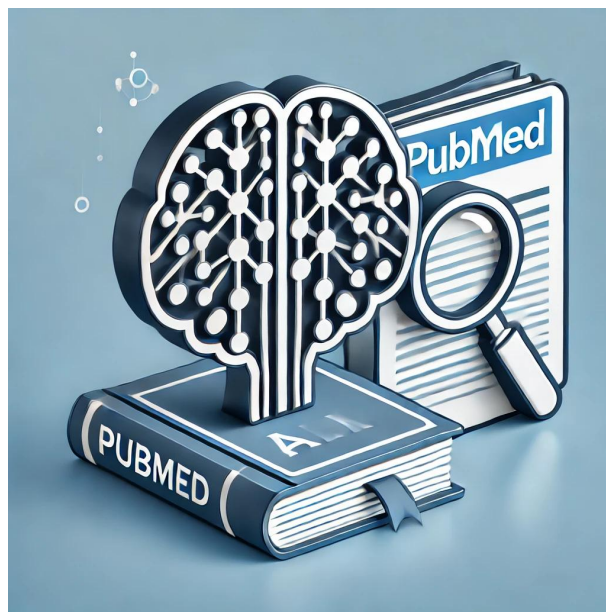


Classification intelligente de données PubMed



Projet de semestre présenté par

Ivan PAVLOVICH

**Informatique et systèmes de communication avec orientation
Informatique logicielle**

mars, 2025

Professeur-e HES responsable

Florent GLÜCK

Mandant (si existant)

Image générée par ChatGPT

TABLE DES MATIÈRES

Remerciements	vi
Résumé	vii
Liste de acronymes	viii
Liste des illustrations	ix
Liste des tableaux	x
Introduction	1
1 Chapitre 1 : Source de données PubMed	3
1.1 Contenu des articles	3
a MeSH Terms	4
1.2 Recherche et fonctionnement de l'API	4
a Recherche d'articles	5
b Automatic Term Mapping	5
c Restrictions et limitations	6
1.3 Études statistiques	6
a Données reçues	7
b Quantité de publications	7
2 Chapitre 2 : Modèles de classification	9
2.1 Transformers	9
a Tokens	10
2.2 N-Shot Learning	10
a Zero-Shot Learning	10
b Bart-Large-MNLI	11
c One-Shot Learning	11
d Few-Shot Learning	11
2.3 Large Language Models (LLM)	12
a Modèles hébergés	12
b Modèles locaux	13
2.4 Fine-tuning	13
2.5 Outils	14
a Hugging Face	14
b Ollama	15
3 Chapitre 3 : Évaluation des modèles de classification	16
3.1 Critères de validation et méthodologie	16
a Catégorie selon les longueurs de texte	16
b Création du dataset de test	17
3.2 Mesure d'évaluation	17
a Matrice de confusion	17

b	Rappel	19
c	Spécificité	19
d	Précision	20
e	Error rate	20
f	F1 Score	20
3.3	Implémentation	21
a	Modification du calcule des résultats	21
4	Chapitre 4 : Résultats et discussion	22
4.1	Résultats globaux	22
a	Classificateurs Zero-Shot	22
b	Large Language Models	23
c	Fine-tuned model	24
4.2	Résultats pour textes courts	25
a	Classificateurs Zero-Shot	25
b	Large Language Models	26
c	Fine-tuned model	27
4.3	Résultats pour textes moyens	28
a	Classificateurs Zero-Shot	28
b	Large Language Models	29
c	Fine-tuned model	30
4.4	Résultats pour textes longs	31
a	Classificateurs Zero-Shot	31
b	Large Language Models	32
c	Fine-tuned model	33
4.5	Résultats pour texte très long	34
a	Classificateurs Zero-Shot	34
b	Large Language Models	35
c	Fine-tuned model	36
4.6	Résultats des temps de classification	37
4.7	Discussion	38
	Conclusion	40
	Références documentaires	42

REMERCIEMENTS

Je remercie Monsieur Florent Glück pour avoir suivi ce projet et m'avoir aidé quand j'en avais besoin.

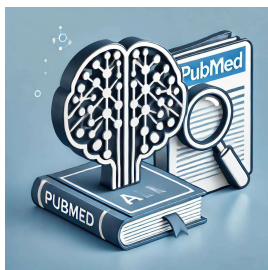
Je remercie Lisa Jaccard pour avoir aidé avec la vérification de l'orthographe.

Je remercie Antoine Giles pour m'avoir fourni une infrastructure pour mes tests.

Je remercie Quantin Leblanc pour m'avoir aidé avec l'infrastructure Phenix de l'HEPIA.

RÉSUMÉ

Ce projet de semestre est réalisé dans le cadre du projet de recherche "Non-Communicable Diseases Platform" qui est fait en collaboration avec l'Université de Genève, les Hôpitaux universitaires de Genève (HUG), le Pérou et le Kirghizistan. Le but final est de réaliser une plateforme web/mobile pour la dissémination de contenu dans le domaine des maladies non transmissibles. Ce projet de semestre s'occupe de l'étude des potentielles solutions pour la classification intelligente de textes biomédicaux. Par la suite, ce travail fera partie du service qui classifera et aggrègera, de manière intelligente, les données de santé en provenance de différentes sources. Lors de la réalisation, nous avons commencé par analyser la source PubMed. C'est une ressource gratuite qui aide à la recherche et facilite la récupération de littérature biomédicale. Après avoir trouvé une façon de récupérer les articles de PubMed, nous nous sommes penchés sur les potentielles solutions de classifications intelligentes. Nous avons fini par retenir quatre modèles de classification Zero-Shot, cinq grands modèles linguistiques (LLM) et nous avons aussi affiné un modèle Zero-Shot. Pour les tests, nous avons choisi plusieurs métriques qui représentent la performance et l'efficacité des modèles. Grâce à PubMed, nous avons pu créer un jeu de données labélisé pour nos tests, ce qui nous a donné un exemple d'application réel. Pour finir, nous avons testé tous les modèles retenus sur le jeu de données et nous avons pu constater que sur les dix modèles quatre étaient utilisables pour la suite du projet selon les besoins, désirs et capacités des collaborateurs.



Candidat-e :

IVAN PAVLOVICH

Filière d'études : ISC

Professeur-e(s) responsable(s) :

FLORENT GLÜCK

En collaboration avec : Université de Genève et HUG

Travail de bachelor soumis à une convention de stage en entreprise : non

Travail soumis à un contrat de confidentialité : non

LISTE DE ACRONYMES

API Application Programming Interface. 4, 6, 12

ATM Automatic Term Mapping. 5

E-utilities Entrez Programming Utilities. 4

IA Intelligence Artificielle. 12

LLM Large Language Model. 12, 23, 38, 39, 40

MeSH Medical Subject Headings. 3, 4, 5, 6, 7, 17, 18

NCBI National Center for Biotechnology Information. 4

NLM National Library of Medicine. 4

NLP Natural Language Processing. 9

RNN Recurrent Neural Network. 9, 10

XML Extensible Markup Language. 4

LISTE DES ILLUSTRATIONS

1.1	Logo Pubmed	3
2.1	Exemple de tokenisation	10
2.2	Logo Hugging Face	14
2.3	Logo Ollama	15
3.2	Formule du rappel	19
3.3	Formule de la spécificité	19
3.4	Formule de la précision	20
3.5	Formule du taux d'erreur	20
3.6	Formule du Score F1	21
4.1	Résultats Zero-Shot sur toutes les données	23
4.2	Résultats LLM sur toutes les données	24
4.3	Résultats pour le modèle affiné sur toutes les données	25
4.4	Résultats Zero-Shot sur les textes courts	26
4.5	Résultats LLM sur les textes courts	27
4.6	Résultats pour le modèle affiné sur les textes courts	28
4.7	Résultats Zero-Shot sur les textes moyens	29
4.8	Résultats LLM sur les textes moyens	30
4.9	Résultats pour le modèle affiné sur les textes moyens	31
4.10	Résultats Zero-Shot sur les textes longs	32
4.11	Résultats LLM sur les textes longs	33
4.12	Résultats pour le modèle affiné sur les textes longs	34
4.13	Résultats Zero-Shot sur les textes très longs	35
4.14	Résultats LLM sur les textes très longs	36
4.15	Résultats pour le modèle affiné sur les textes très longs	37

Références des URL

- URL01 <https://pubmed.ncbi.nlm.nih.gov/>
- URL02 <https://www.ibm.com/fr-fr/topics/confusion-matrix>
- URL03 <https://platform.openai.com/tokenizer>
- URL04 <https://huggingface.co/>
- URL05 <https://ollama.com/>
- URL06 <https://encord.com/blog/classification-metrics-accuracy-precision-recall/>
- URL07 <https://datajello.com/advanced-classification-metrics-precision-recall-and-more/>
- URL08 <https://encord.com/glossary/f1-score-definition/>

LISTE DES TABLEAUX

1.1	Moyenne des publications 1	7
1.2	Moyenne des publications 2	8
4.1	Temps de prédiction de classes	37
4.2	Temps de récupération des résultats	38

INTRODUCTION

Dans le cadre de la formation pour le Bachelor en Informatique et Systèmes de Communication avec l'orientation en Informatique logicielle de la Haute École du Paysage, d'Ingénierie et d'Architecture de Genève (HEPIA), nous avons dû réaliser un projet de recherche sur un sujet choisi par nos soins. Ce projet est réalisé dans le but de nous préparer pour le projet de Bachelor et il se déroule sur 5 mois à hauteur de 1 jour de travail par semaine de novembre à mars. Le projet choisi est réalisé dans le cadre du projet de recherche "Non-Communicable Diseases Platform" en collaboration avec l'Université de Genève, les hôpitaux universitaires de Genève (HUG), le Pérou et le Kirghizistan. Le but final de celui-ci est de réaliser un service qui classifie et agrège de manière intelligente des données biomédicales provenant de nombreuses sources. Pour le projet de semestre, seulement la source PubMed a été demandée d'être analysée, le point clé a été d'étudier et de vérifier les différentes façons de classifier les données de manière intelligente. Ce projet est encadré par Monsieur Florent Glück.

Au cours de la réalisation du projet, il y a eu trois axes principaux. Le premier a été d'analyser la source PubMed. Cela m'a permis par la suite d'avoir accès à une grande quantité de données potentiellement utilisables lors de la classification. Le deuxième axe a été d'explorer les différentes intelligences artificielles capables de faire de la classification de texte et de sélectionner les modèles potentiellement intéressants. Pour finir, nous avons réalisé un système de tests pour les modèles sélectionnés pour pouvoir les comparer. Les différents scripts réalisés et les données récupérées lors de ce travail peuvent être retrouvées dans le Git : https://gitedu.hesge.ch/flg_bachelors/ts/2024/ncd-data-aggregation-and-classification

La structure de ce rapport suit les trois axes de la réalisation du projet. Le premier chapitre présente la source PubMed avec toutes ces particularités et toutes les limitations rencontrée au cours de son analyse et utilisation. Le deuxième chapitre parle des recherches effectuées sur les différents type de modèles d'intelligences artificielles capables de faire de la classification de textes et des modèles retenus pour les tests. Le troisième chapitre est consacré à la réalisation des différents aspects du système de tests. Le quatrième chapitre montre les résultats obtenus et nous permet de débattre sur leur signification. Pour finir ce rapport, nous allons faire une conclusion qui repasse sur tous les points importants de chaque chapitre et qui offre de nouvelles

perspectives pour de futurs travaux.

CHAPITRE 1 : SOURCE DE DONNÉES PUBMED

Dans le cadre du projet, il nous a été demandé d'investiguer et d'utiliser la source de données PubMed, une ressource gratuite qui aide à la recherche et facilite la récupération de littérature biomédicale. Conçue dans le but d'améliorer la santé globale et celle des individus, PubMed met à disposition plus de 38 millions de citations[1] et abstracts d'articles.



ILLUSTRATION 1.1 – Logo de Pubmed, Source : ref. URL01

PubMed répertorie les articles de plusieurs sources, mais il y en a deux qui constituent la plus grande partie des articles. La source majoritaire est MEDLINE, qui représente environ 80% des articles présents sur la plateforme. Elle est constituée principalement de citations et d'articles qui ont été spécifiquement sélectionnés pour elle. La deuxième plus grande source d'articles est PubMed Central (PMC). Elle est constituée d'articles entiers qui sont accessibles gratuitement par les utilisateurs.

1.1. CONTENU DES ARTICLES

Les articles répertoriés dans PubMed ne sont pas forcément tous accessibles gratuitement, car le but de PubMed est de présenter une idée du contenu grâce à un titre, un abstract et beaucoup d'autres métadonnées. Par la suite, si l'utilisateur est intéressé, il peut aller visiter le site où l'article a été publié pour pouvoir en savoir plus. Le problème est que pas tous les sites les mettent à disposition gratuitement.

