**République Algérienne Démocratique et Populaire**
**Ministère de l'Enseignement Supérieur**
**et de la Recherche Scientifique**


**Université Abderrahmane Mira de Béjaïa**
**Faculté des Sciences Exactes**
**Département d'Informatique**

# Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de

## Master en Informatique

**Option : Intelligence Artificielle**

***Thème***

*Conception et réalisation d'un système de reconnaissance de sentiments issus des médias sociaux*

**Réalisé par :**

Ms. MEDJEBAR Thanina
Ms. MEHIDI Siham

**Devant le jury composé de :**

**Mr. BOUCHEBBAH Fatah** Président du jury
**Mr. ACHEROUFENE Achour** Examinateur
**Mme. BOULAHROUZ Djamila** Examinateur
**Mme. SAAD Narimane** Examinateur

**Sous l'encadrement de : Mr. AMROUN Kamal**

**Année universitaire : 2024 – 2025**

# Remerciements

Nous souhaitons commencer ces remerciements par quelques mots pour exprimer notre reconnaissance envers ceux qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Nous remercions **Dieu Tout-Puissant**, pour nous avoir accordé la force, la patience et la persévérance tout au long de notre parcours universitaire.

Nous exprimons notre profonde gratitude à notre encadrant, ***Dr. AMROUN Kamal***, pour son accompagnement, ses conseils avisés et sa disponibilité.

Nous remercions également l'ensemble de nos enseignants, depuis la première année jusqu'à aujourd'hui, pour les connaissances qu'ils nous ont transmises et pour leur engagement dans notre formation.

# Dédicaces

## ✦ Dédicace de Siham

Je dédie ce mémoire à :

***À mes chers parents***, dont la patience, les sacrifices et l'amour inconditionnel ont toujours été ma plus grande source de force. Merci pour votre soutien sans faille et pour avoir toujours cru en moi, même dans les moments de doute.

***À mon frère Anis, et à mes trois sœurs Meriem, Lynda et Basma***, pour leur présence rassurante, leurs encouragements constants et leur confiance en mes capacités. Vous avez toujours su me motiver à donner le meilleur de moi-même.

***À mon amie Mélissa***, Tu as été cette présence rare qu'on ne cherche pas, mais qui tombe juste. Dans mes doutes, tu étais le calme. Dans mes silences, tu lisais clair. Merci d'avoir été là, avec cette tendresse discrète qui ne demande rien, mais qui change tout.

***À mon amie Sarah***, pour ton amitié fidèle et ta disponibilité. Merci pour tes mots rassurants et ton soutien dans les moments clés.

***À mon encadrant, Monsieur Amroun Kamal***, dont l'accompagnement attentif et les conseils avisés ont grandement contribué à la réalisation de ce mémoire.

***À tous ceux et celles*** qui, de près ou de loin, m'ont accompagnée, inspirée et soutenue dans la réalisation de ce mémoire.

**Que ce mémoire soit le reflet de tout ce que vous m'avez transmis et l'écho de la confiance que vous avez toujours placée en moi.**

## ✦ Dédicace de Thanina

Je dédie ce travail de fin d'études à :

***La personne que j'étais***, qui a porté le rêve bien avant de savoir marcher vers lui, et qui n'a jamais cessé d'y croire.

***Mes parents***, dont la présence a marqué mon parcours a sa manière.

***Mes sœurs Hayet et Sabrina***, loin des yeux mais proches du cœur, toujours la sans condition.

***Mon frère Farouk***, cet appui discret, constant et profondément rassurant.

***Mon amie Yasmine***, la première à m'avoir tendu la main dans mes moments les plus sombres, et celle qui ne m'a jamais laissée tomber.

***Mon amie Anissa***, dont les mots rares ont toujours su remettre mes objectifs au centre.

***Mon ami Youcef***, pour cette façon rare de comprendre, et d'offrir sans jamais rien attendre.

**À tous ceux qui ont traversé ma vie, ceux qui m'ont portée et ceux qui m'ont bousculée. À chacun d'eux, pour avoir, d'une manière ou d'une autre, contribué à ce que je suis devenue.**

# À la mémoire de Massinissa Rahmani



*Pour ce qu'il a été.*

*Pour ce qu'il continue de laisser,*

*discrètement, dans ceux qui l'ont connu.*

# Table des matières

# Liste des tableaux

# Table des figures

# Liste des Abréviations

| | |
|---|---|
| IA | Intelligence Artificielle |
| NLP | Natural Language Processing (Traitement automatique du langage naturel) |
| ML | Machine Learning (Apprentissage automatique) |
| ABSA | Aspect-Based Sentiment Analysis (Analyse de sentiments par aspects) |
| SVM | Support Vector Machine |
| LSTM | Long Short-Term Memory |
| WEB | World Wide Web |
| GPT | Generative Pretrained Transformer |
| TALN | Traitement Automatique du Langage Naturel |
| BERT | Bidirectional Encoder Representations from Transformers |
| KNN | K-Nearest Neighbors |
| RNN | Recurrent Neural Network |
| GRU | Gated Recurrent Unit |
| XLNet | Generalized Autoregressive Pretraining for Language Understanding |
| MSA | Analyse Multimodale de Sentiments |
| TP | Vrais Positifs (True Positives) |
| FP | Faux Positifs (False Positives) |
| FN | Faux Négatifs (False Negatives) |
| TN | Vrais Négatifs (True Negatives) |
| LIME | Local Interpretable Model-agnostic Explanations |
| SHAP | SHapley Additive exPlanations |
| B2B | Business to Business |

# Introduction générale

À l'ère du numérique, les réseaux sociaux sont devenus un espace privilégié où les individus s'expriment librement, partageant leurs opinions, ressentis et réactions face aux événements du quotidien. Cette multiplication de contenus textuels constitue une source précieuse d'informations exploitables, notamment dans le cadre de l'analyse des sentiments. Ce domaine, relevant du Traitement Automatique du Langage Naturel (TALN), vise à détecter et classifier les émotions ou opinions véhiculées dans des textes, en identifiant leur polarité (positive, négative ou neutre).

Cependant, malgré les avancées significatives dans ce domaine, certains défis majeurs subsistent. Parmi ceux-ci figurent la reconnaissance du sarcasme, des doubles négations et des tournures linguistiques complexes, qui échappent souvent aux modèles d'analyse classiques. Ces formes de langage détourné, volontairement ambigu ou ironique, rendent l'interprétation sémantique plus difficile et affectent la performance des systèmes traditionnels de classification. Ainsi, il devient nécessaire de concevoir des solutions innovantes, capables de dépasser les simples approches de détection de polarité.

C'est dans ce contexte que s'inscrit le présent travail. L'objectif de ce mémoire est de concevoir et de mettre en œuvre un système de reconnaissance des sentiments issus des médias sociaux, en intégrant des techniques avancées d'Intelligence Artificielle, notamment l'apprentissage profond, afin de mieux capter les formes complexes de sentiments.

Notre démarche s'est articulée autour de plusieurs étapes complémentaires. Dans un premier temps, nous avons évalué les performances de plusieurs modèles de pointe notamment : RoBERTa-base, XLNet et DeBERTa sur un corpus standard de tweets. Cette phase exploratoire visait à établir une base de comparaison fiable et à identifier les limites des approches existantes. RoBERTa-base s'est distingué par ses performances supérieures, ce qui nous a conduits à l'adopter comme base de travail.

Nous avons ensuite construit un dataset personnalisé combinant spécifiquement le sarcasme et la double négation, deux phénomènes linguistiques encore mal maîtrisés par les systèmes classiques. Ce corpus, élaboré à partir de plusieurs sources, a permis d'entraîner RoBERTa-base afin d'évaluer sa capacité à gérer ces cas complexes. Si les résultats quantitatifs étaient prometteurs, l'analyse qualitative a révélé que le modèle ne parvenait pas à bien prédire les nuances des textes ambigus, ce qui a mis en évidence la nécessité d'aller plus loin.

Dans cette optique, nous avons entamé une phase d'amélioration du modèle RoBERTa-base. Nous avons d'abord divisé notre dataset en deux sous-corpus, l'un dédié au sarcasme,

l'autre à la double négation. Le corpus du sarcasme a été annoté selon trois niveaux de subtilité : non sarcastique, sarcasme modéré et sarcasme exagéré, tandis que celui de la double négation est resté binaire. Nous avons ensuite entraîné le modèle amélioré sur ces deux jeux de données.

Les résultats obtenus sur la double négation ont été satisfaisants, aussi bien sur le plan quantitatif que qualitatif. En revanche, pour le sarcasme, les performances sont restées limitées, en particulier dans les cas les plus ambigus. Cela nous a conduit à concevoir une méthode personnalisée, développée de A à Z, spécifiquement dédiée à la reconnaissance du sarcasme. Cette dernière a permis d'obtenir des résultats nettement meilleurs, dépassant les performances des approches précédentes aussi bien sur le plan quantitatif que qualitatif.

La finalité de cette démarche est d'améliorer la robustesse, la précision et la pertinence des analyses effectuées sur des données textuelles souvent brutes, spontanées et non structurées, et plus particulièrement celles véhiculant des émotions exprimées de manière implicite, ambiguë ou détournée.

Le mémoire s'organise autour de six chapitres :

- **Chapitre 1** : présente les généralités liées à l'analyse des sentiments, en définissant les concepts fondamentaux tels que l'opinion, l'émotion, et les principales approches existantes.

- **Chapitre 2** : est consacré à un état de l'art détaillé, mettant en lumière les méthodes traditionnelles et modernes, ainsi que les travaux de recherche les plus pertinents autour de l'analyse de sentiment.

- **Chapitre 3** : présente la méthodologie adoptée, en commençant par une première série d'expérimentations sur un dataset standard à l'aide de plusieurs modèles préentraînés afin d'identifier le plus performant, puis en détaillant la construction d'un dataset personnalisé intégrant le sarcasme et la double négation, les choix d'annotation, ainsi que l'entraînement de ce meilleur modèle sur ce corpus.

- **Chapitre 4** : détaille les améliorations apportées au modèle *RoBERTa-base*, l'adaptation du dataset en sous-corpus (sarcasme et double négation), et l'analyse des résultats obtenus, à la fois sur le plan quantitatif et qualitatif.

- **Chapitre 5** : introduit la conception et la mise en œuvre d'un modèle personnalisé, conçu de bout en bout pour la détection du sarcasme, avec des résultats supérieurs aux approches précédentes.

- **Chapitre 6** : propose une analyse approfondie des erreurs commises par les différents systèmes développés, en identifiant leurs causes, en catégorisant les types d'erreurs, et en suggérant des pistes d'amélioration.

