièrement apprécié la mentalité ouverte et collaborative des étudiants et chercheurs présents sur place. J'ai pu notamment effectuer une présentation de mon travail lors du congrés "SALU" (Symposium Académique de Linguistique de l'UQAM), ayant pour but de favoriser la collaboration intradépartementale entre les différents laboratoires du Département de linguistique. Cette intégration m'a permis de remettre en perspective mes projets professionnels, et envisager de faire une thèse au Québec pourrait être une excellente option, car cela me permettrait de bénéficier de conditions favorables tout en pouvant explorer de nouvelles collaborations.

En somme, ce stage a été une étape cruciale dans mon développement académique, professionnel et personnel, et a renforcé ma détermination à poursuivre une carrière dans le secteur public de la recherche.

Bibliographie

[1] Jean-Claude Anscombre and Oswald Ducrot. *L'argumentation dans la langue*. Pierre Mardaga, Liège, Bruxelles, 1983.

[2] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The Reversal Curse : LLMs trained on "A is B" fail to learn "B is A", 2023.

[3] Gabriella Chronis, Kyle Mahowald, and Katrin Erk. A method for studying semantic construal in grammatical constructions with interpretable contextual embedding spaces, 2023.

[4] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli : Evaluating cross-lingual sentence representations, 2018.

[5] Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, John Robinson, Ed Stabler, Rosemary Stevenson, and Wolfgang Wahlster. The Fracas Consortium : Framework for Computational Semantics (FraCas), 1996.

[6] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Zanzotto. *Recognizing Textual entailment : models and applications*. Morgan & Claypool Publishers, 2013.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[9] Mohamed Hassan Haggag, Marwa M. A. Elfattah, and Ahmed Mohammed Ahmed. Different models and approaches of textual entailment recognition. *International Journal of Computer Applications*, 142 :32–39, 2016.

[10] P. Hitzler and M.K. Sarker. *Neuro-Symbolic Artificial Intelligence : The State of the Art*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2022.

[11] Nanjiang Jiang and Marie-Catherine de Marneffe. Evaluating bert for natural language inference, a case study on the commitmentbank. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*, pages 6086–6091, 2019.

[12] Christopher Kennedy and Louise McNally. Scale structure, degree modification, and the semantics of gradable predicates. *Language*, 81(2) :345–381, 2005.

[13] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. Flaubert : Unsupervised language model pre-training for french, 2019.

[14] Jiyi Li. A comparative study on annotation quality of crowdsourcing and llm via label aggregation, 2024.

[15] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A Semantically and Syntactically Rich Dataset for Textual Entailment and Compositional Distributional Semantics. *Language Resources and Evaluation*.

[16] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT : a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics.

[17] OpenAI. Chatgpt : A large-scale generative model for open-domain chat. `https://github.com/openai/gpt-3`, 2021.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[19] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue : A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[20] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE : A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[21] Grégoire Winterstein. *Argumentative semantics – Meaning with a purpose*. Oxford University Press, 2023. under contract.

[22] Grégoire Winterstein, Ghyslain Cantin-Savoie, Samuel Laperle, Josiane Van Dorpe, and Nora Villeneuve. Informational content vs. discourse orientation : experimental and computational perspectives. *Experiments in Linguistic Meaning*, 2 :299, 01 2023.

[23] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the AI ocean : A survey on hallucination in large language models. *arXiv preprint arXiv :2309.01219*, 2023.

Figures

- Logo UM, 2020, via umontpellier.fr
- Logo UQÀM, 2011, via uqam.ca/
- Figure 3.1, Exemple inventé pour illustrer l'erreur du processus d'inversion, par Gatien HADDAD.
- Figure 3.3, Exemple de clustering, via Apprentissage profond en bref (slide 3).
- Toutes les autres figures sont soit des captures d'écran, soit des illustrations faites par Gatien HADDAD.