Une autre particularité des données trouvées dans les articles répertoriés sur PubMed, est qu'elles peuvent varier d'un article à un autre. Tout dépend de la provenance de ces articles et des données mises à disposition de PubMed par ses sources. Par exemple, les articles provenant de MEDLINE ont en principe tous une liste de mots-clés qui s'appelle les **Medical Subject Headings (MeSH)**, que les autres sources n'ont pas.

a. MeSH Terms

Les termes **MeSH** sont une particularité des articles provenant de MEDLINE. Ce sont des mots-clés standardisés présents dans la base de données **MeSH** qui servent à l'indexation des articles. Cette base de données est un thésaurus de vocabulaire développé par la **National Library of Medicine (NLM)**. Ils fournissent des informations sur le contenu d'un article.

En médecine, les termes utilisés sont constamment en train de changer et cette évolution est prise en compte dans la littérature scientifique grâce aux vedettes **MeSH**. La **NLM** met régulièrement à jour ces termes pour s'assurer qu'ils reflètent les avancées scientifiques et les nouveaux concepts médicaux. Ainsi, la terminologie médicale dans les articles et les recherches s'adapte aux évolutions du domaine.

Ces termes **MeSH** sont assignés par des indexeurs qui ont été entraînés dans ce but. L'indexation est basée sur le contenu des articles et le vocabulaire **MeSH**. Cette particularité nous a beaucoup aidé dans la suite de notre travail, car cela nous a permis d'obtenir un grand nombre d'articles déjà classifiés par des mots-clés standardisés. Cela a été surtout utile dans l'évaluation de l'efficacité des modèles qui seront potentiellement utilisés par la suite.

1.2. RECHERCHE ET FONCTIONNEMENT DE L'API

Pour accéder aux articles de PubMed, il y a deux moyens. Le premier est de passer par l'interface graphique sur leur site web. Le deuxième, celui qui nous intéresse, est de passer par une **Application Programming Interface (API)** les **Entrez Programming Utilities (E-utilities)**. Cette API a été créée et est mise à disposition par le **National Center for Biotechnology Information (NCBI)**. C'est une suite de 8 programmes côté serveur qui nous donne une interface stable dans leur système des Entrez query et base de données.

Les 8 programmes mis à disposition sont : EInfo, ESearch, EPost, ESummary, EFetch, ELink, ESpell et ECitMatch. Sur les 8 cités précédemment, nous en utilisons seulement 2 :

1. ESearch : il permet de faire une recherche dans une base de données de notre choix, en l'occurrence PubMed, et de recevoir une liste d'indexes des articles recherchés en retour.[2]
2. EFetch : il permet de récupérer les contenus des articles sous le format **Extensible Markup Language (XML)** à partir d'une liste d'indexes.[2]

a. Recherche d'articles

Lorsque nous faisons une recherche dans la base de données PubMed il faut tout d'abord créer un "Search term", donc une séquence de recherche qui va définir à quel point la recherche sera précise. Pour ce faire, il y a deux principes clés : les champs ("fields") et les opérateurs binaires.

Les champs sont des parties spécifiques d'une base de données où sont stockés différents types d'informations sur un article. Ils facilitent l'organisation et la récupération des informations, car ils permettent de cibler les recherches sur des sections spécifiques des métadonnées d'un article telles que le titre, l'auteur, la date de publication, les termes **MeSH**, etc. Si aucune balise n'est spécifiée, PubMed effectue la recherche dans tous les champs. Cependant, il applique également la fonction de **Automatic Term Mapping (ATM)** pour associer les termes recherchés aux termes **MeSH**, aux noms de revues, aux noms d'auteurs et à d'autres champs indexés, ce qui permet d'élargir intelligemment la recherche.

Les opérateurs booléens sont utilisés dans le but de rendre la requête plus précise ou, inversement, beaucoup plus large. Il est possible de structurer cette requête à l'aide de parenthèses qui peuvent être placées à n'importe quel endroit.

Lorsque nous avons fini de construire notre recherche, nous devons l'encoder au format URL[2]. Par exemple :

```
research.fcgi?db=pubmed&term="neoplasms"[MeSH+Terms]+AND+
(+ "marketing"[MeSH+Terms]+OR+"health"[MeSH+Terms]+)
```

Cela signifie que le système va rechercher les articles qui ont une combinaison des termes **MeSH** Neoplasms et Marketing ou Neoplasms et Health.

b. Automatic Term Mapping

L'**ATM** est automatiquement utilisé par le système de recherche lorsque le terme entré par l'utilisateur n'est pas assez précis, par exemple lorsque le champ de recherche n'est pas précisé.

L'**ATM** va rechercher dans ses "Translation Tables" qui vont lui permettre d'associer le terme recherché à d'autres. Il existe plusieurs façons de faire l'association[3], mais l'une des principales est de l'associer avec un terme **MeSH** spécifique. En résumé, la base de données recherche s'il existe un terme **MeSH** officiel associé au mot-clé saisi. Si une correspondance est

trouvée, le terme **MeSH** sera ajouté à la recherche en plus du terme saisi.

c. Restrictions et limitations

Lors de notre utilisation de cette **API**, nous avons été confrontés à plusieurs restrictions et limitations. La première est la limite d'articles récupérés en une fois avec le programme EFetch. Il ne permet pas de récupérer les données de plus de 10 000 articles en une fois, donc si le ESearch nous retourne plus de 10 000 articles pour notre requête, nous serons capables de récupérer les données uniquement des 10 000 premiers. Pour contourner ce problème, nous avons dû récupérer les articles en plusieurs requêtes consécutives, en réduisant la période sur laquelle nous faisons notre recherche. La deuxième limitation est que l'**API** ne permet pas de faire plus de 3 requêtes par seconde. Nous avons pu augmenter cette limite à 10 requêtes par seconde en ajoutant une clé de l'**API** dans celle-ci. La conséquence de cette limitation est que, combinée à la première, le temps de récupération des articles devenait trop long, ce qui ralentissait les différents tests que nous voulions réaliser sur l'**API**. La dernière restriction à laquelle nous avons été confrontés est celle de l'URL trop long. Lorsque nous faisons une recherche très précise avec beaucoup d'opérateurs booléens et beaucoup de termes **MeSH**, la recherche était annulée et retournée avec le code d'erreur 414 qui représente : "Le code de statut de réponse HTTP 414 URI Too Long indique que l'URI demandée par le client est plus longue que ce que le serveur est disposé à interpréter." [4]. Une nouvelle fois, la solution à ce problème était de séparer la requête en plusieurs parties, ce qui avait pour conséquence de rendre notre exécution encore plus longue.

Une solution que nous avons trouvée pour contourner ces limitations était de faire une sauvegarde locale d'articles sur 3 ans. De cette façon, nous n'avions plus besoin de faire des requêtes sur l'**API**, mais uniquement lire un fichier JSON.

1.3. ÉTUDES STATISTIQUES

Dans la suite du projet, il devrait y avoir un système qui s'occupe de la récupération des articles d'une façon régulière. Ces articles seront par la suite présentés aux utilisateurs de la plateforme "Non-Communicable Diseases". Une étude sur la quantité moyenne d'articles postés sur PubMed a donc dû être réalisée. Le but était de minimiser cette moyenne le plus possible afin d'obtenir une quantité possible à présenter sur une interface web.

a. Données reçues

Au début du projet de recherche, un document nous a été transmis. Ce dernier répertoriait les différents aspects que nous devions pouvoir retrouver dans les textes, afin que sur l'interface web, une recherche par préférence des utilisateurs puisse être réalisée. Nous nous sommes principalement concentrés sur deux aspects :

1. Les maladies non-transmissibles
2. Les mots-clés qui ont été choisis par les clients

Nous avons dû traduire manuellement toutes les maladies et mots-clés qui nous ont été fournis, en termes **MeSH** pour pouvoir récupérer avec certitude les articles qui parlent des termes recherchés. Malheureusement, pas tous les mots-clés fournis par les clients ont pu être traduits en termes **MeSH** à cause, principalement, d'un manque de connaissances personnelles dans le domaine médical.

b. Quantité de publications

Lors de notre étude, nous avons commencé par regarder le nombre d'articles publiés en utilisant seulement les maladies non transmissibles. Nous nous sommes malheureusement rendus compte que le nombre d'articles publiés par mois était beaucoup trop élevé, comme le montre le tableau suivant.

	jours	semaine	mois
Moyenne de publication	143	997	4337

TABLEAU 1.1 – Moyenne des publications sur PubMed pour les maladies non-transmissible / réalisé par Pavlovich Ivan.

Nous avons dû trouver une solution alternative, donc nous avons décidé de rendre la recherche encore plus précise en ajoutant les mots-clés aux maladies. Heureusement, cela nous a permis d'obtenir un nombre d'articles gérables pour l'interface web et nous n'avons pas eu besoin d'analyser les articles récupérés pour potentiellement en enlever encore plus. Le tableau suivant nous montre le nombre moyen d'articles publié sur PubMed, récupéré grâce à la recherche plus précise.

	jours	semaine	mois
Moyenne de publication	11	80	346

TABLEAU 1.2 – Moyenne des publications sur PubMed pour les maladies non-transmissibles couplées aux mots-clés / réalisé par Pavlovich Ivan.

CHAPITRE 2 : MODÈLES DE CLASSIFICATION

Ce chapitre présente l'état de l'art des modèles de classification en traitement du langage naturel, en anglais, **Natural Language Processing (NLP)**. Nous détaillons les architectures les plus performantes et les approches d'apprentissage actuelles. Depuis l'introduction en 2017 des Transformers[5], une nouvelle architecture d'apprentissage profonde (Deep Learning), ces modèles ont révolutionné le domaine du traitement du langage naturel. Les Transformers ont pris en popularité en 2020 suite à des améliorations significatives des infrastructures informatiques et suite aux capacités impressionnantes de certains modèles tels que BERT et GPT-2 qui ont attiré l'attention d'un grand nombre de chercheurs. Depuis lors, elle a commencé à être développée et utilisée par un grand nombre de personnes.

Dans le cadre du projet, un des objectifs mis en place était d'être capable de classifier les données textuelles avec l'aide d'une intelligence artificielle. Pour y arriver, les modèles utilisés doivent être capables de faire du traitement du langage naturel, ce qui veut dire qu'ils doivent être capables de comprendre, d'interpréter et de générer le langage humain de manière naturelle. Pour cette raison, la plus grande partie des modèles étudiés sont réalisés sur la base de l'architecture de Transformers.

2.1. TRANSFORMERS

Avant les Transformers, le traitement automatique du langage naturel était principalement réalisé par des modèles séquentiels tels que les Réseaux de Neurones Récurents, en anglais **Recurrent Neural Network (RNN)** et leurs variants. Ils sont conçus pour traiter des données séquentielles en gardant une mémoire des informations précédentes lorsqu'ils traitent des nouvelles entrées. Ces modèles étaient donc très adaptés dans les cas de figure où le contexte était important.[6]

L'arrivée des Transformers a résolu quelques problèmes qui limitaient les **RNN**. Premièrement, ils ont résolu le problème d'efficacité. Les **RNN** traitent les données de manière séquentielle, ce qui signifie que les éléments d'une séquence étaient traités un par un en fonction des calculs précédents. Les Transformers, grâce au mécanisme d'attention multi-tête (multi-head attention), traitent tous les éléments de la séquence simultanément, ce qui permet la parallélisation et accélère l'entraînement. Deuxièmement, grâce au même mécanisme, ils sont capables de mo-

déliser les dépendances entre les tokens à longue distance sans rencontrer les mêmes problèmes que les RNN. Pour finir, ils sont, en général plus efficaces, par exemple pour la résolution de tâches très complexes.[7]

a. Tokens

Pour traiter une séquence d'entrée, les Transformers la séparent d'abord en tokens. Les tokens représentent l'unité minimale utilisée pour le traitement de texte. Ils peuvent être des mots entiers, des sous-mots ou des caractères individuels. Cela va dépendre des méthodes de tokenisation utilisées.

Tokens	Characters
5	16

ILLUSTRATION 2.1 – Exemple de tokenisation, Source : ref. URL03

2.2. N-SHOT LEARNING

Le N-Shot Learning est un concept de l'apprentissage automatique dans lequel les modèles sont entraînés à partir d'un nombre limité d'exemples par classe. Lors de son développement, les Transformers ont joué un rôle essentiel grâce à leur connaissance générale du langage, dans le but d'interpréter et répondre à des nouvelles requêtes.

a. Zero-Shot Learning

Le principe de Zero-Shot Learning est une approche d'apprentissage automatique qui permet aux modèles sans exemple spécifique des différentes catégories lors de l'entraînement, de reconnaître et de classer les objets ou concepts. Ils utilisent les descriptions textuelles ou les attributs sémantiques, pour établir des relations entre les classes connues et inconnues.