# Chapitre 1

# Généralités

## 1.1 Introduction

Les réseaux sociaux sont des plateformes clés où les utilisateurs expriment constamment leurs opinions, émotions et idées sur divers sujets. L'analyse de sentiments permet d'extraire et de comprendre ces sentiments, qu'ils soient positifs, négatifs ou neutres.

Toutefois, cette analyse fait face à plusieurs défis, tels que l'ambiguïté du langage, l'ironie, l'utilisation des émojis et la compréhension du contexte global des messages. Ce chapitre abordera les concepts fondamentaux de l'analyse de sentiments, tout en présentant les défis auxquels elle est confrontée, la thématique ainsi que la solution proposée.

## 1.2 Définitions

### 1.2.1 Opinion [1][2]

Une opinion est une expression subjective qui reflète le point de vue, l'évaluation ou le jugement d'une personne sur un sujet, un produit, un service ou un événement. Dans l'analyse de sentiments, les opinions sont souvent analysées pour déterminer leur polarité (positive, négative ou neutre) et leur intensité. Elle est généralement composée de :

- Un sujet (ce dont on parle).
- Un attribut (une caractéristique du sujet).
- Un sentiment (positif, négatif ou neutre).
- Un détenteur d'opinion (la personne qui exprime l'opinion).
- Un moment (le contexte temporel de l'opinion).

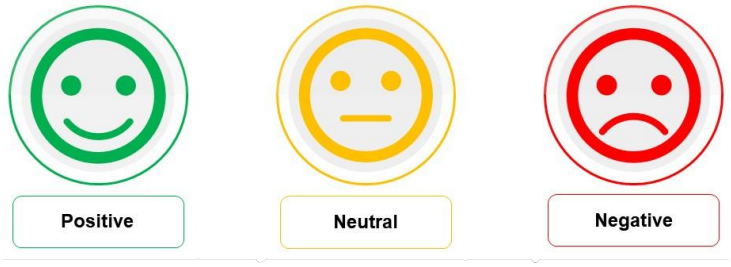


FIGURE 1.1 – Sentiments caractérisant une opinion [33]

### 1.2.2 Émotion[3]

Une émotion est une réaction affective intense, souvent de courte durée, déclenchée par un stimulus externe ou interne. Contrairement à l'opinion, qui est plus rationnelle et évaluative, l'émotion est plus instinctive et liée à des états psychologiques comme la joie, la tristesse, la colère ou la peur. Elles sont souvent caractérisées par :

- Une valence (positive ou négative).
- Une intensité (faible ou forte).
- Des dimensions (comme dans le modèle de Plutchik : joie, tristesse, colère, peur, etc.).

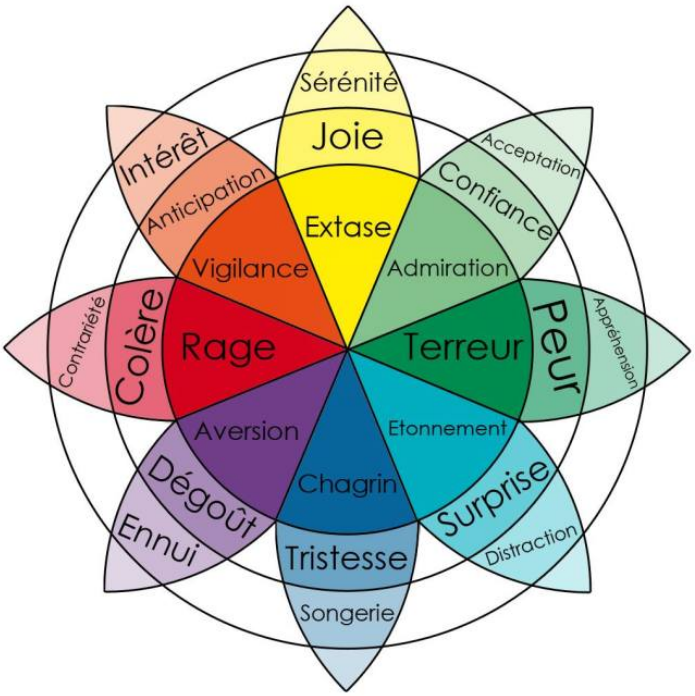


FIGURE 1.2 – Modèle de Plutchik
[4]

### 1.2.3 Différence entre opinion et émotion

- Une opinion est une évaluation raisonnée, souvent exprimée de manière explicite **(exemple : "Ce film est excellent")**.
- Une émotion est une réaction affective, souvent implicite et liée à des états psychologiques **(exemple : "Ce film m'a fait pleurer")**.

### 1.2.4 Applications dans l'analyse de sentiments

- **Opinions :** Utilisées pour évaluer la satisfaction client, les critiques de produits, ou les tendances sur les réseaux sociaux.
- **Émotions :** Utilisées pour comprendre les réactions affectives des utilisateurs.

## 1.3 L'analyse de sentiments

### 1.3.1 Définition [5]

L'analyse de sentiment (ou sentiment analysis en anglais) est le processus qui consiste à déterminer l'opinion, le jugement et l'émotion qui se cache derrière le langage naturel.

Lorsque les individus laissent des avis en ligne, formulent des commentaires sur une marque ou répondent à des études de marché, leurs appréciations sont forcément teintées par des sentiments, qu'ils soient positifs, négatifs ou neutres.

Quand un consommateur exprime une opinion en tapant son avis dans une zone de texte, celle-ci peut être transformée en données catégorielles (par exemple « positif », « négatif », ou « neutre »). Une fois toutes les réponses catégorisées, l'entreprise peut dresser des rapports d'enquêtes pour obtenir une vue d'ensemble des ressentis et opinions des interrogés.

### 1.3.2 Domaines connexes [6]

- **Analyse des sentiments et traitement du langage naturel (NLP)** : L'analyse des sentiments est une sous-catégorie du traitement du langage naturel, ce qui signifie qu'elle n'est que l'une des nombreuses tâches effectuées par le NLP. Le traitement du langage naturel permet aux ordinateurs de comprendre le langage humain écrit ou parlé. Les tâches du NLP incluent la reconnaissance d'entités nommées, la réponse aux questions, la synthèse de texte, la détection de la langue et la génération du langage naturel.

- **Analyse des sentiments et Machine Learning (ML)** : L'analyse des sentiments utilise le Machine Learning pour effectuer l'analyse d'un texte donné. Le Machine Learning utilise des algorithmes qui "apprennent" lorsqu'ils reçoivent des données d'entraînement. Grâce au Machine Learning, l'analyse des sentiments évolue constamment pour mieux interpréter le langage qu'elle analyse.

- **Analyse des sentiments et intelligence artificielle (IA)** : Il ne faut pas confondre analyse des sentiments et intelligence artificielle. L'IA fait plus largement référence à la capacité d'une machine à imiter les capacités humaines d'apprentissage et de résolution des problèmes. Le Machine Learning est un sous-ensemble de l'IA, l'analyse des sentiments du Machine Learning est donc également un sous-ensemble de l'IA. Bien que les trois soient liés, ils sont différents.

- **Analyse des sentiments et exploration des données** : L'analyse des sentiments est une forme d'exploration des données qui explore spécifiquement des données textuelles à des fins d'analyse. L'exploration des données désigne simplement le processus d'extraction et d'analyse de grands ensembles de données pour découvrir divers types d'informations et de schémas.

## 1.4 Types d'analyse de sentiments[6]

Il existe plusieurs types d'analyse des sentiments, qu'il s'agisse d'une analyse basée sur des règles, d'une analyse par Machine Learning ou d'une analyse hybride. Ces types d'analyse incluent :

- **L'analyse fine des sentiments :**ou analyse graduée des sentiments, permet à une entreprise d'étudier les notes des clients dans les avis. L'analyse fine affine également les polarités en plusieurs catégories : très positive, positive, neutre, négative et très négative. Ainsi, par exemple, un avis 1 étoile sera considéré comme très négatif, un avis 3 étoiles comme neutre et un avis 5 étoiles comme très positif.

- **L'analyse des sentiments basée sur l'aspect :** ou ABSA, se concentre sur le sentiment envers un seul aspect d'un service ou d'un produit. Par exemple, une entreprise technologique lance un nouveau casque sans fil. Dans ce cas, les aspects à prendre en compte sont la connectivité, la conception esthétique et la qualité du son. Grâce à une classification d'analyse demandée, l'analyse des sentiments basée sur l'aspect permet à une entreprise de capturer ce que les clients pensent d'une partie spécifique de son produit ou service. "Je trouve ce casque joli" indique un sentiment envers la conception esthétique du casque. "J'aime son apparence, mais le contrôle du volume est problématique" peut alerter l'entreprise sur un défaut de conception pratique.

- **L'analyse des sentiments de détection des émotions :** elle va au-delà de la détection de polarité pour identifier les sentiments des clients tels que le bonheur, la tristesse ou la colère. Ce type d'analyse peut utiliser des lexiques pour évaluer un langage subjectif. Des mots comme affreux et scandaleux suggèrent la colère. Malheureux et bouleversant peuvent indiquer de la tristesse. Génial ou super peuvent suggérer le bonheur. Bien sûr, les lexiques ne tiennent pas compte du contexte et les gens expriment leurs émotions de différentes manières. Prenons l'exemple suivant :
  Des mots comme "coincé" et "frustrant" signifient une émotion négative, tandis que "généreux" est positif. Ce sentiment est nuancé et l'émotion difficile à classer.

- **L'analyse des sentiments basée sur l'intention :**elle permet à une entreprise d'identifier l'intention et les niveaux d'intérêt des clients. Parmi les types d'intention, on peut citer l'achat, la mise à niveau, la rétrogradation, l'annulation ou le désabonnement. L'analyse basée sur l'intention nécessite un entraînement de classification avec du texte pertinent, tel que des e-mails ou des requêtes de clients. Par exemple, "Je n'ai plus d'espace de stockage, que dois-je faire ?" pourrait être classé comme une opportunité de mise à niveau. L'intention dans "Je n'aime pas les échantillons que je reçois, je n'ai pas besoin de plus d'eye-liners" pourrait être classée comme une annulation, mais alerte également l'entreprise sur une opportunité d'amélioration du service. Ce type d'analyse aide les entreprises à gérer et à maintenir leur base de clientèle, et à optimiser les opportunités de vente.

## 1.5 Les approches de l'analyse de sentiments [6]

L'analyse des sentiments peut être abordée de trois manières différentes :

- **L'analyse des sentiments basée sur des règles :** utilise des algorithmes écrits manuellement, ou règles, pour évaluer le langage. Ces règles utilisent des méthodes de linguistique informatique, telles que la conversion en tokens, la lemmatisation, la racinisation et l'étiquetage morpho-syntaxique. Elles peuvent également utiliser des lexiques (banques de mots).