Glossaire

API Ensemble de protocoles permettant à des applications de communiquer entre elles. 15

baseline Référence de performance initiale utilisée pour comparer et évaluer l'amélioration des modèles plus avancés. 27

classification Technique de ML qui consiste à entraîner un modèle pour attribuer des étiquettes prédéfinies à de nouvelles données en se basant sur des exemples d'entraînement. 12, 18

clustering Technique de ML qui consiste à regrouper des objets similaires en groupes, pour permettre d'identifier des similarités entre les données, et faciliter leur interprétation. 13

downsampling Réduction du nombre d'éléments dans une classe majoritaire pour égaler la classe minoritaire et obtenir un ensemble de données équilibré. 22

embeddings Représentations vectorielles de mots ou de phrases dans l'espace, permettant de capturer les significations sémantiques et contextuelles des termes. 12

fine-tuné Le "fine-tuning" désigne le processus d'ajustement d'un modèle de langue sur des tâches spécifiques, en utilisant des jeux de données annotés pour améliorer ses performances sur ces tâches. 13, 19

LLM "Large Language Model", c'est un type de modèle de traitement automatique du langage naturel conçu pour comprendre et générer du texte. 1, 9

loss En machine learning, la *loss* (ou fonction de perte) quantifie l'écart entre les prédictions du modèle et les valeurs réelles. L'objectif de l'entraînement du modèle est de minimiser cette fonction de perte pour améliorer la précision des prédictions.. 19

ML "Machine Learning", sous-domaine de l'intelligence artificielle qui se concentre sur des techniques permettant aux ordinateurs d'apprendre à partir de données, et de faire des prédictions basées sur ces données. 6

RTE "Recognizing Textual Entailment", acronyme anglais de "reconnaisance inférences textuelles". Tâche consistant à déterminer si un texte (l'hypothèse) peut être inféré ou est logiquement impliqué par un autre texte (la prémisse). 1, 13

surapprentissage Phénomène en ML où un modèle s'ajuste trop fortement aux données d'entraînement au lieu de généraliser à de nouvelles données non vues. Cela conduit à de mauvaises performances sur des ensembles de test ou des données réelles.. 19

TALN "Traitement Automatique du Langage Naturel", domaine qui vise à développer des algorithmes permettant aux machines de comprendre, interpréter et générer du langage humain. 1

threads Un thread (ou "fil d'exécution") représente un chemin d'exécution au sein d'un processus. La gestion des threads de manière simultanée permet l'exécution de plusieurs tâches au sein d'un même programme, permettant d'améliorer ainsi l'efficacité des applications. 15

token Unité de base dans le traitement du langage naturel, correspondant généralement à un mot, un signe de ponctuation ou un symbole dans un texte. 12

Token CLS Un token spécial utilisé dans les modèles de langage comme BERT, qui est ajouté au début d'une phrase et dont la représentation sous forme de vecteur est utilisée pour les tâches de classification. 12

upsampling Augmentation du nombre d'éléments dans une classe minoritaire en générant des exemples supplémentaires. 23

Annexe

Données, corpus et code

- Le code utilisé pour ce projet est disponible sur gite.lirmm.fr
- Les données brutes extraites depuis Wikipedia, qui contiennent également des phrases avec d'autres opérateurs, sont présentes dans extracted/wikipedia/
- Les corpus annotés sont disponible dans le dossier extracted/annotated/