Ce principe est mis en œuvre par les classificateurs Zero-Shot. Ils utilisent des modèles qui ont déjà été entraînés sur une vaste base de données et qui ont développé une compréhension profonde des concepts et relations entre les mots pour pouvoir classer des classes potentiellement jamais vues lors de l'entraînement. Nous avons décidé de commencer par ces modèles, car nous étions parti du principe que les collaborateurs pourraient ajouter de nouvelles classes à

n'importe quel moment, ce qui ne nécessiterait pas de réentraîner les modèles pour les nouvelles classes.

b. Bart-Large-MNLI

Lors de nos recherches, un modèle en particulier était toujours utilisé dans les exemples d'implémentation de classificateur Zero-Shot : le modèle Bart-Large-MNLI[8] développé par des chercheurs de chez Facebook AI. Il est considéré comme le pilier de cette catégorie grâce à son efficacité pour la classification précise sans nécessité d'entraînement spécifique. Il est basé sur une version plus puissante de BART (Bidirectional and Auto-Regressive Transformers). C'est un modèle de génération de texte entraîné comme auto-encodeur débruitant, ce qui signifie qu'il apprend à reconstruire un texte à partir d'une version dégradée. La version plus puissante de BART est BART-Large qui possède simplement plus de couches dans la partie encodeur et décodeur. Pour améliorer les capacités d'analyse et d'inférence textuelle, le modèle a été entraîné sur le dataset MNLI (Multi-Genre Natural Language Inference). Grâce à cet entraînement, BART-Large-MNLI[8] peut être utilisé pour de la classification Zero-Shot de textes.

C'est pour ces raisons que nous avons décidé de retenir ce modèle et de tester ses performances pour classer les articles récupérés depuis PubMed, ce qui nous montrerait aussi sa capacité générale pour classer de la littérature biomédicale.

c. One-Shot Learning

L'approche d'apprentissage automatique One-Shot Learning permet aux modèles de reconnaître et classer de nouvelles catégories d'objets ou de données à partir d'un seul exemple par catégorie. La généralisation depuis un seul exemple est souvent réalisée en comparant les similitudes entre les nouvelles données et les exemples connus. C'est une méthode utilisée lorsque la collecte de nombreuses données est difficile, impossible ou trop coûteuse.

d. Few-Shot Learning

Le Few-Shot Learning est une approche de l'apprentissage automatique qui rend les modèles capables de généraliser à de nouvelles tâches ou classes sur la base de seulement quelques exemples d'entraînement par classe. Cette méthode est utilisée dans les cas de figure où la col-

lecte de nombreuses données pour l'entraînement est difficile ou coûteuse.

2.3. LARGE LANGUAGE MODELS (LLM)

Un grand modèle linguistique, en anglais, **Large Language Model (LLM)** est un modèle d'**Intelligence Artificielle (IA)** capable, principalement, de comprendre, reconnaître, générer et prédire du texte en langage naturel. Ces modèles sont entraînés sur un très grand nombre de données textuelles, dont une grande partie est entraînée sur des données directement recueillies sur Internet, ce qui leur permet d'acquérir une compréhension approfondie des structures linguistiques et du contexte sémantique.[9]

Les **LLM** modernes reposent principalement sur l'architecture des Transformers. Elle leur permet de traiter efficacement les séquences de textes en retenant les relations entre les mots, de comprendre les nuances du langage et surtout de traiter de grandes quantités de données.

Ces modèles utilisent le principe de prompts. C'est une instruction textuelle qui décrit et guide le modèle dans la tâche à accomplir. Dans le contexte de classification zero-shot, le prompt joue un rôle crucial car il permet au modèle en s'appuyant sur sa compréhension du langage naturel de générer des réponses précises sans entraînement préalable. En fonction des instructions fournies dans le prompt, le modèle peut effectuer de la classification.[10]

a. Modèles hébergés

Un modèle hébergé désigne un modèle déployé sur une infrastructure informatique telle qu'un serveur ou un service cloud pour qu'il soit accessible en temps réel depuis n'importe quel endroit. Cela permet aux utilisateurs d'interagir avec le modèle au travers des interfaces logicielles telles que des **API**, sans avoir à l'exécuter localement par leurs propres moyens. Cette approche est très utilisée par les entreprises leaders du marché d'**IA** et d'apprentissage machine pour pouvoir monétiser l'utilisation de leurs modèles. Ces modèles ont une taille de plus en plus grande et demandent de plus en plus de ressources pour pouvoir fonctionner, ce qui a pour conséquence qu'une grande partie de la population n'aura jamais les moyens d'exécuter ces modèles. Il existe un grand nombre de modèles mis à disposition de cette façon de la part d'entreprises tels que : OpenAI, Anthropic, DeepSeek AI, Mistral AI, Cohere, Google DeepMind, Meta AI, pour ne citer que les plus connues.

Toutes ces entreprises utilisent la même stratégie de monétisation qui consiste à faire payer l'utilisateur pour tous tokens donnés en entrée au modèle et pour tous tokens générés par le

modèle en sortie. Les prix sont, la plupart du temps, donnés pour le million de tokens et varient beaucoup selon les différents modèles.

Pour le projet, nous avons voulu explorer ces modèles et tester leurs performances. Pour cela, nous avons dû utiliser les versions gratuites des modèles mis à disposition par les entreprises. Le problème est que l'utilisation des versions gratuites vient avec des limitations qui ont des conséquences négatives sur les performances de ces modèles. Aussi, les entreprises ne mettent pas à disposition leurs meilleurs modèles gratuitement. Malheureusement, seulement deux des entreprises citées précédemment donnent un accès gratuit à certains de leurs modèles :

- Google DeepMind mettent à disposition le modèle Gemini 1.5 Flash[11]
- Cohere mettent à disposition le modèle Command R+[12]

b. Modèles locaux

Un modèle local désigne un modèle d'intelligence artificielle déployé et exécuté sur un ordinateur ou un serveur interne. Le principe est de ne pas dépendre d'une connexion constante à des serveurs distants ou au cloud.

Les grands inconvénients de ces types de modèles sont :

- Les coûts de mise en place de l'infrastructure pour l'exécution de modèles sont très élevés.
- Les ressources nécessaires pour exécuter efficacement des modèles IA complexes sont élevées.
- Les modèles IA locaux peuvent entraîner une consommation énergétique conséquente.

2.4. FINE-TUNING

Le réglage fin (Fine-Tuning) est une technique d'apprentissage automatique qui consiste à ré-entraîner un modèle existant sur un ensemble de données pour l'adapter à une tâche spécifique. Le processus de fine-tuning d'un modèle se fait en quatre étapes séquentielles :

1. Choisir un modèle existant dont l'architecture est la plus adaptée pour la tâche à réaliser.
2. Récupérer un jeu de données et le traiter pour qu'il soit utilisable.
3. Configurer les paramètres de l'entraînement comme le taux d'apprentissage, le nombre d'époques et la taille des lots de données.
4. Entraîner le modèle choisi sur le jeu de données avec les paramètres choisis.

Pour ce projet, j'ai décidé d'affiner le modèle BART-Large-MNLI[8][13]. Car comme nous allons le voir dans le chapitre résultats, ce modèle était performant dans tous les cas de figure. Le jeu de données pour l'affinage a été récupéré de PubMed. Nous avons récupéré 3 ans d'articles pour nos mots-clés, puis nous les avons filtrés pour enlever les articles qui seront présents dans le jeu de données de tests, et pour finir, nous avons entraîné le modèle sur 20 époques.

Le problème principal que nous avons rencontré lors de ce processus était le temps d'entraînement, car avec le matériel qui nous a été mis à disposition, l'affinage prenait des journées entières et sous certaines conditions, il pouvait monter jusqu'à plusieurs semaines.

2.5. OUTILS

Pour pouvoir utiliser les modèles trouvés lors de nos recherches, nous avons dû trouver des outils. Pour cette raison, nous avons pensé intéressant d'explorer et d'essayer de trouver plusieurs outils qui nous permettraient de faire des implémentations de manière simple et facile à comprendre.

a. Hugging Face



ILLUSTRATION 2.2 – Logo de Hugging Face, Source : ref. URL04

Hugging Face est une entreprise franco-américaine créée dans le but de rendre l'intelligence artificielle accessible à tous.[14] Ils ont développé plusieurs outils utilisés par des millions de personnes dans le monde. Entre autres, ils ont créé une plateforme d'hébergement nommée Hugging Face Hub qui héberge des dépôts Git, des modèles de machine learning, des ensembles de données et des applications web. Ils ont aussi développé plusieurs bibliothèques qui facilitent la gestion des données et des modèles existants, le développement et l'entraînement de nouveaux modèles, telle que la bibliothèque Datasets qui permet l'accès à plus de 100 jeux de données NLP ou la bibliothèque Transformers qui permet aux utilisateurs d'utiliser et modifier des modèles d'apprentissage profond basé sur l'architecture Transformer, qui elle aussi, a joué un grand rôle dans la montée en popularité de cette architecture.

Nous avons utilisé ces outils pour trouver et utiliser des modèles de classification Zero-Shot. En plus du modèle BART-Large-MNLI[8], nous avons sélectionné trois modèles supplémentaires. Étant donné que la plateforme possède des milliers de modèles, nous ne pouvions pas tous les sélectionner, donc nous avons décidé de sélectionner les modèles en fonction de leur popularité auprès de la communauté de Hugging Face. Les trois modèles sélectionnés sont :

- DeBERTa-v3-base-mnli-fever-anli[15]
- deberta-v3-base-zeroshot-v1.1-all-33[16]
- multilingual-MiniLMv2-L6-mnli-xnli[17]

b. Ollama



ILLUSTRATION 2.3 – Logo de Ollama, Source : ref. URL05

Ollama est un outil open-source qui permet l'exécution de grands modèles linguistiques en locale, donc sur la machine de l'utilisateur. Son but est de faciliter le téléchargement, la gestion et l'utilisation de différents modèles. Il possède un catalogue de modèles très grand, mais surtout parfois des versions différentes pour des modèles qui sont potentiellement trop volumineux. Les différentes versions ont moins de paramètres et sont plus légères, ce qui les rend exécutables par des utilisateurs lambdas.

Pour ce projet, nous avons utilisé Ollama dans le but de trouver des versions légères de LLM qui n'étaient pas mis à disposition gratuitement par les entreprises. Nous avons fini par retenir trois modèles, listés ci-dessous.

- Mistral Small[18] de Mistral AI
- Llama 3.2[19] de Meta AI
- DeepSeek-V2[20] de DeepSeek AI

CHAPITRE 3 : ÉVALUATION DES MODÈLES DE CLASSIFICATION

Dans le cadre du projet, il nous a été demandé de classifier des textes médicaux provenant pour l'instant uniquement de PubMed à l'aide d'une intelligence artificielle. Dans ce but, nous avons dû créer un système de tests pour pouvoir évaluer l'efficacité des modèles sélectionnés par nos soins. Ce système de test est très important, car il faut que nous soyons sûrs des modèles à utiliser par la suite et qu'ils soient assez performants pour minimiser l'erreur, car les textes médicaux seront par la suite recommandés aux utilisateurs de la plateforme "Non-Communicable diseases" selon les mots-clés prédits par ces modèles. Pour maximiser la satisfaction des utilisateurs, nous devons obtenir des résultats de très bonne qualité.

3.1. CRITÈRES DE VALIDATION ET MÉTHODOLOGIE

Lors de l'évaluation de l'efficacité des différents modèles, nous nous sommes d'abord penché sur un aspect global. Nous avons commencé par regarder s'il n'y avait pas un modèle qui arriverait à réaliser notre tâche de manière très efficace et avec un très bon taux de réussite. Mais après avoir testé plusieurs modèles connus ou populaires, nous avons remarqué que certains avaient des résultats très similaires, mais pas forcément satisfaisants. Nous avons donc décidé de regarder s'il y avait des métriques qui influent sur les performances des différents modèles comme par exemple, s'il y avait des modèles qui étaient meilleurs que d'autres sous certaines conditions.

a. Catégorie selon les longueurs de texte

Nous avons décidé d'observer les performances des modèles sur les différentes longueurs de texte. Pour cela, nous avons commencé par regarder les longueurs moyennes de textes et, suite à cela, nous avons séparé les articles en 4 catégories :

- **SHORT** : textes courts qui sont en dessous de 300 caractères.
- **MEDIUM** : textes moyens qui sont en dessous de 600 caractères.
- **LONG** : textes longs qui sont en dessous de 900 caractères.
- **VERY LONG** : textes très longs qui sont au-dessus de 900 caractères.