  Ce type d'analyse analysera des mots spécifiques dans les phrases, puis évalue leur polarité et leur subjectivité pour déterminer le sentiment et l'intention. Une fois qu'une polarité (positive, négative) est attribuée à un mot, une approche basée sur des règles compte le nombre de mots positifs ou négatifs apparaissant dans un texte donné pour déterminer son sentiment général.

  L'inconvénient évident est que ce type de système nécessite des efforts considérables pour créer toutes les règles. De plus, ces règles ne tiennent pas compte de la façon dont les mots sont utilisés dans une phrase (leur contexte). Même si de nouvelles règles peuvent être écrites pour tenir compte de la complexité, cela affecte la complexité générale de l'analyse. Une évaluation et un ajustement réguliers sont également nécessaires pour maintenir la précision de cette approche.

- **L'analyse des sentiments de Machine Learning :** est une version automatisée de l'analyse des sentiments basée sur des règles qui s'appuie plutôt sur des capacités de Machine Learning (ML). Ce modèle nécessite que l'outil d'analyse des sentiments de ML soit alimenté avec des données d'entraînement afin qu'il puisse apprendre quels mots correspondent à quelles polarités. Voici quelques exemples courants : critiques de films, avis sur des produits Amazon ou établissements notés par Yelp. Hugging Face, une communauté d'IA, fournit des bibliothèques, des ensembles de données et des modèles open source pouvant aider à créer et à entraîner des outils d'analyse des sentiments.

  Une fois l'entraînement de l'analyse des sentiments de Machine Learning terminé, le processus se résume à l'extraction et à la classification des caractéristiques. Pour produire des résultats, une méthode d'analyse des sentiments de Machine Learning s'appuie sur différents algorithmes de classification, tels que le Deep Learning, Naïve Bayes, les régressions linéaires ou les machines à vecteurs de support.

- **L'analyse des sentiments hybride :** combine des méthodes d'analyse des sentiments basée sur des règles et de Machine Learning. Lorsqu'elle est adaptée aux besoins spécifiques d'une entreprise ou d'un utilisateur, elle peut se révéler le plus précis des outils. Elle est particulièrement utile lorsque les sentiments sont plus subtils, comme dans la communication interentreprises (B2B) où les émotions négatives sont exprimées de manière plus professionnelle.

## 1.6 Processus de l'analyse de sentiments [7]

L'analyse de sentiments fonctionne en plusieurs étapes, impliquant diverses techniques et divers algorithmes.

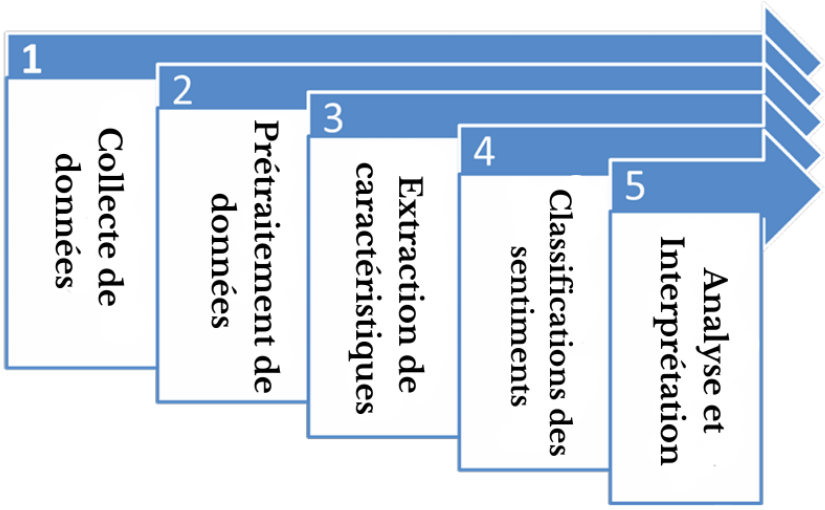


FIGURE 1.3 – Processus de l'analyse de sentiments

- **Collecte de données :** Les données textuelles sont collectées à partir de diverses sources comme les réseaux sociaux, les sites web et forums, les blogs, et les avis publiés en ligne. Ces données peuvent être structurées (comme les questionnaires) ou non structurées (comme les commentaires en ligne).

- **Prétraitement des données :** Avant d'analyser le langage naturel dans un texte, il est essentiel de le nettoyer et de le préparer. Cela inclut la suppression des mots inutiles (stop words), la normalisation des mots (lemmatisation ou stemming), et la correction des fautes de frappe.

- **Extraction des caractéristiques :** Les caractéristiques pertinentes sont extraites du texte pour l'analyse. Cela peut inclure des mots individuels, des n-grams (groupes de mots), ou des aspects syntaxiques et sémantiques du texte.

- **Classification des sentiments :** Des algorithmes de Machine Learning ou des techniques de NLP (langage naturel) sont utilisés pour classifier le texte en différentes catégories de sentiment. Divers outils d'analyse des sentiments sont utilisés pour classifier le texte en différentes catégories de sentiment. Les modèles les plus courants incluent les modèles de régression logistique, les machines à vecteurs de support (SVM), et les réseaux neuronaux profonds (comme les LSTM et les Transformers).

- **Analyse et interprétation :** Les résultats de l'analyse sont interprétés pour en tirer des conclusions. Les entreprises peuvent utiliser ces insights pour améliorer leurs produits, ajuster leurs stratégies marketing, former leur service client, et répondre de manière plus efficace aux besoins des consommateurs.

## 1.7 Les taches de l'analyse de sentiments [8]

Les instruments d'analyse des sentiments peuvent être impliqués par des associations ou entreprises, pour tout un assortiment d'utilisations, notamment :

- Reconnaître la pleine conscience, la notoriété et la prévalence de la marque à une seconde donnée ou à long terme.
- À la suite de la collecte par l'acheteur de nouveaux articles ou de faits saillants.
- Évaluation du résultat d'un effort de présentation.
- Identification d'un groupe d'intérêt ou de socio-économie.
- Recueillir les commentaires des clients à partir de médias, de sites ou de structures en ligne basés sur le Web.
- Diriger l'arpentage statistique.
- Trier des demandes d'assistance à la clientèle.

## 1.8 Défis courants de l'analyse des sentiments [9]

Le langage est un outil de communication humaine complexe, imparfait et en constante évolution. L'analyse des sentiments reposant sur l'interprétation du langage, elle est intrinsèquement difficile.

- **Avis interentreprises :** Comprendre les avis des concurrents est l'un des défis de l'analyse des sentiments. Si une entreprise établit une règle pour qualifier de positif un certain langage décrivant un sentiment envers elle, le même langage utilisé pour décrire un concurrent sera également considéré comme positif. Par exemple :

  — J'aime la rapidité avec laquelle [votre entreprise] expédie ses produits.
  — J'aime le fait de pouvoir définir ma plage de livraison avec [votre concurrent].

  Ces deux déclarations sont positives, mais l'outil d'analyse des sentiments ne fera pas la distinction entre une entreprise et ses concurrents, sauf s'il est entraîné à reconnaître ce qui est positif concernant les concurrents comme négatif.

- **Ironie, sarcasme et contexte :** La détection et la compréhension de l'ironie et du sarcasme constituent un autre défi de l'analyse des sentiments. Le sarcasme utilise des mots positifs pour décrire des sentiments négatifs mais il n'existe souvent aucun indice textuel permettant à une machine de distinguer le sérieux du sarcasme ou de l'ironie.

  Par exemple, en réponse à *"Vous aimez la pulpe dans le jus d'orange ?"*, *"Oh oui j'adore"* pourrait être compris comme positif si l'auteur était sincère ou comme négatif s'il était sarcastique.
  Le contexte peut également fausser le sentiment. Prenons les deux réponses suivantes :
  -*"Juste un petit peu."* -*"Énormément !"*
  Si les commentaires sont en réponse à une question du type *"Est-il probable que vous recommandiez ce produit ?"*, la première réponse est considérée comme négative, alors que la seconde est positive. Cependant, si la question est *"Dans quelle me-*

*sure l'ajustement des prix vous a-t-il dérangé ?"*, les polarités sont inversées.

- **Différences culturelles :** L'utilisation d'une langue en fonction de la culture représente l'un des principaux défis de l'analyse des sentiments. Voyez à quel point l'humour est différent d'une culture à l'autre. Même dans la langue française, les différences dialectiques rendent la distinction du sens complexe.

  Par exemple : En français de France, un *"dépanneur"* est une personne qui dépanne des véhicules. Au Canada, un "dépanneur" est une épicerie de quartier.

  De telles différences affectent la précision de l'analyse. Les expressions idiomatiques diffèrent également d'une culture à l'autre. Leur analyse présente un défi similaire.

- **Subjectivité :** L'un des principaux défis de l'analyse des sentiments est que le langage est subjectif. Cela complique la classification en catégories, aspects ou polarités bien définis. Prenons l'exemple suivant :

  — "Ce téléphone est super" indique clairement un sentiment positif.
  — "Ce téléphone est petit" est plus difficile à classer.

  Selon les sentiments de l'auteur sur la taille, il peut s'agir d'une déclaration positive, neutre ou négative.

  La signification d'un mot donné peut être subjective en raison du contexte, de l'utilisation de l'ironie ou du sarcasme, et d'autres particularités du discours.

## 1.9 Quelques applications principales

Voici quelques-unes des principales applications, illustrées dans la figure ci-dessous :


FIGURE 1.4 – Quelques applications principales de l'analyse de sntiments

## 1.10 Les évolutions possibles de l'analyse de sentiments [7]

L'analyse des sentiments continue d'évoluer à mesure que de nouvelles technologies émergent et que les besoins des entreprises et des consommateurs évoluent. Dans les prochaines années, il n'est pas impossibles que cette technologie soit plus aboutie, et donc plus efficaces sur plusieurs points grâce à l'amélioration du Machine Learning :

- **Amélioration de la précision grâce au Deep Learning :**

  **Modèles de langage plus avancés :** Le développement de modèles de langage encore plus avancés, basés sur le deep learning et les transformers, comme GPT (Generative Pre-trained Transformer), permettra une compréhension plus fine du contexte et des nuances dans les données textuelles, améliorant ainsi la précision de l'analyse des sentiments.

  **Apprentissage continu :** Les modèles de deep learning continueront à être entraînés sur de vastes ensembles de données afin d'améliorer leur capacité à comprendre les subtilités du langage naturel et à s'adapter à l'évolution des expressions et processus linguistiques et des tendances culturelles.

- **Analyse multimodale :** On parle ici d'une combinaison de données textuelles, visuelles et audio. L'intégration de l'analyse des sentiments avec d'autres modalités de données, telles que les images, les vidéos et l'audio, permettra une compréhension plus précise et plus approfondie des émotions et des opinions des utilisateurs. Cela pourrait ouvrir de nouvelles possibilités pour les applications dans les médias sociaux, la publicité et la surveillance de la marque.