Annotations à l'aide de GPT-3.5

Listing 7.1 : Prompt envoyé à l'API de GPT-3.5 pour annoter les paires de phrases

```
context = "Votre but est d'annoter des énoncés en fonction de leur
    similarité argumentative, suivez cette procédure :
Contexte : Considérez le contexte donné, composé de phrases précédant et/
    ou suivant les énoncés A et B.
Répondez avec la relation entre les énoncés A et B dans ce contexte, en r
    épondant uniquement "Co-orientation" ou "Opposition".
Interprétation des relations :
- Co-orientation : Les deux énoncés sont alignés argumentativement, ils
    sont interchangeables sans altérer l'intention de l'énonciateur.
- Opposition : Les deux énoncés sont en opposition l'un par rapport à l'
    autre, ils sont interchangeables avec la négation de l'un des deux é
    noncés.
Vous répondrez uniquement avec l'un de ces deux label, sans explications
    ."
prompt (pour chaque key du corpus) = f"{corpus_data[key]['context-before
    ']}\n
                                        A: {corpus_data[key]['enonce']}\n
                                        B: {corpus_data[key]['prejacent
                                            ']}\n
                                        {corpus_data[key]['context-after
                                            ']}\n
                                        Répondez uniquement avec le label
                                            ."
```

Métriques entraînement

- Précision : mesure la proportion de prédictions positives correctes par rapport à toutes les prédictions positives.
- Rappel : mesure la proportion de véritables exemples positifs qui ont été correctement prédits par le modèle.

- F1-score : la moyenne pondérée de la précision et du rappel, ce qui fournit un équilibre entre les deux métriques et qui est particulièrement utile lorsque les classes sont déséquilibrées.
- AUC : mesure la capacité du modèle à distinguer entre les classes, particulièrement dans des contextes où les classes sont déséquilibrées.

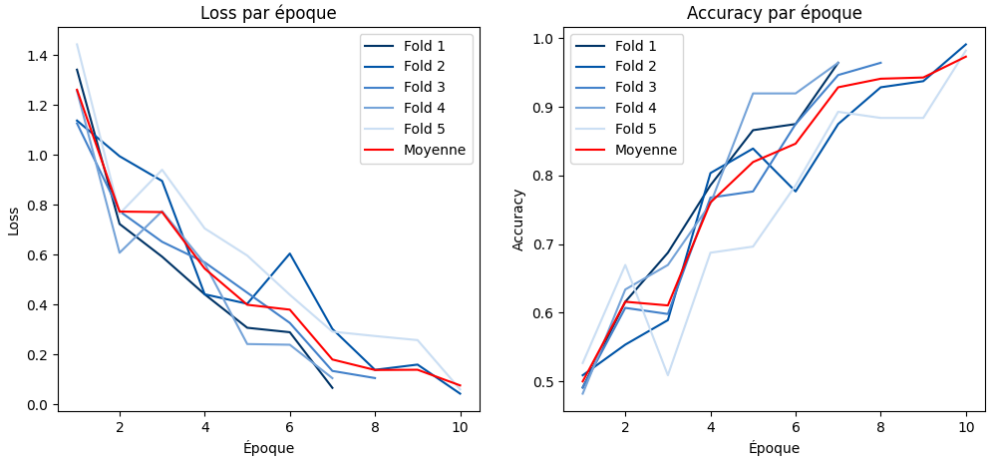
Listing 7.2 : Résultats des métriques des entraînements pour "presque" et "à peine"

```
"presque": [
    {
        "fold": 1, // meilleur fold
        "accuracy": 0.7041095890410959,
        "precision": 0.6986301369863014,
        "recall": 0.7041095890410959,
        "f1_score": 0.7068493150684931,
        "auc": 0.7013698630128571,
    },
    {
        "fold": "moyenne",
        "accuracy": 0.5659863945578232,
        "precision": 0.6204081632653061,
        "recall": 0.5795918367346939,
        "f1_score": 0.5115646258503401,
        "auc": 0.6081683265530619
    }
]

"a_peine": [
    {
        "fold": 2, // meilleur fold
        "accuracy": 0.7894736842105263,
        "precision": 0.7944444444444444,
        "recall": 0.8011363636363636,
        "f1_score": 0.788888888888889,
        "auc": 0.8011363636363636,
    },
    {
        "fold": "moyenne",
        "accuracy": 0.6269005847953216,
        "precision": 0.6712301587301588,
        "recall": 0.6798701298701298,
        "f1_score": 0.6248763955342904,
        "auc": 0.6755501443001443,
    }
]
```