En plus des tests réalisés auparavant, nous avons observé les performances de nos modèles dans toutes ces catégories de longueur de texte.

b. Création du dataset de test

Pour pouvoir réaliser tous les tests, nous avons dû créer un jeu de données adapté à notre tâche. Nous l'avons créé à partir des articles PubMed qui proviennent de la source MEDLINE. Ils ont la particularité d'être labellisés par des experts. Ces labels, qui se nomment termes **MeSH**, ont aussi l'avantage d'être standardisés. Ils ne vont donc pas varier d'un article à un autre. Nous avons trouvé manuellement les termes **MeSH** qui correspondent aux mots-clés sur lesquels nous devons classer les textes et nous avons créé un jeu de données qui comporte un nombre égal d'articles pour chaque mot-clé et un nombre égal d'articles appartenant à chaque catégorie de longueur de texte pour les mots-clés.

3.2. MESURE D'ÉVALUATION

Pour pouvoir évaluer l'efficacité et la performance de nos modèles, nous avons dû trouver des métriques pertinentes pour notre cas d'application. Les deux premières que nous avons décidées d'utiliser sont des métriques de temps :

- **Temps de classification** : le temps que le modèle prend pour classer un texte, cela comprend l'appelle à l'API ou l'appelle à la fonction de classification.
- **Temps de récupération des résultats de classification** : le temps que le programme prend pour récupérer les résultats, cela comprend le temps de classification, le temps pour traiter les résultats et les potentielles attentes nécessaires pour faire fonctionner le modèle.

Dans notre cas de figure, le temps de récupération des résultats est très important, car si le système prend plus de temps à obtenir les résultats de la classification des articles qu'il les récupère, cela sera un gros problème. Pour le reste des métriques, nous avons reçu de l'aide de la part de différents professeurs et assistants au sein de la Haute École du Paysage, d'Ingénierie et d'Architecture de Genève (HEPIA).

a. Matrice de confusion

La matrice de confusion, ou autrement appelée matrice d'erreur, permet d'évaluer les performances d'un modèle de classification. Elle donne la possibilité de comparer les valeurs prédites aux valeurs réellement présentes dans un jeu de données. Dans notre cas, elle nous permet de comparer les classes prédites par nos modèles aux classes qui sont présentes dans les articles à

travers les termes MeSH.

La matrice de confusion se présente sous la forme d'un tableau qui comporte 4 valeurs, comme nous pouvons le voir dans l'image suivante.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

ILLUSTRATION 3.1

Elle est construite de la manière suivante :

- **Vrai positif (TP)** : les classes qui ont été prédites sont présentes dans les classes attendues de l'article.
- **Faux positif (FP)** : les classes qui ont été prédites ne sont pas présentes dans les classes attendues de l'article.
- **Faux négatif (FN)** : les classes qui sont attendues pour l'article, n'ont pas été prédites par le modèle.
- **Vrai négatif (TN)** : les classes qui ne sont pas attendues pour l'article, n'ont pas été prédites par le modèle.

Cette matrice nous permet de voir les performances globales d'un modèle ou aussi les statistiques par classe. Par exemple, nous pouvons voir sur toutes nos classes, lesquels ont été le plus prédites incorrectement ou lesquels n'ont pas été prédites alors qu'elles le devaient. Cela nous indique les classes qui posent problème aux modèles et par la suite cela nous a permis d'apporter des ajustements sur notre façon de calculer nos résultats globaux.

b. Rappel

Le rappel (recall en anglais) est une mesure de performance très utilisée pour les modèles de classification. Parfois appelée taux de réussite, sensibilité ou taux de vrai positive (True Positive Rate, TPR), elle permet d'évaluer la capacité d'un modèle à prédire correctement toutes les occurrences positives d'un jeu de données. Dans notre cas, les occurrences positives sont les classes qui doivent être prédites pour un article. Elle est calculée de la façon suivante :

$$Recall = \frac{TP}{TP + FN}$$

ILLUSTRATION 3.2 – Formule pour calculer le rappel, Source : ref. URL06

La métrique calcule le taux de bonnes prédictions (TP) sur la totalité des prédictions qui doivent être faites (TP + FN). Plus le modèle arrive à identifier les cas positifs réels plus le score sera élevé.

c. Spécificité

La spécificité, aussi appelée taux de vrai négatifs (True Negative Rate, TNR) est une mesure de performance utilisée pour les modèles de classification. Elle nous donne la capacité d'un modèle à prédire les occurrences négatives d'un jeu de données. Elle est calculée de la façon suivante :

$$TNR = \frac{TN}{TN + FP}$$

ILLUSTRATION 3.3 – Formule pour calculer la spécificité, Source : ref. URL07

La métrique calcule le taux de prédiction négative correcte (TN) sur la totalité des prédictions qui ne doivent pas être faites (TN + FP). Plus le modèle n'identifie pas les cas négatifs réels plus le score sera élevé.

d. Précision

La précision est une métrique très similaire au rappel. Là où le rappel nous donne le taux de bonnes prédictions sur toutes les prédictions qui devraient être faites, la précision nous permet d'obtenir le taux de bonnes prédictions sur la totalité des prédictions faites. Elle est calculée de la manière suivante :

$$Precision = \frac{TP}{TP + FP}$$

ILLUSTRATION 3.4 – Formule pour calculer la précision, Source : ref. URL06

La métrique calcule le taux de bonne prédiction positives (TP) sur la totalité des prédictions qui doivent être positives (TP + FP). Moins le modèle fait de prédictions fausses, plus le score sera élevé.

e. Error rate

Le taux d'erreur (error rate) est une métrique en apprentissage automatique qui permet d'évaluer la performance d'un modèle de classification. Elle représente la proportion de prédictions incorrectes sur la totalité des prédictions faites. Elle est calculée de la manière suivante :

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN}$$

ILLUSTRATION 3.5 – Formule pour calculer le taux d'erreur, Source : réalisé par Pavlovich Ivan

La métrique calcule le taux de mauvaise prédictions (FP + FN) sur la totalité des prédictions faites (TP + TN + FP + FN). Plus le modèle se trompe sur les prédictions réalisées, plus le score sera élevé.

f. F1 Score

Le F1 score est métrique souvent utiliser pour évaluer la performance global des modèles de classification. Elle donne une mesure équilibrée en combinant le rappel et la précision et elle

est calculé de la façon suivante :

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

ILLUSTRATION 3.6 – Formule pour calculer le Score F1, Source : ref. URL08

La métrique est calculée en faisant une moyenne harmonique de la précision et du rappel. Un F1 score de 80% est considéré comme bon et au dessus de 85% comme très satisfaisant.

3.3. IMPLÉMENTATION

Pour l'implémentation du système de tests, nous avons réalisé un script en python. Ce script utilise les surcouches que nous avons réalisées pour chaque modèle pour utiliser les différentes méthodes de prédiction. Lors des prédictions le script sauvegarde les résultats après chaque articles pour éviter de perdre le progrès et pour recommencer à l'endroit où il s'est arrêté en cas de problème.

a. Modification du calcul des résultats

En regardant les matrices de confusion après avoir effectué les tests, nous nous sommes rendu compte que tous les modèles avaient de la peine avec l'une de nos classes. Ils avaient de la peine car cette classe est très proche de deux autres qui sont *Diabetes*, *Diabetes type 1* et *Diabetes type 2*. Très souvent quand les modèles devaient prédire *Diabetes type 1* ou *Diabetes type 2*, ils prédisaient aussi *Diabetes*, ce qui était considéré comme incorrecte par le système. Vu la proximité des termes, nous avons décidé de modifier le calcul pour que quand les modèles prédisent *Diabetes* pour *Diabetes type 1* ou *Diabetes type 2* et inversement, le système considère cela comme des réponse correctes.

CHAPITRE 4 : RÉSULTATS ET DISCUSSION

Ce chapitre présente les résultats obtenus suite à l'utilisation des différents modèles retenus lors des recherches effectuées. Tous les résultats présentés ont été réalisés sur la dernière version du jeu de données de test et sur une machine qui a un Ryzen 5800x, 32GB de RAM et une RTX 3090. Pour faire un rappel, le but des tests est d'analyser les performances de différents modèles existants ou créés par nos soins pour la classification de textes biomédicaux. Dans ce cas, tous les textes sont des articles provenant de PubMed. C'est une partie très importante, car elle permet d'avoir une image globale de tout le modèle exploré. Elle permet aussi d'avoir une idée des modèles à utiliser dans la continuation du projet et dans quels cas ces modèles seront utilisés. L'analyse des résultats nous permettra aussi de repérer les potentielles améliorations qui peuvent être apportées.

4.1. RÉSULTATS GLOBAUX

Nous allons commencer par observer les résultats globaux des différents types de modèles utilisés. Ces résultats sont réalisés sur l'entièreté du jeu de données sans prendre en compte les différentes longueurs de textes.

a. Classificateurs Zero-Shot

En observant le graphique suivant :

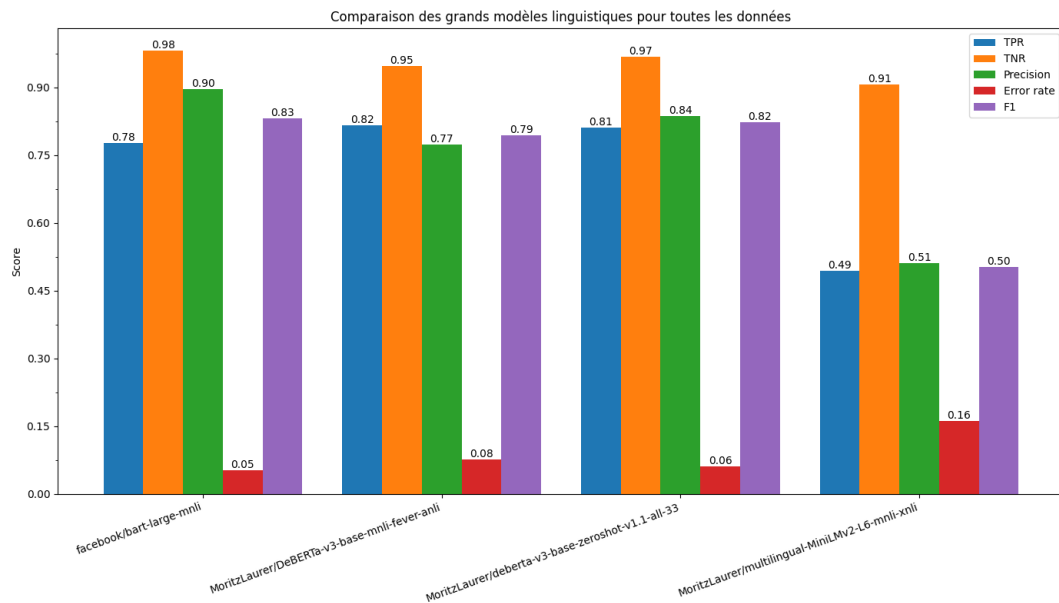


ILLUSTRATION 4.1 – Résultats pour les classificateur Zero-Shot sur toutes les données ,
Source : réalisé par Pavlovich Ivan

nous pouvons voir que le modèle BART-Large-MNLI[8] est le meilleur des quatre modèles de classification Zero-Shot, mais que les modèles DeBERTa-v3-base-mnli-fever-anli[15] et deberta-v3-base-zeroshot-v1.1-all-33[16] ne sont pas loin en performances. Même si le DeBERTa-v3-base-mnli-fever-anli[15] est le seul des trois à être en dessous du seuil de 80% pour le F1 score. En contrepartie, le modèle multilingual-MiniLMv2-L6-mnli-xnli[17] n'est vraiment pas satisfaisant. Tous ses scores sont très bas comparés aux trois autres modèles.

b. Large Language Models

Pour les LLM dans un cadre global, nous pouvons observer dans l'illustration 4.2 :