- **Adaptation culturelle :** Les progrès dans la compréhension des différences culturelles et linguistiques permettront aux systèmes d'analyse des sentiments de s'adapter plus efficacement aux particularités linguistiques et culturelles de différentes régions du monde. Cela rendrait l'analyse des sentiments plus précise et pertinente à l'échelle mondiale.

- **Analyse de sentiments en temps réel :** Les entreprises chercheront à intégrer l'analyse des sentiments en temps réel dans leurs outils de surveillance des médias sociaux. Cela leur permettrait de détecter et de réagir rapidement aux tendances émergentes, aux crises potentielles et aux opportunités d'engagement-client.

- **Prise en compte du contexte :** Les futurs systèmes d'analyse des sentiments seront capables de mieux comprendre le contexte dans lequel les expressions sont utilisées. Cela permettra une interprétation plus précise des sentiments et des intentions des utilisateurs. Les risques de mauvaise interprétation se verront ainsi réduits et la qualité des insights générés améliorée.

- **Protection de la vie privée et éthique :** Les développements futurs de l'analyse des sentiments mettront l'accent sur la protection de la vie privée des utilisateurs en garantissant que les données personnelles sont traitées de manière responsable et conforme aux réglementations en matière de protection des données.

- **Personnalisation avancée :** Les entreprises utiliseront l'analyse des sentiments pour offrir des expériences plus personnalisées et adaptées aux émotions et aux préférences des utilisateurs. Cela, en proposant du contenu, des produits et des services qui répondent mieux à leurs besoins émotionnels.

- **Intégration avec l'IA conversationnelle :** L'intégration de l'analyse des sentiments avec les technologies d'IA conversationnelle, comme les chatbots et les assistants virtuels, permettra des interactions plus empathiques et personnalisées. Cela améliorera l'expérience utilisateur tout en renforçant les relations client.

## 1.11 L'importance de l'analyse de sentiments [8]

Par-dessus tout, l'analyse des sentiments est importante parce que les sentiments et les perspectives vers un point, peuvent devenir des extraits remarquables de valeurs de données, dans divers domaines de l'entreprise et de la recherche.

En outre, cela permet d'économiser du temps et des efforts, car le cours de l'extraction des sentiments est entièrement informatisé. Dans l'analyse de calcul et les ensembles de données d'opinion traités de cette manière, la coopération humaine est rare.

Seriez-vous en mesure d'envisager de parcourir le Web, de rechercher des textes importants, de les comprendre et d'examiner le ton qu'ils transmettent « physiquement » ? C'est possible, cependant, il est possible que cela prenne des siècles.

De plus, il se transforme en un thème de plus en plus célèbre au fur et à mesure que des perfectionnements sont créés en matière de raisonnement artificiel, d'apprentissage profond, de procédures d'IA et de traitement du discours standard.

Ensuite, à mesure que l'innovation peut se créer, l'analyse des sentiments sera plus ouverte et plus raisonnable pour la société en général et pour les entreprises, plus modestes.

En conclusion, les instruments deviennent de plus en plus brillants de manière constante. Plus ils sont pris en charge avec des informations, plus ils deviennent brillants et précis dans l'extraction de l'opinion

## 1.12 Thématique

Dans une ère où les utilisateurs expriment librement leurs sentiments et points de vue sur les réseaux sociaux, l'analyse de sentiments devient un défi majeur pour comprendre le sens réel de ces derniers. La thématique de ce mémoire s'inscrit dans le domaine de l'Intelligence Artificielle appliquée au Traitement Automatique du Langage Naturel (TALN), en particulier dans la détections de sentiments complexes tels que le sarcasme et la double

négation qui généralement sont difficiles à interpréter pour les systèmes classiques.

Le but principal est de concevoir et de mettre en œuvre des systèmes capables d'analyser des contenus textuels issus des réseaux sociaux tout en tenant compte de ces formes d'ambiguïté sémantique. Cette démarche vise à améliorer la robustesse et la précision de la classification des sentiments, en allant au-delà des méthodes traditionnelles de détection de polarité simple.

## 1.13 Conclusion

En conclusion, l'analyse de sentiments constitue aujourd'hui un outil stratégique incontournable dans un monde où les données textuelles prolifèrent à un rythme exponentiel, notamment à travers les réseaux sociaux, les forums, les avis en ligne et les services de messagerie. En transformant ces données brutes en insights exploitables, elle permet aux entreprises, aux institutions et aux chercheurs de mieux comprendre les perceptions, les émotions et les attentes des individus.

Cette capacité à capter en temps réel l'opinion publique ou la satisfaction client offre un avantage concurrentiel majeur, en facilitant la prise de décision, l'adaptation des produits ou services, et le renforcement des relations avec les utilisateurs.

Malgré les défis persistants tels que la complexité du langage naturel, les nuances culturelles, le sarcasme, l'ironie, ou encore la polysémie des mots l'analyse de sentiments continue de progresser grâce aux avancées constantes en intelligence artificielle, en particulier dans les domaines du traitement automatique du langage naturel (TALN) et de l'apprentissage profond. Les modèles préentraînés de type transformer, comme BERT, RoBERTa ou DeBERTa, ont radicalement amélioré la précision des prédictions, en capturant plus finement le contexte et les subtilités linguistiques.

À mesure que ces technologies deviennent plus accessibles et plus performantes, l'analyse de sentiments ne se limite plus à de simples dichotomies entre opinions positives et négatives. Elle tend désormais vers une compréhension plus fine et nuancée des émotions humaines, ouvrant la voie à des applications encore plus riches : détection du sarcasme, analyse des intentions, anticipation des comportements, ou encore surveillance de la réputation en ligne.

En somme, bien qu'encore perfectible, l'analyse de sentiments est promise à un avenir prometteur, à la croisée de la linguistique, des sciences sociales et de l'intelligence artificielle. Elle s'impose comme un champ d'étude et d'application à fort potentiel, dont l'impact continuera de croître dans les années à venir.

# Chapitre 2

# État de l'art : Analyse de Sentiments

## 2.1 Introduction

L'analyse de sentiments, ou "opinion mining", est une tâche du traitement automatique du langage naturel (TALN) qui vise à identifier et classifier les émotions ou opinions exprimées dans des textes. Elle s'applique à plusieurs domaines comme la politique, la santé, le marché boursier et les médias sociaux. Les niveaux d'analyse incluent :

| **Niveau d'analyse** | **Description** |
|---|---|
| Document Level | Polarité globale du texte. |
| Sentence Level | Classification par phrase. |
| Phrase Level | Évaluation au niveau des clauses. |
| Aspect Level | Extraction et classification liées à des aspects spécifiques. |

Table 2.1 – Niveaux d'analyse des sentiments

## 2.2 Méthodes de l'Analyse de Sentiments [10]

### 2.2.1 Approche Lexicale

Cette méthode se base sur des dictionnaires, on associe à chaque mot un score de polarité (positif, négatif ou neutre). Le sentiment global est calculé en additionnant les scores des mots individuels.

| **Avantages** | **Inconvénients** |
|---|---|
| •Pas besoin de données étiquetées.<br>•Rapide et adapté aux petits jeux de données. | •Limitée par le contexte linguistique.<br>•Incapable de détecter le sarcasme ou l'ironie. |

Table 2.2 – Résumé des avantages et inconvénients de l'approche lexicale

### 2.2.2 Approche Machine Learning

Les textes sont classés par les algorithmes d'apprentissage automatique en catégories de sentiments basées sur des données étiquetées.

Algorithmes courants :

- **Naive Bayes :** Modèle probabiliste rapide mais sensible aux classes déséquilibrées.
- **SVM :** Très efficace pour les classifications binaires, coûteux en calculs.
- **Logistic Regression :** Interprétable, gère bien les problèmes multi-classes.
- **Decision Trees :** Facile à interpréter, sujet à l'overfitting.
- **KNN :** Simple mais coûteux .

### 2.2.3 Approche Hybride

Cette approche combine les méthodes lexicales et celles d'apprentissage automatique.

| **Avantages** | **Inconvénients** |
|---|---|
| • Améliore la précision.<br>• Flexible face aux défis spécifiques. | • Complexité accrue. |

TABLE 2.3 – Résumé des avantages et inconvénients de l'approche hybride

### 2.2.4 Réseaux de Neurones et Apprentissage Profond

Les modèles d'apprentissage profond, comme **RNN**, **LSTM**, **BERT** et **GPT**, capturent les dépendances séquentielles et contextuelles.
Modèles Clés :

- **RNN**[11] : Modèle de Deep Learning dédié à l'analyse et au traitement des données séquentielles, prenant en compte les résultats des éléments précédents.
- **LSTM**[12] : Type particulier de RNN conçu pour surmonter les problèmes de disparition et d'explosion des gradients, avec une meilleure capacité de mémoire à long terme.
- **GRU**[13] : Variante simplifiée et plus rapide des LSTM, conservant les dépendances à long terme.
  — Capture efficacement les dépendances temporelles.
- **BERT**[14] : Modèle de traitement du langage naturel bidirectionnel, analysant le contexte à partir des mots précédents et suivants.
  — Pré-entraîné et performant pour les tâches de compréhension du langage.
- **GPT**[15] : Modèle neuronal basé sur les Transformers, orienté génération de texte.
  — Modèle génératif pré-entraîné.
- **XLNet**[16] : Extension de Transformer-XL, utilisant une méthode autorégressive pour apprendre les contextes bidirectionnels par permutations de séquences.

- **RoBERTa-base**[18] : Variante optimisée de BERT avec un pré-entraînement plus long, sans prédiction de phrase suivante et avec masquage dynamique.
  — 12 couches, 768 dimensions, 12 têtes d'attention, 125M de paramètres.

- **DeBERTa**[20] : Amélioration des Transformers avec attention désentrelacée (contenu/position), offrant de meilleures représentations sémantiques.
  — Performant en compréhension, analyse de sentiments et classification.

### 2.2.5 Analyse de Sentiments Basée sur les Aspects (ABSA)

L'ABSA (Aspect-Based Sentiment Analysis) consiste à identifier les aspects spécifiques mentionnés dans un texte, puis à leur associer une polarité (positive, négative ou neutre).

**Principales étapes :**

- **Détection des aspects** : repérage des éléments ou caractéristiques évoqués (ex. : qualité du son, autonomie, service client).

- **Classification de la polarité** : détermination du sentiment exprimé pour chaque aspect détecté.

- **Agrégation des résultats** : synthèse des polarités à travers les différents aspects pour obtenir un aperçu global.

**Défis majeurs :**

- **Détection des aspects implicites** : certains aspects ne sont pas nommés explicitement mais sous-entendus.

- **Dépendance au contexte** : un même mot peut refléter des sentiments différents selon le contexte ou l'aspect concerné.