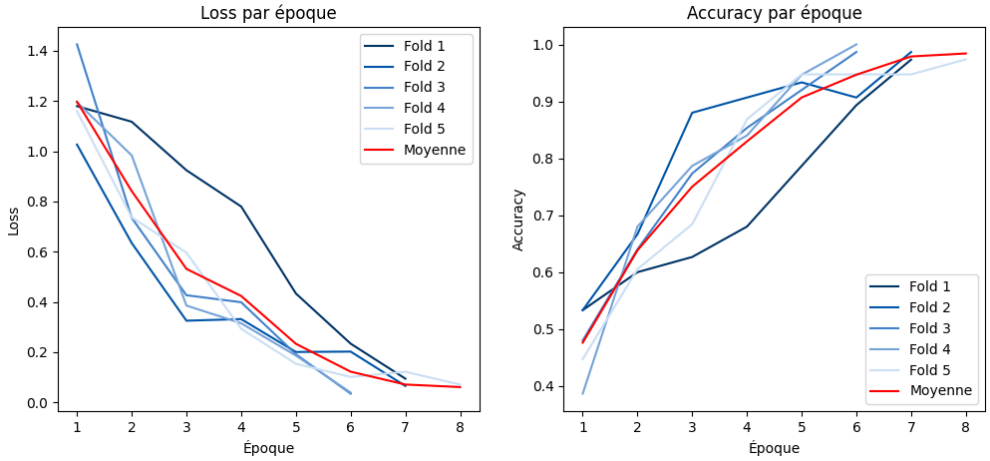
Listing 7.3 : Résultats des métriques des entraînements après l'ajout des catégories grammaticales

```
"presque_spacy": [
    {
        "fold": 3,
        "accuracy": 0.7635869565217391,
        "precision": 0.8478260869565217,
```

```
        "recall": 0.8016304347826086,
        "f1_score": 0.7521521739130435,
        "auc": 0.8247282608766666,
    },
    {
        "fold": "moyenne",
        "accuracy": 0.6521739130434783,
        "precision": 0.6793478260869565,
        "recall": 0.6657608695652174,
        "f1_score": 0.6467391304347826,
        "auc": 0.6725543478849462,
    },
]

"a_peine_spacy": [
    {
        "fold": 2,
        "accuracy": 0.631578947368421,
        "precision": 0.6277777777777778,
        "recall": 0.6306818181818181,
        "f1_score": 0.6274509803921569,
        "auc": 0.6306818181818182,
    },
    {
        "fold": "moyenne",
        "accuracy": 0.5748538011695906,
        "precision": 0.638896528308293,
        "recall": 0.6218506493506494,
        "f1_score": 0.556731908957915,
        "auc": 0.6303735888215488,
    }
]
```



(a) Entraînement pour "presque"



(b) Entraînement pour "à peine"

FIGURE 7.1 : Courbes de loss et d'accuracy durant l'entraînement des modèles

Résultats des distances des embeddings

Listing 7.4 : Données des distances cosinus entre les embeddings pour "presque"/"à peine" et "coorientation"/"opposition"

```
Traitement de l'operateur a_peine
Nombre de paires :
  - Co-orientation : 51
  - Opposition : 51
Longueur de phrase moyenne:
  - Co-orientation : 75.843
  - Opposition : 77.314
Distance cosinus moyenne :
  - Co-orientation: 0.012
  - Opposition: 0.014


Traitement de l'operateur presque
Nombre de paires :
  - Co-orientation : 71
  - Opposition : 71
Longueur de phrase moyenne:
  - Co-orientation : 77.366
  - Opposition : 72.549
Distance cosinus moyenne
  - Co-orientation: 0.006,
  - Opposition: 0.008
```