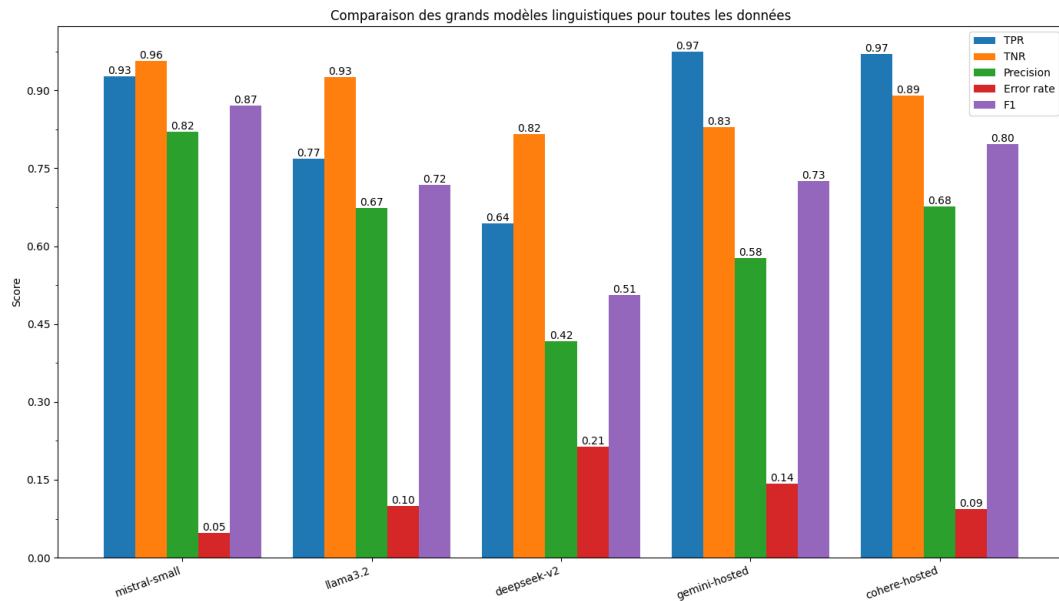


ILLUSTRATION 4.2 – Résultats pour les grands modèles linguistiques sur toutes les données ,
Source : réalisé par Pavlovich Ivan

que le modèle Mistral Small[18] a le meilleur score F1 et aucun des autres modèles ne s'en rapproche. Son score F1 dépasse le seuil de 85%, ce qui veut dire qu'il est considéré comme très satisfaisant. On peut aussi constater que les deux modèles hébergés ont un rappel (TPR) très élevé mais une précision très basse, ce qui signifie qu'ils identifient correctement une grande proportion des cas positifs réels mais qu'une grande partie des prédictions positives sont incorrectes. Par conséquent, leur score F1 est bas. Pour DeepSeek-V2[20], on voit qu'il n'est en général pas performant. On le remarque par le fait que toutes les métriques ont des scores très bas et que son taux d'erreur est très élevé. Pour Llama 3.2[19], on peut observer que ses scores sont équilibrés, mais qu'ils ne sont pas satisfaisants.

c. Fine-tuned model

Nous avons décidé de comparer le modèle affiné avec les modèles qui ont un score F1 qui dépasse 80%, donc les modèles qui sont considérés comme bons. En observant le graphique suivant :

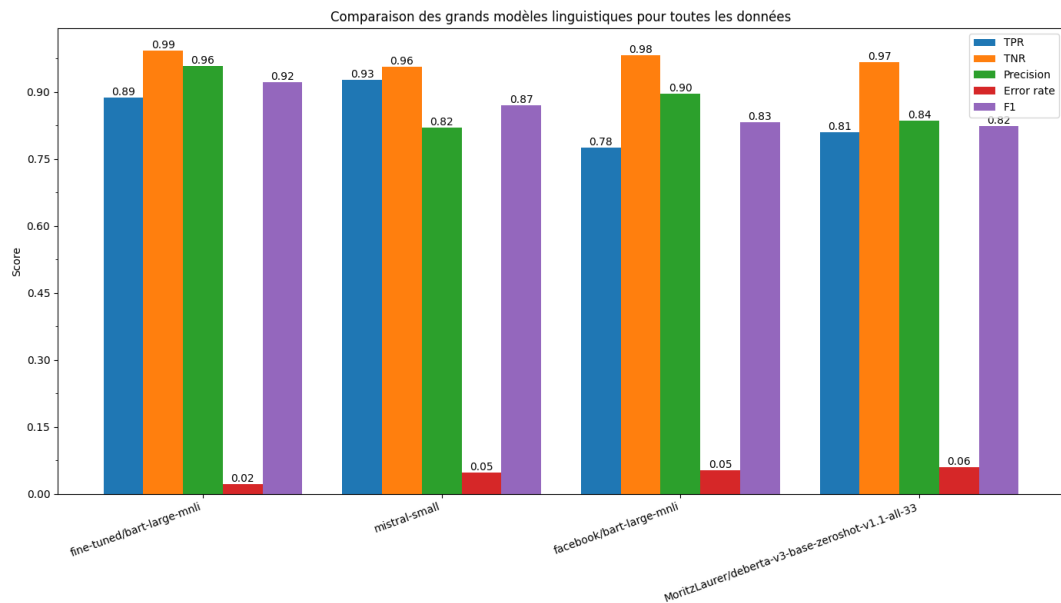


ILLUSTRATION 4.3 – Résultats pour le modèle affiné et les meilleurs modèles sur toutes les données , Source : réalisé par Pavlovich Ivan

nous pouvons voir que Mistral Small[18] est en général meilleur que les deux classificateurs Zero-Shot, mais que le modèle affiné a les meilleures performances d'entre tous. Il a un score F1 de 92%, ce qui veut dire qu'en général, le modèle arrive à identifier correctement une grande partie des cas positifs réels et qu'il fait très peu de prédictions incorrectes sur toutes les prédictions positives. Aussi, il a 9% de plus sur le score F1 que le modèle BART-Large-MNLI[8], qui est sa base.

4.2. RÉSULTATS POUR TEXTES COURTS

Après les tests de classification généraux, nous avons décidé de regarder si la longueur du texte classifié avait un impact sur les performances. Dans ce sous-chapitre, nous allons observer les résultats des tests sur les textes courts, donc qui sont en dessous de 300 caractères.

a. Classificateurs Zero-Shot

En regardant le graphique de l'illustration 4.4 :

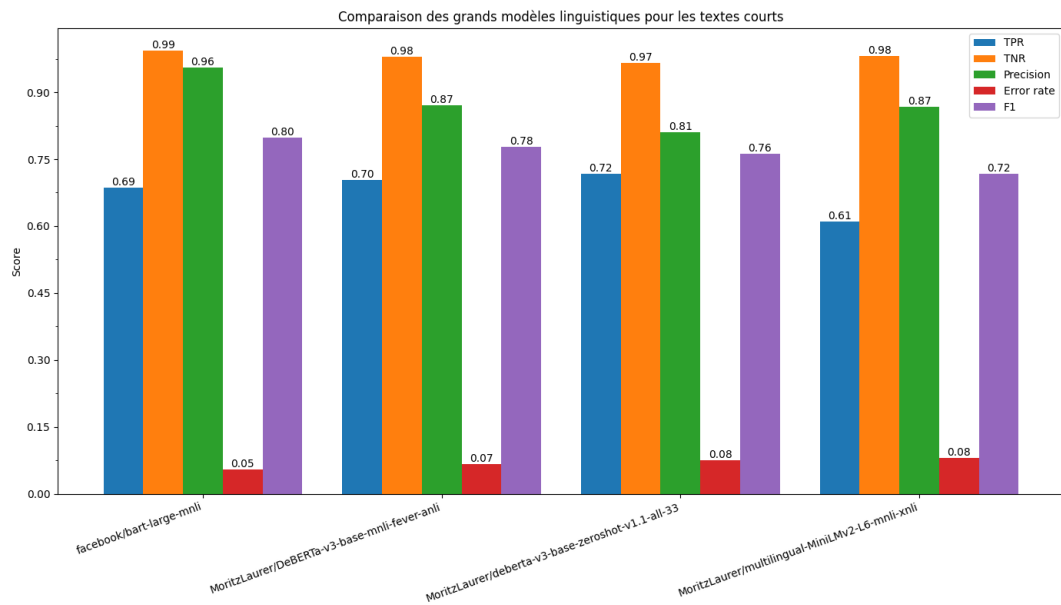


ILLUSTRATION 4.4 – Résultats pour les classificateur Zero-Shot sur les textes courts , Source : réalisé par Pavlovich Ivan

nous pouvons constater que le meilleur modèle est le BART-Large-MNLI[8]. Avec son score F1 de 80%, c'est le seul qui peut être considéré comme bon. Les trois autres modèles ont des résultats très similaires entre eux. On peut voir que leurs taux d'erreurs et leurs précisions sont presque identiques.

b. Large Language Models

En observant le graphique suivant :

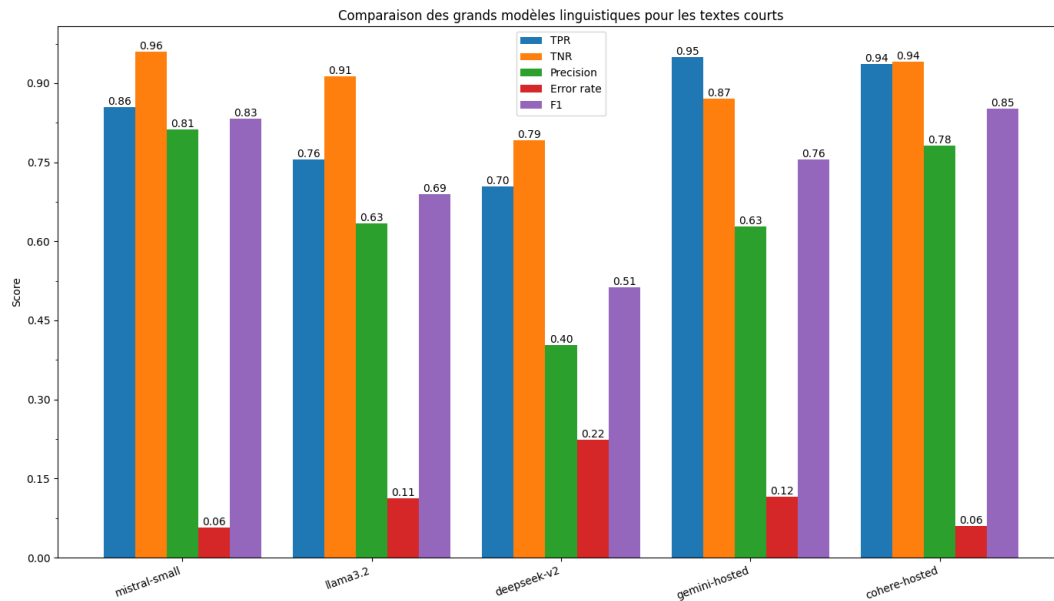


ILLUSTRATION 4.5 – Résultats pour les grands modèles linguistiques sur les textes courts ,
Source : réalisé par Pavlovich Ivan

nous observons que le meilleur des grands modèles linguistiques est Mistral Small[18]. C'est le seul modèle qui est considéré comme bon avec son score F1 de 82%. On peut néanmoins observer que les deux modèles hébergés ne sont pas satisfaisants, mais qu'ils se rapprochent du seuil de 80% sur le score F1. On peut aussi voir que le modèle Command R+[12] de Cohere est le seul à avoir 100% de rappel. Il a donc réussi à prédire toutes les occurrences réelles positives du jeu de données. Les deux modèles locaux, Llama 3.2[19] et DeepSeek-V2[20] ne sont vraiment pas performants pour les textes courts.

c. Fine-tuned model

Nous avons décidé de comparer le modèle affiné avec les modèles qui ont un score F1 qui dépasse 80% dans le cadre des textes courts. Dans l'**illustration 4.6** :

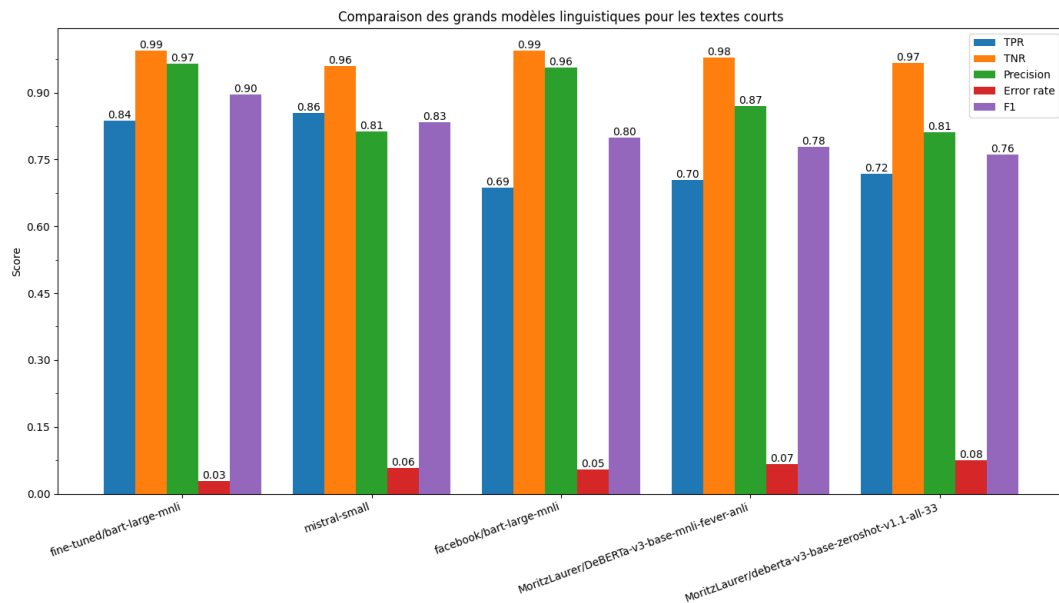


ILLUSTRATION 4.6 – Résultats pour le modèle affiné et les meilleurs modèles sur les textes courts , Source : réalisé par Pavlovich Ivan

nous pouvons constater que les modèles Mistral Small[18] et BART-Large-MNLI[8] sont très proches en performance, avec Mistral Small[18] qui est légèrement meilleur et qui arrive à mieux identifier les occurrences réelles positives et BART-Large-MNLI[8] qui fait moins de fausses prédictions positives en général. Le modèle affiné reste meilleur, avec un score F1 de 90%, ce qui le rend très satisfaisant.