### 2.2.6 Transfert d'Apprentissage

Le transfert d'apprentissage consiste à exploiter des modèles pré-entraînés (comme BERT ou GPT) et à les adapter à des tâches spécifiques via un ajustement appelé *fine-tuning*.

**Applications typiques :**

- **Classification inter-domaines** : appliquer un modèle entraîné dans un domaine (par ex. santé) à un autre (par ex. finance).

- **Traitement des langues à ressources limitées** : transférer les connaissances d'une langue riche vers une langue peu dotée.

**Avantages :**

- Réduction du besoin en données annotées, souvent coûteuses et difficiles à obtenir.

- Meilleures performances grâce à la réutilisation des connaissances apprises sur de vastes corpus.

### 2.2.7 Analyse Multimodale de Sentiments (MSA)[34]

L'analyse des sentiments multimodale (MSA) désigne une approche consistant à combiner les informations issues de différentes modalités : texte, image et audio afin de mieux détecter les émotions humaines.

## 2.3 Métriques d'évaluation des Modèles [17]

### 2.3.1 Matrice de Confusion

| **Préduit Réel** | **Positif** | **Négatif** |
|---|---|---|
| **Positif** | TP | FP |
| **Négatif** | FN | TN |

TABLE 2.4 – Matrice de confusion

Où :

- **TP** : Vrais Positifs (True Positives)
- **FP** : Faux Positifs (False Positives)
- **FN** : Faux Négatifs (False Negatives)
- **TN** : Vrais Négatifs (True Negatives)

### 2.3.2 Métriques Communes

- **Exactitude (Accuracy) :** Proportion d'instances correctement classifiées.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Totale.des.échantillons}} \tag{2.1}$$

- **Précision (Precision) :** Rapport entre les vrais positifs et tous les positifs prédits.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

- **Rappel (Recall) :** Rapport entre les vrais positifs et tous les positifs réels.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

- **Score F1 (F1 Score) :** Moyenne harmonique entre précision et rappel.

$$\text{F1}\mathit{Score} = \frac{\text{Precision * Rcall}}{\text{Precision * Recall}} \tag{2.4}$$

- **Spécificité (Specificity) :** Rapport entre les vrais négatifs et tous les négatifs réels.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.5}$$

- **ROC-AUC :** L'aire sous la courbe ROC (AUC) représente la probabilité que le modèle, s'il reçoit un exemple positif et négatif choisi au hasard, classera le positif plus haut que le négatif.

## 2.4 Travaux pertinents

Les modèles de transformeurs ont été utilisés dans diverses recherches pour l'analyse des sentiments basée sur l'aspect.

### 2.4.1 Explainable Aspect-Based Sentiment Analysis Using Transformer Models [18]

*Isidoros Perikos* et *Athanasios Diamantopoulos* [18] proposent une étude qui combine fine-tuning de modèles pré-entraines et techniques d'explicabilité.
Elle fait la comparaison de plusieurs modeles dont *BERT ( Base et Large)*, *RoBERTa ( Base et Large)* , *ALBERT ( base et Large)*, *DistilBERT*, *XLNet* et G*PT ( 3.5 Turbo et 4 )*.
Pour la prise de décision, ils ont applique les methodes *LIME* et *SHAP*. *RoBERTa-Base* l'emporte avec les meilleures performances sur les *datasets MAMS* et *SemEval*.

#### Jeux de données utilisés

*Isidoros Perikos* et *Athanasios Diamantopoulos* [18] ont utilisés plusieurs ensembles de données étiquetés pour mener cette analyse.
Chaque ensemble de données comprend des aspects spécifiques ainsi que leurs polarités respectives. Plus précisément, six ensembles de données largement utilisés ont été sélectionnés :

| **Ensemble de données** | **Nombre de phrases** | **Nombre d'aspects** | **Description** |
|---|---|---|---|
| MAMS | 4297 | 1186 | Ensemble de données multi-aspects à grande échelle. Chaque phrase contient au moins deux aspects avec des polarités différentes. |
| SemEval 2014 Task 4 | 3845 (ordinateurs) + 3041 (restaurants) | - | Contient des critiques d'ordinateurs portables et de restaurants, annotées avec des aspects et leur polarité (positive, négative, neutre). |
| SemEval 2015 Task 12 | 2541 | - | Se concentre sur les avis de restaurants, abordant des aspects tels que la nourriture, le service, l'ambiance, le prix et l'emplacement. |

| Ensemble de données | Nombre de phrases | Nombre d'aspects | Description |
|---|---|---|---|
| SemEval 2016 Task 5 | 3500 | - | Analyses des sentiments des avis de restaurants, comprenant des annotations détaillées sur chaque aspect mentionné. |
| ACOS | - | - | Étend SemEval en ajoutant des annotations pour les aspects implicites, opinions implicites et quadruples. Inclut aussi des avis Amazon sur les ordinateurs portables. |
| Naver Labs | 1006 (échantillon de 585 avis) | - | Avis d'utilisateurs de Foursquare sur les restaurants, annotés selon les directives de SemEval 2016. |

**Combinaison des ensembles de données**

Ces six ensembles ont été fusionnés pour créer un corpus plus vaste comprenant :

- 16 100 phrases de texte

- 30 813 aspects au total

  - 14 455 polarités positives
  - 9 261 polarités négatives
  - 7 097 polarités neutres

**Processus de fine-tuning des modèles :**

- **Fractionnement des données :** 70 % entrainement, 15 % validation et 15 % test.
- **Tokenisation spécifique :** utilisation des tokens spéciaux pour séparer le texte de l'aspect.
- **Hyperparamètres optimaux :**
  Learning rate : 2 × 10-5.
  Batch size : 16.
  Nombre d'époques : 15.

Fonction de perte : Cross-entropy.
Optimiseur : AdamW.

**Techniques d'Explicabilité**

- **LIME :** Il évalue l'impact de chaque mot pour la prise de décision.
- **SHAP :** Identification des mots ayant un poid dans la prédicition.
- **Attention Weights :** Montre les relations entre mots dans l'analyse.
- **Integrated Gradients :** Analyse les gradients afin de mesurer l'importance de chaque mot.
- **Grad-CAM :** Il visualise les sections de textes qui ont le plus de poids sur la classification.

**Résultats des Performances**

On compare les méthodes par rapport à MAMS et SemEval Datasets :

| Modèle | Exactitude | Précision | Rappel | Score F1 |
|---|---|---|---|---|
| **RoBERTa-base** | **89.16%** | **87.94%** | **88.23%** | **88.08%** |
| RoBERTa-large | 88.6% | 87.65% | 87.63% | 87.63% |
| XLNet-base-cased | 88.68% | 87.31% | 87.72% | 87.51% |
| BERT-base-uncased | 86.64% | 85.6% | 85.06% | 85.30% |
| BERT-large-uncased | 87.14% | 86.31% | 84.62% | 85.32% |
| DistilBERT-base-uncased | 85.95% | 84.94% | 84.21% | 84.55% |
| ALBERT-base-v1 | 85.49% | 85.10% | 83.31% | 84.00% |
| ALBERT-large-v1 | 86.38% | 85.73% | 84.31% | 84.80% |
| Bi-LSTM | 73.18% | 71.65% | 69.83% | 70.55% |

TABLE 2.6 – Performances des modèles sur la tâche d'analyse de sentiments par rapport à MAMS et SemEval Datasets

On compare les méthodes par rapport à Naver Labs Europe Dataset

| Modèle | Exactitude | Précision | Rappel | Score F1 |
|---|---|---|---|---|
| **RoBERTa-base** | **97.62%** | **93.30%** | **95.77%** | **94.48%** |
| RoBERTa-large | 97.62% | 91.67% | 98.65% | 94.77% |
| XLNet-base-cased | 94.44% | 89.47% | 82.43% | 85.48% |
| BERT-base-uncased | 96.03% | 89.72% | 91.98% | 90.80% |
| BERT-large-uncased | 95.24% | 88.65% | 88.65% | 88.65% |
| DistilBERT-base-uncased | 96.83% | 94.83% | 89.55% | 91.96% |
| Bi-LSTM | 87.30% | 67.52% | 61.08% | 63.16% |

TABLE 2.7 – Performances des modèles sur la tâche d'analyse de sentiments par rapport à Naver Labs Europe Dataset

### 2.4.2 January 6th on Twitter : measuring social media attitudes towards the Capitol riot through unhealthy online conversation and sentiment analysis [19]

L'étude de *Erik-Robert Kovacs*, *Liviu-Adrian Cotfas* et *Camelia Delcea* [19] permet d'analyser les conversations sur Twitter à propos de l'émeute du Capitole du 6 janvier 2021 en détectant les conversations toxiques et l'analyse des sentiments. Elle vise à évaluer la polarisation politique et les émotions dominantes dans ces échanges.

#### Méthodologies

- **Collecte et prétraitement des données :** 1.3 Million de tweets du dataset «Election2020» ont été extrait, les tweets non anglais qui valent $\tilde{1}2\%$ ont été supprimés et la taille initiale des tweets a été réduite à 3.5% pour faciliter les calculs.

- **Détection des conversations toxiques :** entrainement du modèle XLNet sur le dataset annoté Unhealthy Comment Corpus (UCC) qui contient des commentaires toxiques issus de plateformes en ligne, et classification des tweets soit en « sains » et « toxiques » divises en 7 sous-catégories qui sont : Hostile, antagoniste, condescendant, sarcastique, généralisation, généralisation injuste et dédaigneux.

- **Analyse des émotions et des sentiments :** la détection des émotions et l'analyse fine ont été utilisées comme types d'analyse. NRClex a été adopte pour l'identification de 8 émotions : peur, colère, anticipation, confiance, surprise, tristesse, dégout et joie.

- **Modélisation des sujets :** l'extraction des sujets dominants dans les tweets toxiques a été faite en appliquant Latent Dirichlet Allocation (LDA), ensuite des émotions et des sujets détectés ont été associes pour analyser les tendances discursives.

#### Résultats des Performances

On compare les resultats par rapport au dataset Election2020.

| Modèle | RoBERTa-base | BERT | SVM | XLNet | Random Forest | Logistic Regression |
|---|---|---|---|---|---|---|
| **ROC-AUC** | 0.8026 | 0.7745 | 0.6602 | **0.8075** | 0.6802 | 0.7068 |

TABLE 2.8 – Performances des modèles selon la métrique ROC-AUC

### 2.4.3 A comparative analysis of transfer learning models on suicide and non-suicide textual data [20]

L'étude de *Merinda Lestandy*, *Abdurrahim*, *Amrul Faruq* et *Muhammad Irfan* [20] explore l'utilisation des modèles de transfert d'apprentissage pour détecter les textes liés au suicide sur Reddit. L'objectif est d'améliorer l'identification des tendances suicidaires à partir de messages textuels.

**Méthodologies**

- **Collecte et prétraitement des données :** Le dataset est composé de 232 074 posts provenant de Reddit, répartis en deux catégories : suicide et non-suicide. Les données ont été nettoyées en supprimant les balises HTML, les caractères spéciaux, la ponctuation et les stopwords. Une lemmatisation a également été appliquée.

- **Modèles de classification utilisés :** Quatre modèles de deep learning basés sur des transformers ont été testés :

  — BERT (Bidirectional Encoder Representations from Transformers)

  — RoBERTa (Robustly Optimized BERT Approach)

  — ALBERT (A Lite BERT)

  — DeBERTa (Decoding-enhanced BERT with Disentangled Attention)

  Ces modèles ont été entraînés sur le dataset de Reddit et évalués avec des métriques standards : accuracy, precision, recall et F1-score.

**Résultats des performances**

Les résultats des modèles sont comparés en termes d'accuracy, precision, recall et F1-score :

| Modèle | Exactitude | Précision | Rappel | Score F1 |
|---|---|---|---|---|
| **BiLSTM+GloVe** | 93.46% | 95.32% | 91.42% | 93.32% |
| **BERT** | 98.00% | 98.00% | 98.00% | 98.00% |
| **RoBERTa** | 98.34% | 98.34% | 98.34% | 98.34% |
| **ALBERT** | 95.00% | 95.00% | 95.00% | 95.00% |
| **DeBERTa** | **98.70%** | **98.70%** | **98.70%** | **98.70%** |

TABLE 2.9 – Performances des modèles en classification de textes suicidaires

**Analyse des résultats**

- **DeBERTa** a obtenu les meilleures performances avec une précision de **98.70%**, surpassant tous les autres modèles.

- **RoBERTa** et **BERT** suivent de près, mais avec une légère baisse de performance par rapport à DeBERTa.

- **ALBERT** est le moins performant, probablement en raison de son architecture allégée.

- **BiLSTM (un modèle non-transformer)** est largement moins performant que les modèles de transfert d'apprentissage.

## 2.5 Conclusion

L'analyse de sentiments a connu des avancées majeures grâce aux modèles de deep learning et au transfert d'apprentissage, permettant une compréhension plus précise du langage humain. Les travaux étudiés confirment que certaines architectures surpassent les autres selon le contexte, les objectifs spécifiques ou la complexité des données :

- **RoBERTa-Base** s'est imposé comme le modèle le plus performant pour l'analyse de sentiments basée sur les aspects, notamment sur les jeux de données MAMS et SemEval [20]. Sa capacité à capturer finement le contexte et à apprendre sur des corpus riches et variés en fait un excellent choix pour l'analyse fine de la polarité des opinions.

- **XLNet**, quant à lui, a obtenu les meilleurs résultats dans le cadre de la détection de conversations toxiques sur Twitter. Il a surpassé RoBERTa et BERT en termes de ROC-AUC [21], notamment grâce à sa méthode d'entraînement autoregressive permutée, qui lui confère une meilleure modélisation des dépendances à long terme. Il s'avère donc particulièrement adapté à la classification de textes complexes et ambigus.

- **DeBERTa** a démontré une précision remarquable (jusqu'à 98,70 %) sur des données sensibles, comme le dataset Reddit portant sur la classification entre contenu suicidaire et non-suicidaire. Cette performance exceptionnelle s'explique par sa capacité à dissocier les représentations de contenu et de position, ce qui lui permet de modéliser avec finesse les relations sémantiques implicites entre les mots.

# Chapitre 3

# Méthodologie

## 3.1 Introduction

Dans ce chapitre, nous analysons trois des modèles les plus performants de l'état de l'art : **XLNet**, **RoBERTa-base** et **DeBERTa**.

Pour ce faire, nous les testons sur un jeu de données existant, Twitter Sentiment Analysis, en comparant leurs performances à travers des métriques standard (accuracy, précision, rappel, F1-score), dans le but d'identifier le modèle le plus efficace.

Dans un second temps, le modèle sélectionné sera appliqué à un jeu de données personnalisé, intégrant des phénomènes linguistiques complexes tels que le sarcasme et la double négation, afin d'évaluer sa capacité à gérer des subtilités sémantiques plus difficiles.

## 3.2 Test Préliminaire sur un Dataset Existant (Twitter Sentiment Analysis)

### 3.2.1 Objectif du Test Préliminaire

Dans le deuxième chapitre, nous avons défini les métriques qui permettent d'évaluer la performance d'un modèle.

L'objectif de ce test est de comparer la robustesse de XLNet, RoBERTa-base et DeBERTa sur le dataset TWITTER SENTIMENT ANALYSIS par rapport à ces métriques (Accuracy, Precision, Recall, F1-score) et ainsi identifier la meilleure alternative.

### 3.2.2 Présentation du Dataset

Twitter Sentiment Analysis est un jeu de données étiqueté conçu pour l'analyse de sentiments au niveau des entités et continent des tweets en plusieurs langues. Il est constitué des classes principales suivantes :

- **Positive :** Pour les tweets qui expriment les opinions favorables.
- **Négative :** Pour les tweets qui expriment les opinions défavorables.

- **Neutral :** Pour les tweets neutres.
- **Irrelevant :** Pour exprimer les tweets non pertinents bien qu'ils appartiennent à la classe "Neutral".

Ce Dataset est composé de deux fichiers :

- **Twitter_training.csv :** qui est un ensemble entraînement.
- **Twitter_validation.csv :** qui est un ensemble de validation.

Chaque donnée est sous forme de ligne contenant :

- **Tweet ID :** identifiant unique du tweet.
- **Entity :** le sujet du tweet.
- **Sentiment :** l'étiquette ou la classe associée au tweet.
- **Tweet content :** le texte du tweet.

**Exemple de données dans le Dataset :**

| Tweet ID | Entity | Sentiment | Tweet Content |
|---|---|---|---|
| 2401 | Borderlands | Positive | I'm getting on Borderlands and I will murder you all. |
| 5123 | Microsoft | Negative | I hate how Microsoft keeps pushing updates. |
| 7812 | Twitter | Neutral | Twitter changed its UI again. |

TABLE 3.1 – Exemples de tweets annotés avec des sentiments

### 3.2.3 Expérimentation sur le Dataset Existant

Dans cette partie, nous détaillons l'expérimentation réalisée pour effectuer la comparaison entre les trois modèles afin de déterminer le plus efficace avant de l'utiliser sur un jeu de données que nous allons concevoir.

**Prétraitement des Données :** Dans le but de garantir la qualité des données d'entrainement, plusieurs étapes de prétraitement ont été suivies :

- **Chargement et sélection de données :** Récupération des colonnes pertinentes contenant le texte des tweets et leur annotation.
- **Encodage des labels :** Transformation des annotations textuelles en valeurs numériques.
- **Nettoyage des tweets :** Élimination des caractères spéciaux, des URL et des valeurs manquantes.

- **Tokenisation :** La conversion des tweets en séquences de tokens peut être réalisée en utilisant des tokenizers adaptés à chaque modèle.

- **Partitionnement du Dataset :** échantillonnage de 50 000 tweets pour l'entraînement et 999 tweets pour l'évaluation.

**Configuration des Modèles et Entraînement :** Les trois modèles ont été entraines de manière similaire en utilisant les mêmes paramètres :

- **Couche de classification :** une couche de sortie conçue pour une classification en 4 classes.

- **Optimiseur :** AdamW

- **Fonction de perte :** CrossEntropyLoss

- **Taux d'apprentissage :** 2e-5

- **Taille du batch :** 16

- **Nombre d'époques :** 5

- **Utilisation du processeur graphique** (GPU ou Graphics Processing Unit).

**Évaluation des Performances :** Suite à l'entraînement, chaque modèle a été évalué en utilisant les métriques mentionnées dans les données de validation : Accuracy, Precision, Recall et F1-score et la Matrice de confusion.

**Visualisation des Résultats :**

- Les matrices de confusion ont été générées afin d'analyser les erreurs de classification entre les différentes catégories (Negative, Neutral, Positive et Irrelevant).

- Une analyse détaillée des performances des modèles est présentée dans la section suivante.

### 3.2.4 Résultats et Interprétation

Nous présentons ci-dessous les résultats de la comparaison de XLNet, RoBERTa-base et DeBERTa ainsi que leurs matrices de confusion :

| Modèle | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| **RoBERTa-base** | **95.30%** | **95.36%** | **95.30%** | **95.31%** |
| XLNet | 92.40% | 92.59% | 92.40% | 92.41% |
| DeBERTa | 94.50% | 94.62% | 94.50% | 94.49% |