4.3. RÉSULTATS POUR TEXTES MOYENS

Dans ce sous-chapitre, nous allons observer les résultats des tests sur les textes qui sont de longueur moyenne, donc qui sont au-dessus de 300 caractères et en dessous de 600.

a. Classificateurs Zero-Shot

En observant le graphique suivant :

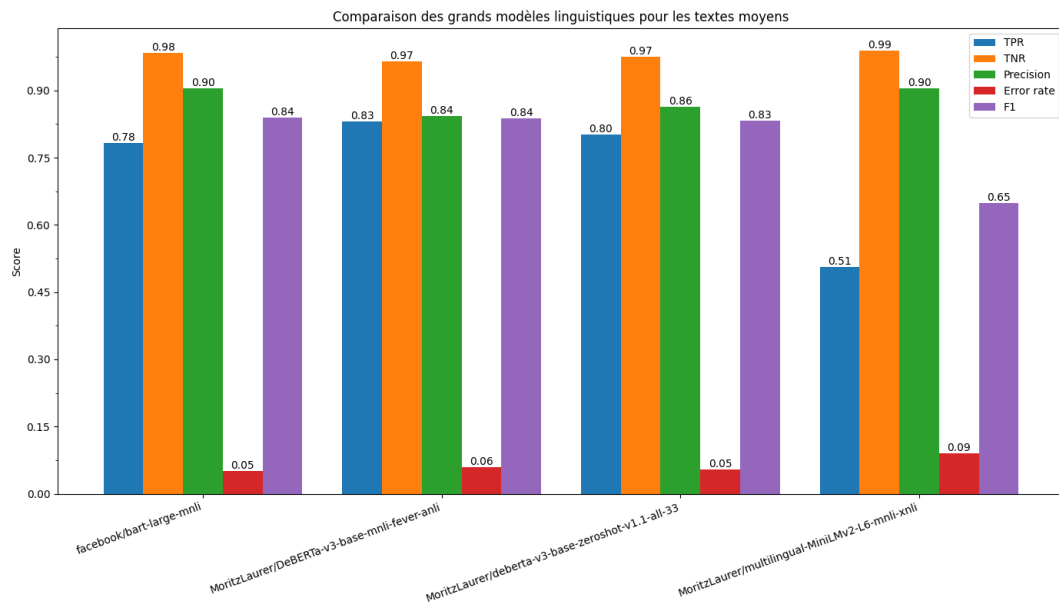


ILLUSTRATION 4.7 – Résultats pour les classificateur Zero-Shot sur les textes moyens , Source : réalisé par Pavlovich Ivan

nous pouvons voir que les trois modèles BART-Large-MNLI[8], DeBERTa-v3-base-mnli-fever-anli[15] et deberta-v3-base-zeroshot-v1.1-all-33[16] sont très similaires en performances. Les trois sont considérées comme bon car leur score F1 dépasse les 80%, mais le modèle multilingual-MiniLMv2-L6-mnli-xnli[17] n'est vraiment pas satisfaisant, il a une précision élevée et un rappel très bas, ce qui veut dire qu'il fait peu d'erreurs sur les prédictions positives mais il n'identifie pas bien les occurrences réelles positives.

b. Large Language Models

En observant le graphique suivant :

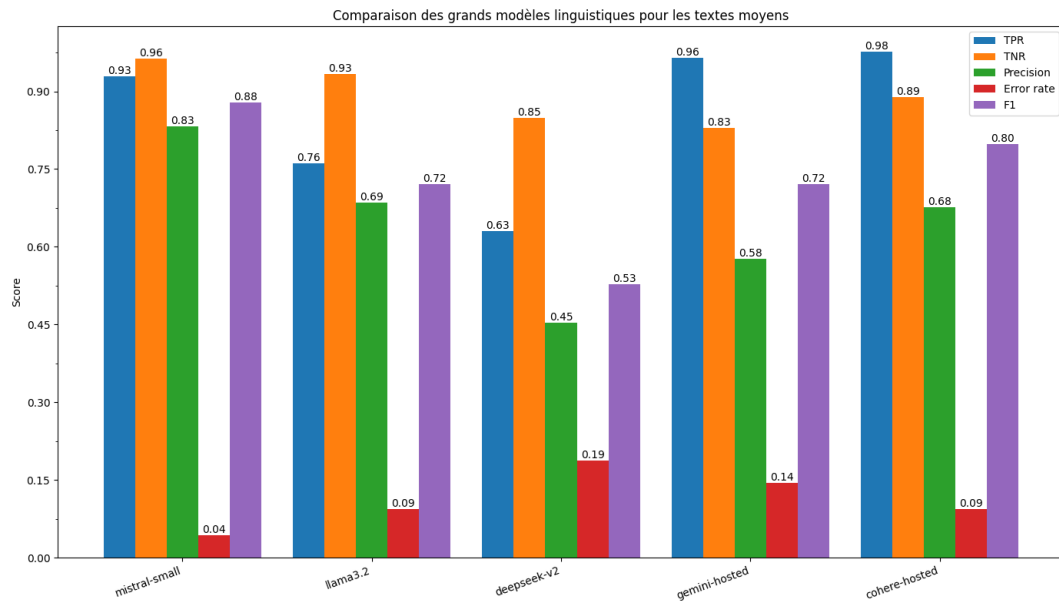


ILLUSTRATION 4.8 – Résultats pour les grands modèles linguistiques sur les textes moyens ,
Source : réalisé par Pavlovich Ivan

nous pouvons voir que le modèle le plus performant est Mistral Small[18] avec un score F1 de 88%. Les modèles Llama 3.2[19] et Gemini 1.5 Flash[11][11] hébergé ne sont les deux pas satisfaisants mais pas aussi mauvais que les modèles DeepSeek-V2[20] et Command R+[12] hébergé. Le modèle Command R+[12] de Cohere est le seul qui a 100% de rappel, il arrive donc à identifier correctement tous les cas positifs réels, mais, en contrepartie, il fait beaucoup de fausses prédictions positives montrées par sa précision très basse et son taux d'erreur haut.

c. Fine-tuned model

Nous avons décidé de comparer le modèle affiné avec les modèles qui ont un score F1 qui dépasse 80% dans le cadre des textes de longueurs moyennes. En observant l'**illustration 4.9** :

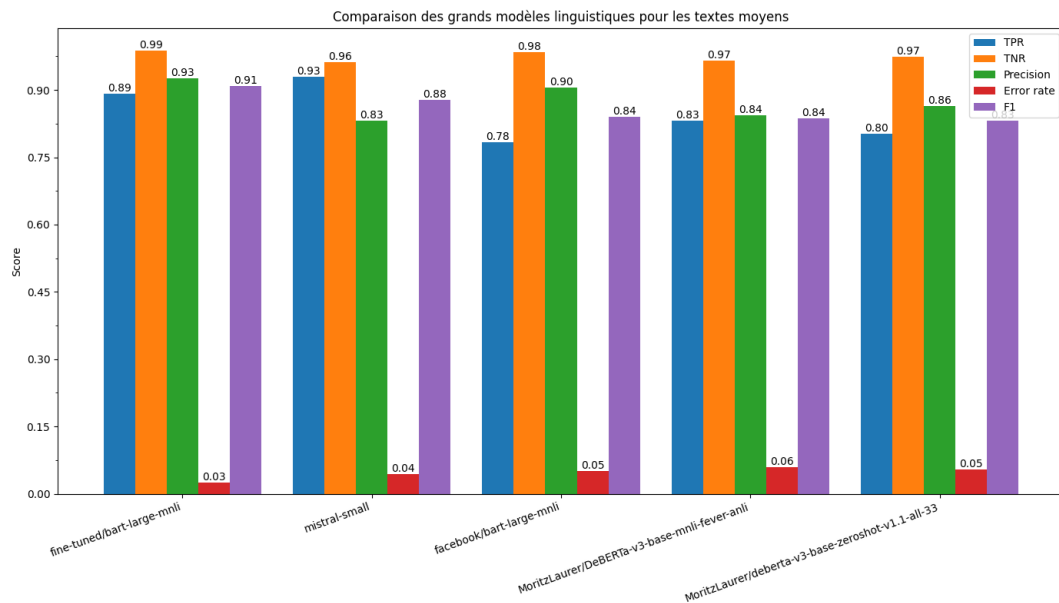


ILLUSTRATION 4.9 – Résultats pour le modèle affiné et les meilleurs modèles sur les textes moyens , Source : réalisé par Pavlovich Ivan

nous pouvons voir que le meilleur modèle est le modèle affiné avec un score F1 de 91%. Avec le modèle Mistral Small, ils sont considérés comme très satisfaisants, alors que les trois modèles de classification Zero-Shot sont considérés comme bons car leur score F1 dépasse les 80%, mais ils sont en dessous des 85%.

4.4. RÉSULTATS POUR TEXTES LONGS

Dans ce sous-chapitre nous allons observer les résultats des tests sur les textes longs, donc qui sont au-dessus de 600 caractères et en dessous de 900.

a. Classificateurs Zero-Shot

En observant le graphique suivant :

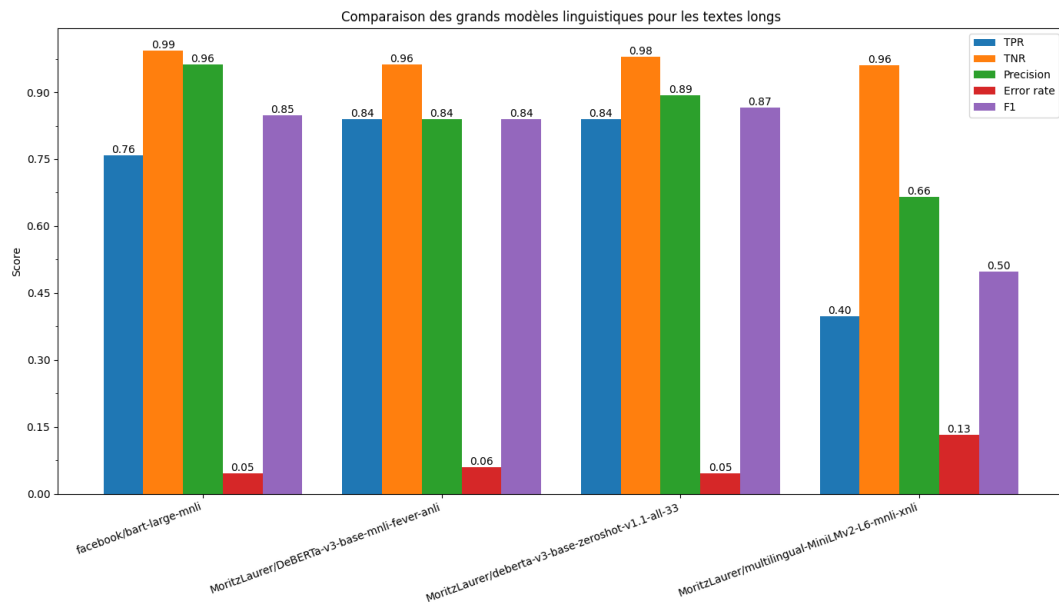


ILLUSTRATION 4.10 – Résultats pour les classificateur Zero-Shot sur les textes longs , Source : réalisé par Pavlovich Ivan

nous pouvons observer que les deux seul modèles vraiment performant sont BART-Large-MNLI[8] et deberta-v3-base-zeroshot-v1.1-all-33[16], avec deberta-v3-base-zeroshot-v1.1-all-33[16] qui peut être considéré comme très satisfaisant car son score F1 est à 85%. Le modèle DeBERTa-v3-base-mnli-fever-anli[15] n'est juste pas assez performant pour les textes longs, avec un score F1 de 78%.

b. Large Language Models

En observant le graphique suivant :