TABLE 3.2 – Performances des modèles sur l'ensemble de test

**Matrices de Confusion :**

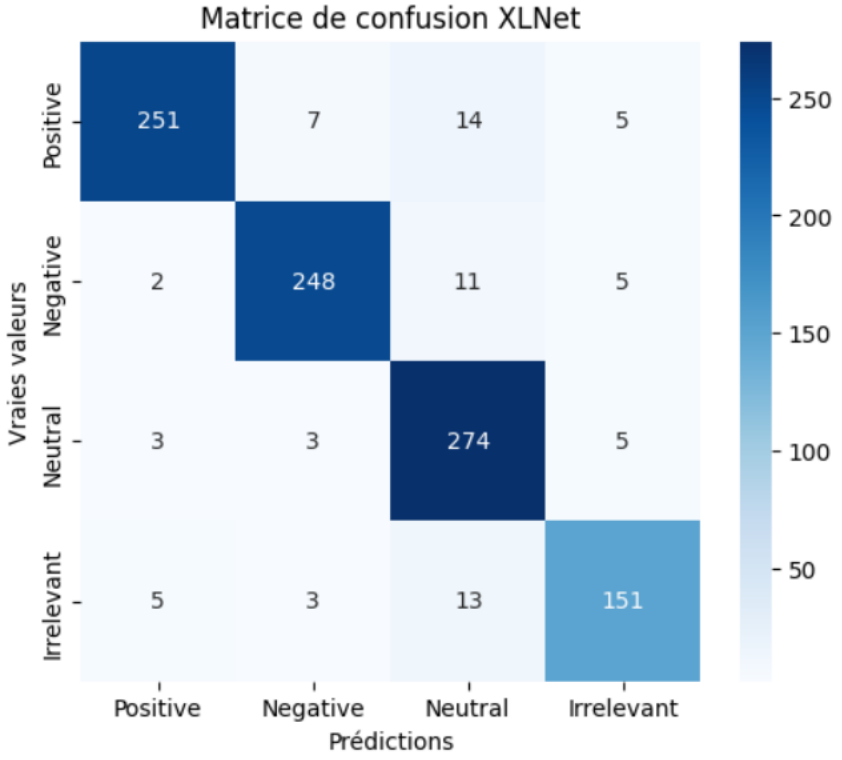


FIGURE 3.1 – Matrice de confusion du modèle XLNet

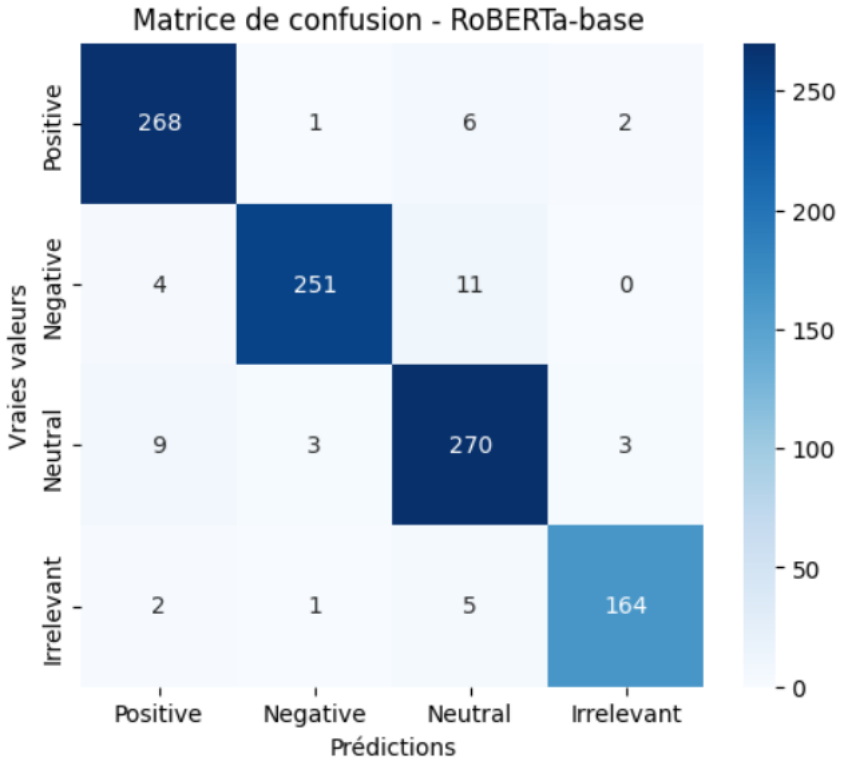


FIGURE 3.2 – Matrice de confusion du modèle RoBERTa-Base

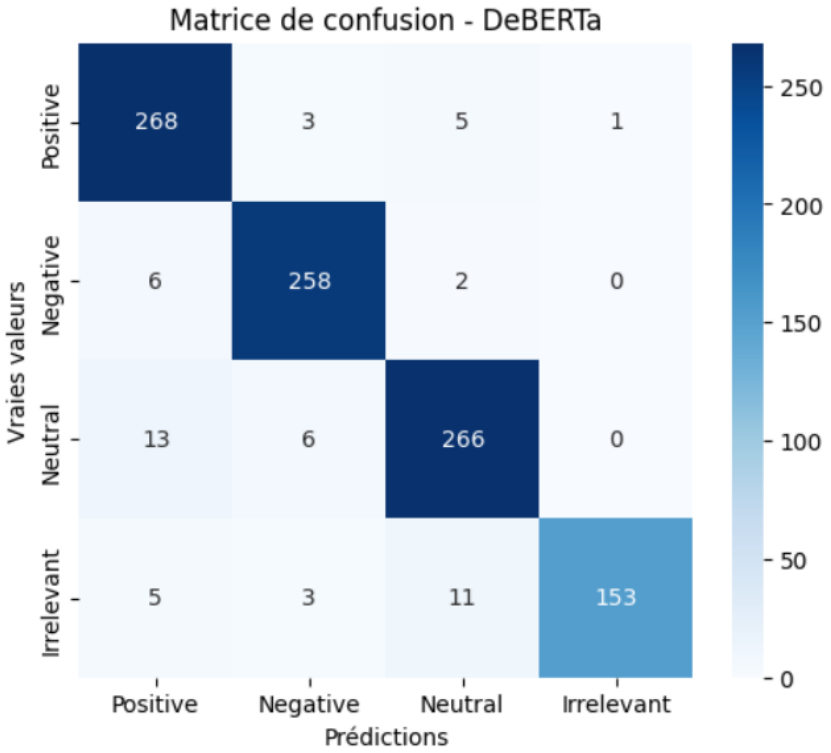


FIGURE 3.3 – Matrice de confusion du modèle DeBERTa

**Analyse des résultats :** On constate que RoBERTa-base obtient les meilleurs résultats avec une exactitude de 95.30% par rapport à XLNet qui a donné 92.40% et DeBERTa 94.50%.
En se basant sur les matrices de confusions :

- **XLNet :**

  · Il montre une forte confusion entre classes, notamment avec la classe "Irrelevant" qui est souvent mal classée.

  · Le nombre de faux positifs et faux négatifs est plus élevé, expliquant pourquoi son recall et précision sont plus bas.

  · Il pourrait être moins robuste aux données déséquilibrées.

- **DeBERTa :**

  · Moins d'erreurs que XLNet mais encore un peu plus que RoBERTa.

  · La confusion entre classes est réduite, mais certaines classes spécifiques peuvent être mal différenciées.

  · Sa précision et son recall proches de 94.5 % suggèrent un bon équilibre entre faux positifs et faux négatifs.

- **RoBERTa-base :**

· Meilleure séparation des classes : le taux de vraies classifications est plus élevé.

· Moins de confusion dans la matrice, ce qui se traduit par de meilleurs scores sur toutes les métriques.

· Génère moins de faux négatifs et faux positifs, ce qui signifie qu'il est plus stable et fiable pour la tâche.

**Choix du modèle :**
En se basant sur les résultats, on retient **RoBERTa-base** comme modèle principal.

## 3.3 Expérimentations de RoBERTa-base sur un autre ensemble de données

Cette section a pour but de présenter et d'analyser les résultats de l'application de RoBERTa-base sur un dataset qui intègre deux grands défis linguistique complexes : le sarcasme et la double négation.

Cette expérimentation vise à évaluer les performances du modèle et sa capacité à classer correctement les sentiments issus des réseaux sociaux.

On a opté pour le modèle RoBERTa-base en raison de sa compétence démontrée à saisir les nuances du langage naturel, dû à son apprentissage sur un vaste corpus de textes.

Néanmoins, les difficultés liées au sarcasme et à la double négation nécessitent une analyse approfondie pour évaluer si ce modèle est en mesure de généraliser ces compétences de manière efficace.

### 3.3.1 Description du Dataset

#### Présentation du dataset

Afin d'évaluer les performances du modèle RoBERTa-base sur des textes contenant du sarcasme et de la double négation, nous avons construit un dataset personnalisé, après avoir constaté l'absence d'un corpus public combinant ces deux phénomènes linguistiques. Ce dataset final est composé de données issues de trois sources différentes :

- **Pour le sarcasme :**
  - **Le dataset de sarcasme disponible sur Kaggle :** Sarcasm Dataset (Dan Ofer), composé de commentaires issus de Reddit annotés( annotation binaire : sarcastique/ non sarcastique).

— **Le dataset d'émotions textuelles :** Text Dataset for Text Emotion Detection, dont une partie des exemples contient un ton ironique ou sarcastique.

- **Pour la double négation :**

— Un dataset de 10 000 phrases collectées manuellement, dont 5 000 exemples contenant une double négation annotés avec l'étiquette 1, et 5 000 exemples sans double négation annotés avec 0.

**Méthodologie de nettoyage**

- **Filtrage des données :** sélection et consolidation des exemples pertinents issus des trois sources, en veillant à équilibrer les occurrences de sarcasme et de double négation.

- **Nettoyage linguistique :** On a supprimé tous les caractères spéciaux et mis en minuscule afin d'adapter les données.

Après agrégation, nettoyage et filtrage, nous avons retenu uniquement les textes qui contiennent une des deux caractéristiques linguistiques cibles, pour le sarcasme nous avons retenue 23715 données et pour la double négation 10000 données, afin de faciliter une évaluation ciblée, équilibrée et pertinente.

**Objectifs et applications**

Le but de ce dataset est de :

- Évaluer les performances du modèle RoBERTa-base sur des données qui contiennent sarcasme et doubles négations.

- Analyser les défis liés à la détection automatique du sarcasme et de la double négation en analyse de sentiments.

- Étudier les faiblesses des modèles classiques de classification des sentiments face à ces formes linguistiques complexes.

### 3.3.2 Configuration et Entraînement de RoBERTa-base

Dans cette partie, nous détaillons les étapes de configuration de **RoBERTa-base** ainsi que les résultats donnés lors de l'évaluation sur notre dataset.

**Prétraitement des Données :**

Pour garantir la qualité des données d'entraînement et d'évaluation, nous avons appliqué plusieurs étapes de préparation :

- **Chargement et sélection des données :** Extraction des colonnes pertinentes contenant les textes et leurs étiquettes (sarcastique ou présence de la double négation).
- **Tokenisation :** Utilisation du tokenizer roberta-base pour transformer les textes en séquences de tokens, avec padding et troncature activés.
- **Partitionnement du dataset :** Division en deux sous-ensembles, 80 % pour l'entraînement et 20 % pour l'évaluation.

**Configuration du Modèle et Entraînement :**

Le modèle utilisé pour cette expérimentation est RoBERTa-base, un transformeur pré-entraîné adapté à la classification binaire. L'entraînement a été réalisé sur GPU avec les paramètres suivants :

- **Couche de classification :** sortie avec 2 neurones (classification binaire)
- **Optimiseur :** AdamW
- **Fonction de perte :** CrossEntropyLoss
- **Taux d'apprentissage :** 2e-5
- **Taille du batch :** 16
- **Nombre d'époques :** 5
- **Environnement :** Google Colab avec GPU activé
- L'entraînement s'est effectué via une boucle PyTorch, avec évaluation automatique après chaque époque.

### 3.3.3 Analyse des Résultats et Limites

**Résultats**

Les performances du modèle ont été évaluées sur le jeu de validation à l'aide des métriques classiques : Accuracy, Precision, Recall et F1-score. Les résultats sont présentés ci-dessous :

| Métrique | Accuracy | Précision | Rappel | F1-score |
|---|---|---|---|---|
| **RoBERTa-base** | 99.87 % | 99.87 % | 99.87 % | 99.87 % |

TABLE 3.3 – Métriques globales de performance du modèle de classification.