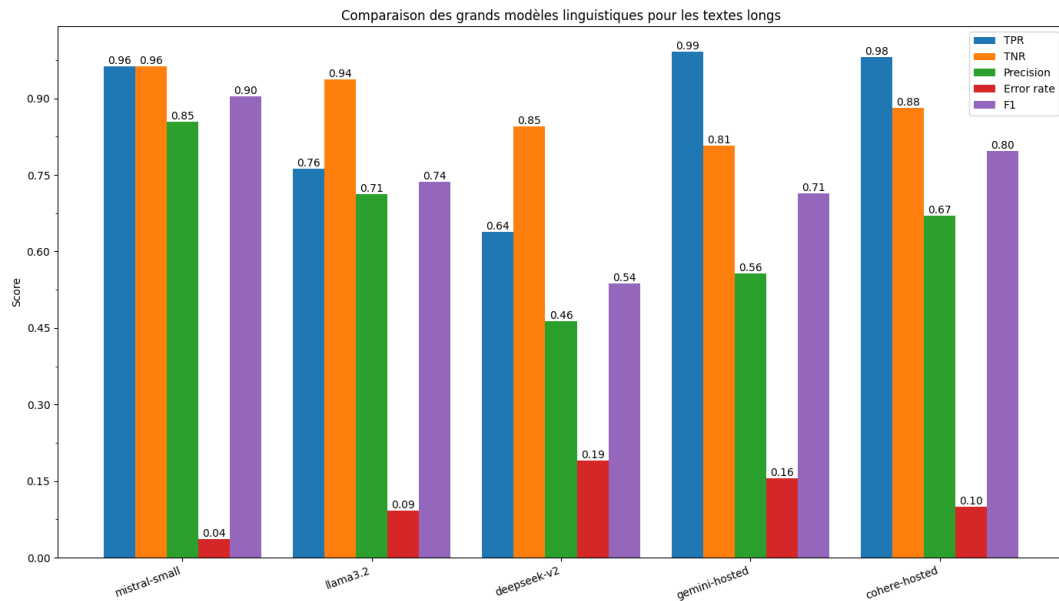


ILLUSTRATION 4.11 – Résultats pour les grands modèles linguistiques sur les textes longs ,
Source : réalisé par Pavlovich Ivan

on peut voir que le seul modèle qui est très performant est le modèle Mistral Small[18]. Il a un score F1 de 88% et il est considéré comme très satisfaisant.

c. Fine-tuned model

Nous avons décidé de comparer le modèle affiné avec les modèles qui ont un score F1 qui dépasse 80% dans le cadre des textes long. En observant le graphique suivant :

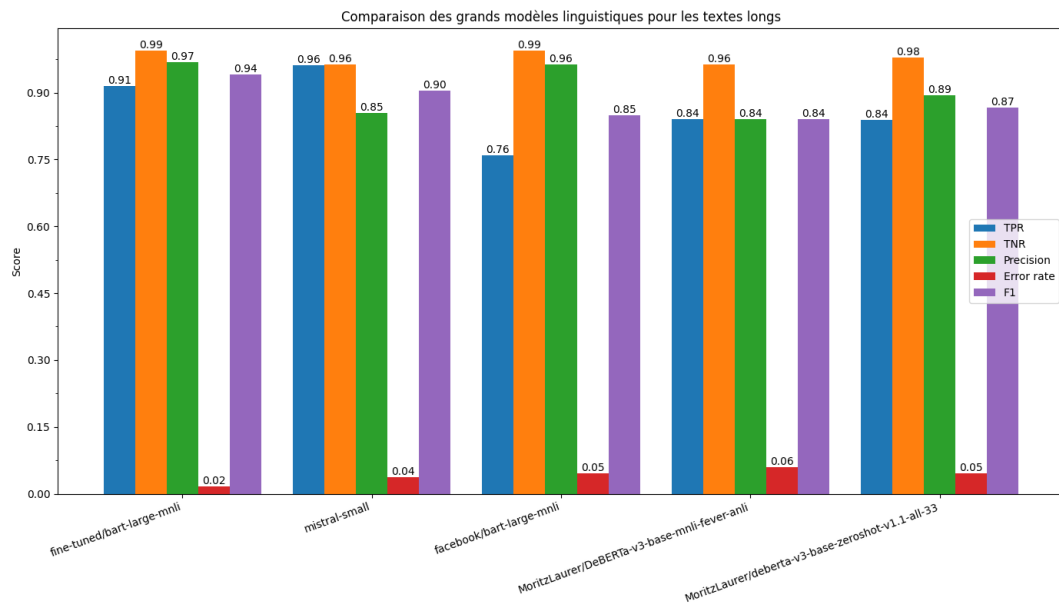


ILLUSTRATION 4.12 – Résultats pour le modèle affiné et les meilleurs modèles sur les textes longs , Source : réalisé par Pavlovich Ivan

nous observons que le meilleur modèle est le modèle affiné. Son score F1 est de 94%, ce qui largement au dessus du modèle Mistral Small[18] qui est à 88% et du modèle deberta-v3-base-zeroshot-v1.1-all-33[16] qui est à 85%. Ces trois modèles sont considérés comme très satisfaisants alors le modèle de classification Zero-Shot BART-Large-MNLI[8] est considéré comme bon car son score F1 est à 84%.

4.5. RÉSULTATS POUR TEXTE TRÈS LONG

Dans ce sous-chapitre nous allons observer les résultats des tests sur les textes qui sont très longs, donc qui sont au-dessus de 900 caractères.

a. Classificateurs Zero-Shot

En observant le graphique suivant :

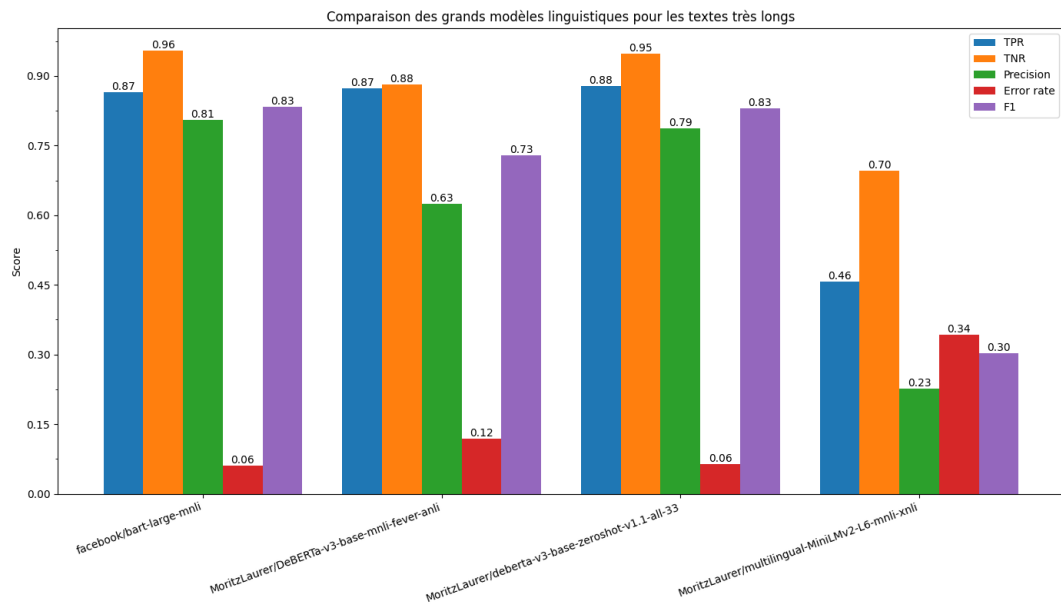


ILLUSTRATION 4.13 – Résultats pour les classificateur Zero-Shot sur les textes très longs ,
Source : réalisé par Pavlovich Ivan

nous pouvons observer que les deux seuls modèles vraiment performant sont BART-Large-MNLI[8] et deberta-v3-base-zeroshot-v1.1-all-33[16]. Le modèle DeBERTa-v3-base-mnli-fever-anli[15] n'est pas performant pour les textes très longs, avec un score F1 de 73%.

b. Large Language Models

En observant le graphique suivant :

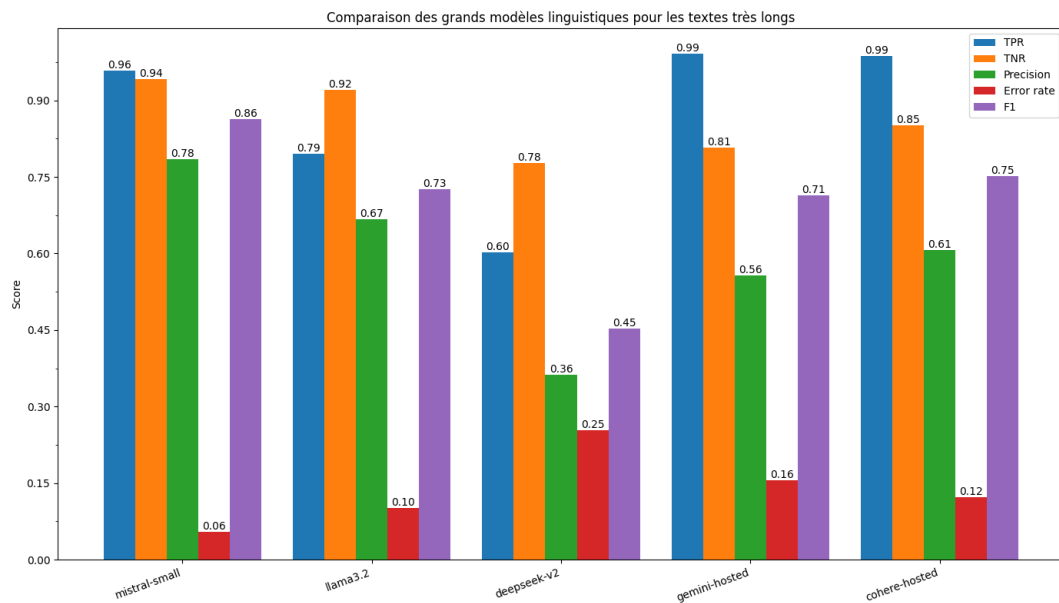


ILLUSTRATION 4.14 – Résultats pour les grands modèles linguistiques sur les textes très longs , Source : réalisé par Pavlovich Ivan

Le seul modèle vraiment satisfaisant est le modèle Mistral Smal[18] avec un score F1 de 86%.

c. Fine-tuned model

En observant le graphique suivant :

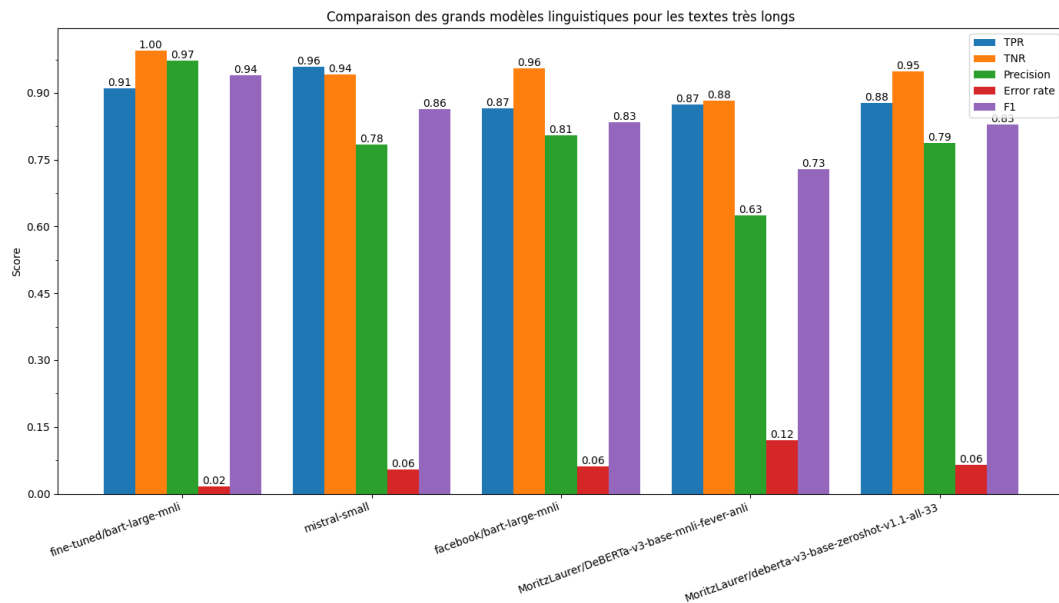


ILLUSTRATION 4.15 – Résultats pour le modèle affiné et les meilleurs modèles sur les textes très longs , Source : réalisé par Pavlovich Ivan

que les modèle Zero-Shot bon mais pas autant que Mistral Small[18] et le modèle affiné. Le modèle a un score F1 de 94%, donc 11% de plus que le modèle sur le quel il est basé.

4.6. RÉSULTATS DES TEMPS DE CLASSIFICATION

Les derniers résultats que nous allons présenter sont les temps de prédiction de classes (**Tableau 4.1**) et les temps de récupération des résultats (**Tableau 4.2**). Ce sont des résultats qui peuvent beaucoup influencer sur le choix de modèle à utiliser par la suite. Nous allons comparer les deux métriques et essayer d'expliquer les potentielles anomalies.

	ALL	SHORT	MEDIUM	LONG	VERY LONG
BART-Large-MNLI[8]	112	17	19	24	51
DeBERTa-v3-base-MNLI...	74	17	17	17	22
DeBERTa-v3-base-zeroshot...	75	17	18	18	23
Multilingual-MiniLMv2...	19	4	4	5	5
Mistral Small	516	98	109	120	189
Llama 3.2[19]	105	21	21	23	39
DeepSeek-V2[20]	327	61	63	72	130
Gemini 1.5 Flash[11]	X	X	X	X	X
Command R+[12]	1813	440	227	142	1002
Fine-Tuned	14	2	2	3	6

TABLEAU 4.1 – Temps de prédiction de classes pour chaque modèle sélectionné / réalisé par Pavlovich Ivan.

	ALL	SHORT	MEDIUM	LONG	VERY LONG
BART-Large-MNLI[8]	112	17	19	24	51
DeBERTa-v3-base-MNLI...	74	17	17	17	22
DeBERTa-v3-base-zeroshot...	75	17	18	18	23
Multilingual-MiniLMv2...	19	4	4	5	5
Mistral Small	516	98	109	120	189
Llama 3.2[19]	105	21	21	23	39
DeepSeek-V2[20]	327	61	63	72	130
Gemini 1.5 Flash[11]	7817	X	X	X	X
Command R+[12]	6613	1640	1427	1342	2202
Fine-Tuned	14	2	2	3	6

TABLEAU 4.2 – Temps de récupération des résultats pour chaque modèle sélectionné / réalisé par Pavlovich Ivan.

Nous pouvons observer que en général il n’y a pas beaucoup de différence entre les valeurs du tableau 4.1 et les valeurs du tableau 4.2. Cela est totalement normal, car le tableau 4.1 nous montre le temps que le modèle prend pour faire les prédictions pour chaque classe. Le tableau 4.2 nous donne le temps total que nous avons dû attendre pour recevoir les données prédites, donc en temps normal, la différence doit être petite car les valeurs du tableau 4.2 incluent les valeurs du tableau 4.1.

Les deux seuls modèles qui ont des valeurs qui changent beaucoup entre les deux tableaux sont les modèles hébergés. Ce test permet de voir l’impact des restrictions mises par les entreprises sur les modèles qu’ils nous mettent à disposition gratuitement.

4.7. DISCUSSION

Pour les **LLM**, nous avons trouvé que le seul modèle satisfaisant est Mistral Small[18]. A travers tous les tests réalisés, il reste toujours au-dessus de 85% de score F1, à part pour les textes de longueurs courtes où il descend à un score F1 de 82%, ce qui reste bon pour un modèle de classification.

Pour les classificateurs Zero-Shot, les résultats des trois modèles BART-Large-MNLI[8], deberta-v3-base-zeroshot-v1.1-all-33[16] et DeBERTa-v3-base-mnli-fever-anli[15] sont très similaires, même si pour les textes longs et très longs le modèle DeBERTa-v3-base-mnli-fever-anli[15] est moins bon et son score F1 descend en dessous de 80%. En général, le modèle BART-Large-MNLI[8] est meilleur que les trois autres car c’est le seul qui, dans tous les cas de figure, a un score F1 qui est égal ou plus grand que 80%.

En termes de scores des différentes métriques, le meilleur modèle de tous est le modèle BART-Large-MNLI[8] affiné sur les articles de PubMed. Il est le plus rapide pour faire la classification et il atteint dans tous les cas de figure un score F1 supérieur à 90%, ce qui est extrêmement satisfaisant. Cela indique une classification fiable et robuste dans la plupart des applications pratiques. Néanmoins, le modèle n'est pas parfait, car il y a plusieurs points à prendre en compte avant de prendre la décision. Premièrement, c'est un modèle qui doit être affiné, donc si les collaborateurs décident de changer les classes à prédire en cours de route, il devra être ré-affiné sur un nouveau jeu de données qui devra représenter les nouvelles tâches à réaliser. Deuxièmement, l'affinage est une tâche très gourmande en performances matérielles, donc si le collaborateur ne possède pas le matériel informatique nécessaire pour pouvoir exécuter cette tâche de manière efficace, ce n'est pas un modèle recommandé, car dans les pires cas de figure, l'affinage peut prendre jusqu'à des semaines entières à être réalisé.

Dans les cas de figure où l'affinage n'est pas une solution plausible, les modèles à considérer sont Mistral Small[18], BART-Large-MNLI[8] et deberta-v3-base-zeroshot-v1.1-all-33[16]. Pour les cas de figure où l'on cherche à avoir les meilleures performances sans prendre en compte le temps de classification, le meilleur choix est Mistral Small[18] de Mistral AI. Il a en moyenne un score F1 qui dépasse les 85%, mais il faut prendre en compte le fait que c'est une LLM et que la classification est réalisée par le biais d'un prompt. Le modèle a une chance de ne pas bien comprendre ce qu'on lui demande et de retourner des résultats qui ne pourront pas être récupérés par notre programme, ce qui ralentit encore un peu la classification.

Pour finir, si le temps est une métrique importante dans le cas d'utilisation des collaborateurs, le meilleur modèle à utiliser est BART-Large-MNLI[8] de Facebook AI. Il a de bonnes performances dans tous les cas de figure et son temps de classification n'est pas très élevé. Son score F1 ne passe jamais en dessous des 80%. Pour le modèle deberta-v3-base-zeroshot-v1.1-all-33[16], même s'il est plus rapide pour la classification, son score F1 passe en dessous de 80% pour les textes courts. Pour cette raison, nous conseillons d'utiliser le modèle de Facebook, mais vu que le modèle deberta-v3-base-zeroshot-v1.1-all-33[16] a quand même de bonnes performances globales, il reste un choix plausible.

CONCLUSION

En conclusion, notre projet comportait quatre parties principales. La première consistait à analyser la source PubMed. Cette source est par la suite devenue la fondation de nos tests réalisés sur les différents modèles. La deuxième partie nous a permis d'étudier et d'explorer les différentes solutions pour réaliser de la classification intelligente de textes biomédicaux. Suite à nos études, nous avons retenu dix modèles pour les essayer et vérifier leurs performances dans la tâche qui nous a été assignée. Sur les dix modèles retenus, il y avait quatre classificateurs Zero-Shot dont le modèle considéré comme le pilier de cette approche de l'apprentissage automatique, le modèle créé par Facebook AI, BART-Large-MNLI[8], cinq LLM dont deux étaient des modèles hébergés mis à disposition gratuitement par Google DeepMind et Cohere et trois modèles locaux. Le dernier modèle utilisé était une version affinée de BART-Large-MNLI[8] sur les articles de PubMed. Après avoir trouvé les modèles que nous voulions vérifier, nous avons créé un système de tests qui nous permettait d'obtenir une mesure de la performance et de l'efficacité de chaque modèle. Pour finir, nous avons réalisé les tests et en avons conclu que sur les dix modèles, ils en avaient quatre qui étaient vraiment performants et qui pourraient être utilisés selon la situation et les envies des collaborateurs. Les modèles sont : le modèle BART-Large-MNLI[8] affiné, Mistral Small[18], BART-Large-MNLI[8] et deberta-v3-base-zeroshot-v1.1-all-33[16].

Durant ce travail, nous avons rencontré plusieurs problèmes, mais aussi nous avons acquis de nouvelles connaissances dans le domaine de l'intelligence artificielle. Les plus gros problèmes rencontrés lors de ce travail ont tous été des problèmes liés au temps d'exécution des différentes parties. Cela nous a à chaque fois poussé à aller chercher des solutions alternatives. Les solutions variaient de changer notre approche à essayer de trouver une nouvelle infrastructure matérielle pour pouvoir faire nos tâches. Toutes les impasses rencontrées nous ont poussé à essayer de devenir plus indépendant et plus exhaustif dans notre façon de travailler. Pour prendre du recul, nous avons apprécié le projet dans son ensemble.

Les potentielles améliorations de ce projet sont d'explorer les classificateurs Few-Shot et One-Shot. Pour ce qui est des cas de figure où nous nous trouvons avec une quantité très petite de données, nous pourrions quand même avoir des modèles performants et efficaces.

En somme, notre travail constitue une première étape dans l'évaluation de modèles de clas-

sification biomédicale, mais il ouvre la voie à de nombreuses améliorations et perspectives.

RÉFÉRENCES DOCUMENTAIRES

- [1] N. C. for BIOTECHNOLOGY INFORMATION. “About PubMed.” Consulté le 10 novembre 2024. (n.d.), adresse : <https://pubmed.ncbi.nlm.nih.gov/about/>.
- [2] E. SAYERS, *The NCBI Handbook*. National Center for Biotechnology Information, 2009, Consulté le 10 novembre 2024. adresse : <https://www.ncbi.nlm.nih.gov/books/NBK25499/>.
- [3] NATIONAL LIBRARY OF MEDICINE. “PubMed Help - Automatic Term Mapping.” Consulté le 10 novembre 2024. (n.d.), adresse : <https://pubmed.ncbi.nlm.nih.gov/help/#automatic-term-mapping>.
- [4] MOZILLA DEVELOPER NETWORK (MDN). “HTTP 414 URI Too Long.” Consulté le 15 mars 2025. (n.d.), adresse : <https://developer.mozilla.org/fr/docs/Web/HTTP/Reference/Status/414>.
- [5] A. VASWANI, N. SHAZEER, N. PARMAR et al. “Attention Is All You Need.” Consulté le 25 mars 2025. (2017), adresse : https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [6] CONTRIBUTEURS DE WIKIPÉDIA. “Transformer (architecture d’apprentissage profond).” Consulté le 25 mars 2025. (2025), adresse : [https://en.wikipedia.org/wiki/Transformer_\(deep_learning_architecture\)](https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)).
- [7] ACADÉMIE EITCA. “Comment le mécanisme d’auto-attention dans les modèles de transformateur améliore-t-il la gestion des dépendances à longue portée dans les tâches de traitement du langage naturel ?” Consulté le 27 mars 2025. (2024), adresse : <https://fr.eitca.org/intelligence-artificielle/eitc-ai-adl-apprentissage-en-profondeur-avanc%C3%A9/traitement-du-langage-naturel/apprentissage-profond-avanc%C3%A9-pour-le-traitement-du-langage-naturel/r%C3%A9vision-d%27examen-apprentissage-profond-avanc%C3%A9-pour-le-traitement-du-langage-naturel/comment-le-m%C3%A9canisme-d%27auto-attention-dans-les-mod%C3%A8les-de-transformateur-am%C3%A9liore-t-il-la-gestion-des-d%C3%A9pendances-%C3%A0-longue-port%C3%A9e-dans-les-t%C3%A2ches-de-traitement-du-langage-naturel/>.

- [8] F. AI. “BART Large MNLI.” Consulté le 5 decembre 2024. (n.d.), adresse : <https://huggingface.co/facebook/bart-large-mnli>.
- [9] MICROSOFT AZURE. “Que sont les grands modèles de langage (LLM) ?” Consulté le 26 mars 2025. (n.d.), adresse : <https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-are-large-language-models-llms>.
- [10] IBM. “Qu’est-ce que le zero-shot prompting ?” Consulté le 26 mars 2025. (2025), adresse : <https://www.ibm.com/think/topics/zero-shot-prompting>.
- [11] G. AI. “Documentation des modèles de l’API Gemini.” Consulté le 10 février 2025. (n.d.), adresse : <https://ai.google.dev/gemini-api/docs/models>.
- [12] COHERE. “Documentation de Command R Plus.” Consulté le 10 février 2025. (n.d.), adresse : <https://docs.cohere.com/v2/docs/command-r-plus>.
- [13] L. ES. “Finetuning Huggingface Facebook Bart model.” Consulté le 18 mars 2025. (2023), adresse : <https://medium.com/@lidores98/finetuning-huggingface-facebook-bart-model-2c758472e340>.
- [14] WIKIPEDIA CONTRIBUTORS. “Hugging Face.” Consulté le 26 mars 2025. (2025), adresse : https://en.wikipedia.org/wiki/Hugging_Face.
- [15] M. LAURER. “DeBERTa-v3-base-mnli-fever-anli.” Consulté le 28 mars 2025. (n.d.), adresse : <https://huggingface.co/MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli>.
- [16] M. LAURER. “DeBERTa-v3-base-zeroshot-v1.1-all-33.” Consulté le 5 decembre 2024. (n.d.), adresse : <https://huggingface.co/MoritzLaurer/deberta-v3-base-zeroshot-v1.1-all-33>.
- [17] M. LAURER. “multilingual-MiniLMv2-L6-mnli-xnli.” Consulté le 5 decembre 2024. (n.d.), adresse : <https://huggingface.co/MoritzLaurer/multilingual-MiniLMv2-L6-mnli-xnli>.
- [18] M. AI. “Mistral Small 3.” Consulté le 15 mars 2025. (n.d.), adresse : <https://ollama.com/library/mistral-small>.
- [19] M. AI. “Llama 3.2.” Consulté le 15 mars 2025. (n.d.), adresse : <https://ollama.com/library/llama3.2>.

- [20] DEEPSEEK. “DeepSeek-V2.” Consulté le 15 mars 2025. (n.d.), adresse : <https://ollama.com/library/deepseek-v2>.