**Matrice de confusion :**

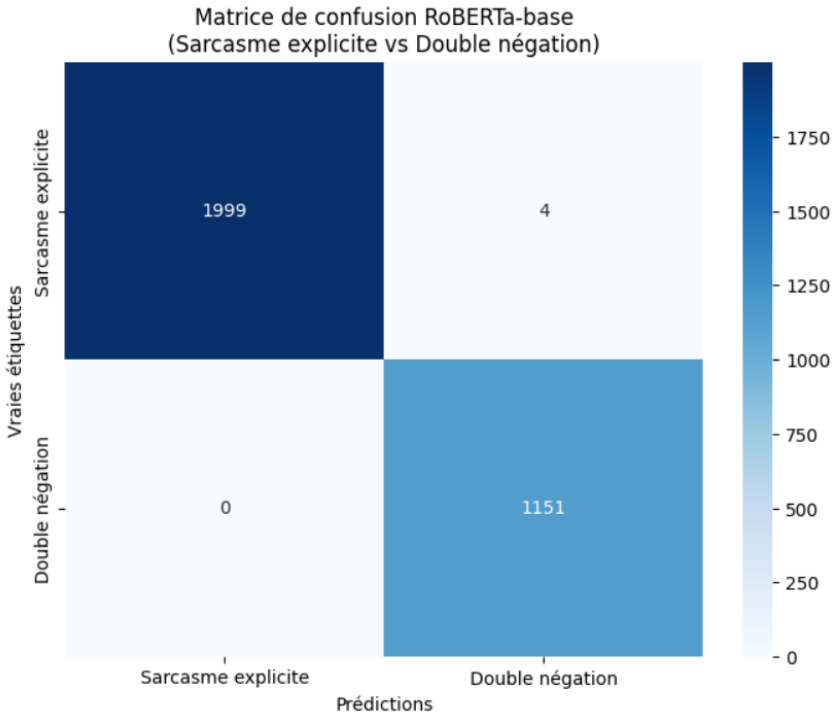


FIGURE 3.4 – Matrice de confusion du modèle RoBERTa-base sur un autre corpus

D'après les résultats ci-dessus, le modèle présente une excellente robustesse et fiabilité pour la classification entre sarcasme explicite et double négation.

Cependant, afin de vérifier si ces bons résultats traduisent réellement une classification des sentiments exprimés et non simplement une reconnaissance textuelle, nous allons effectuer un test qualitatif sur un ensemble de phrases.

### Test qualitatif sur des phrases sarcastiques et ambigües

Afin de tester la robustesse du modèle sur des cas réels, nous avons soumis des phrases incluant du sarcasme explicite et de la double négation :

| **Phrase** | **Classe réelle** | **Classe prédite** |
|---|---|---|
| Oh great, my favorite thing—working late on a Friday ! | Sarcastique | Double négation |
| Yeah, because staying up all night is so healthy. | Sarcastique | Double négation |
| I totally needed my coffee to spill all over my laptop this morning. | Sarcastique | Double négation |
| Perfect ! Another software update right before my presentation. | Sarcastique | Double négation |
| Just what I needed today : more spam emails. | Sarcastique | Double négation |
| Of course I'm thrilled to redo the entire project last minute. | Sarcastique | Double négation |
| Wow, this app crashing for the tenth time is really amazing. | Sarcastique | Double négation |
| Sure, let's have another three-hour meeting about nothing. | Sarcastique | Double négation |
| It's not that I didn't like the idea, but it could be better. | Double négation | Double négation |
| He's not completely unaware of the consequences. | Double négation | Double négation |
| I don't think it's not working. | Double négation | Double négation |
| She never said she wouldn't help. | Double négation | Double négation |
| That's not entirely untrue. | Double négation | Double négation |
| I'm not saying you didn't do well. | Double négation | Double négation |
| They didn't *not* approve it. | Double négation | Double négation |
| I don't dislike this movie. | Double négation | Double négation |

TABLE 3.4 – Exemples de phrases avec leur type réel et la prédiction faite par le modèle RoBERTa-base.

### Analyse de résultats

Bien que les scores globaux du modèle soient presque parfaits (Accuracy, Précision, Rappel et F1-score : 99.87 %), le test qualitatif a révélé des limites importantes dans sa capacité à généraliser à des cas complexes :

Toutes les phrases ont été classées comme "double négation", y compris les 8 phrases sarcastiques, pourtant explicites dans leur formulation.

Cela indique que le modèle ne distingue pas le sarcasme avec des structures syntaxiques négatives, ce qui revient à dire qu'il s'appuie fortement sur des marqueurs linguistiques formels, sans une compréhension réelle contextuelle ou pragmatique.

En conséquence, les bons résultats quantitatifs obtenus en test sont liés à la classification correcte de la polarité (positive/négative) dans un contexte classique, mais pas à la détection correcte du type de sentiment exprimé (ironie, sarcasme, ambiguïté, etc.).

### Limites de RoBERTa-base dans la détection du sarcasme et des doubles négations

L'architecture de **RoBERTa-base**, illustrée dans la figure 3.5, repose sur celle de BERT, à savoir un encodeur Transformer bidirectionnel profond. Elle est composée de :

- Une tokenisation en sous-mots (Byte-Pair Encoding modifié),
- Plusieurs couches (12 pour RoBERTa-base) de self-attention,
- Un position embedding,
- Un mécanisme d'attention multi-têtes permettant de capturer les relations entre tokens.

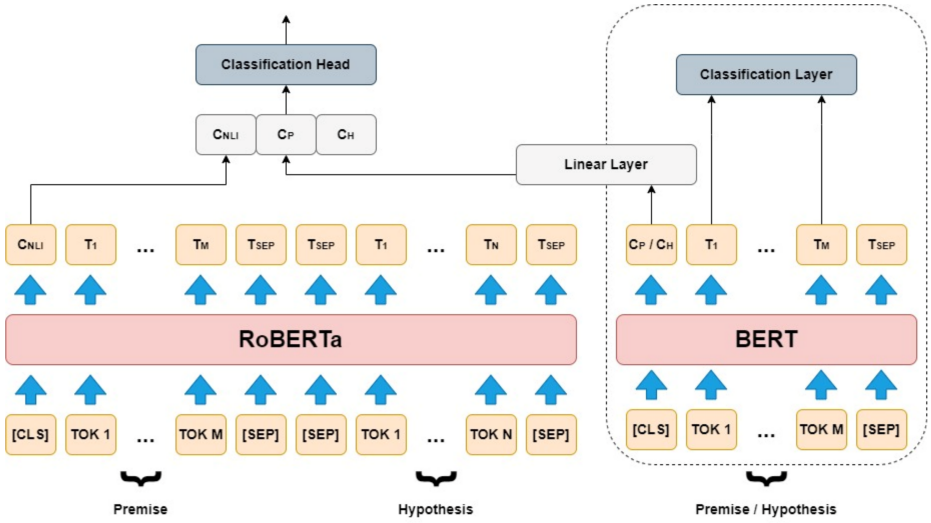


FIGURE 3.5 – Architecture de RoBERTa-base.
[21]

Cette structure permet au modèle d'encoder efficacement le contexte syntaxique et sémantique des mots dans une phrase. Elle excelle dans les tâches classiques de classification ou d'analyse de sentiments explicite.

Cependant, cette architecture présente des limitations structurelles face à des phénomènes plus complexes comme le sarcasme ou les doubles négations. Ces limites s'expliquent par plusieurs facteurs liés directement à la conception du modèle :

### 1. Confusion systématique entre sarcasme et double négation

Lors du test qualitatif sur 16 phrases, dont 8 sarcastiques et 8 contenant des doubles négations, le modèle les a toutes classé comme des doubles négations.

*"Oh great, my favorite thing—working late on a Friday !"*
**Classe réelle :** Sarcastique
**Classe prédite :** Double négation

Cette confusion montre que RoBERTa-base interprète le sarcasme comme de simples doubles négations.

### 2. Absence de compréhension pragmatique

RoBERTa se base uniquement sur les mots vus pendant l'entraînement, et sur les liens qu'il a appris entre eux. Il n'a cependant aucune compréhension du ton employé, de l'intention derrière les mots, ni des contradictions présentes dans une phrase. Par exemple :

*"Of course I'm thrilled to redo the entire project last minute."*
**Classe réelle :** Sarcastique
**Classe prédite :** Double négation

Pour nous les humains, le mot *thrilled (content(e))* est utilisé de manière sarcastique. RoBERTa, en l'absence d'un signal explicite d'ironie, ne remet pas en question la formulation.

**3. Interprétation des doubles négations**

Même pour des phrases qui contiennent des doubles négations, RoBERTa ne semble pas comprendre la logique exprimée, il se contente d'identifier des motifs syntaxiques superficiels :

*"I don't dislike this movie."*
**Classe prédite :** Double négation
**Classe réelle :** Opinion globalement positive (atténuée)

**4. Raisons techniques des erreurs**

- **Tokenisation en sous-mots** :La ségmentation peut casser des expressions toutes faites ou des tournures familières, ce qui rend plus difficile la détection de l'ironie ou du second degré.

- **Pas de modélisation explicite de la logique** : RoBERTa ne possède pas de structure logique déductive pour gérer les doubles négations.

- **Manque d'annotations pragmatiques** : L'absence d'exemples annotés pour des phénomènes comme l'ironie empêche le modèle d'apprendre à les distinguer.

## 3.4 Conclusion

Dans un premier temps, nous avons procédé à une étude comparative entre trois modèles de l'état de l'art : RoBERTa-base, XLNet et DeBERTa sur un dataset existant (Twitter Sentiment Analysis). Cette analyse nous a permis d'évaluer objectivement les performances de chaque modèle sur des données textuelles classiques, à l'aide de métriques standards telles que l'accuracy, la précision, le rappel et le F1-score. Les résultats ont clairement désigné RoBERTa-base comme le modèle le plus performant, tant en termes de précision que de stabilité.

Dans un second temps, nous avons approfondi l'évaluation de RoBERTa-base en l'appliquant à un dataset personnalisé, conçu spécifiquement pour inclure deux phénomènes linguistiques complexes : le sarcasme et la double négation. Si les scores globaux obtenus étaient particulièrement élevés, une analyse qualitative a révélé des limites majeures du modèle dans la compréhension réelle du contenu implicite. Toutes les phrases sarcastiques ont été confondues avec des doubles négations, traduisant une incapacité du modèle à saisir les nuances contextuelles, pragmatiques ou tonales.

Ainsi, cette double expérimentation met en lumière une réalité importante : les modèles préentraînés tels que RoBERTa-base peuvent exceller sur des tâches classiques, mais

peinent à généraliser lorsqu'ils sont confrontés à des subtilités linguistiques plus fines. Ces résultats ouvrent la voie à la conception de modèles plus spécialisés, mieux adaptés aux défis propres à l'analyse de sentiments en contexte réel.

# Chapitre 4

# Amélioration du modèle

## 4.1 Introduction

Dans ce chapitre, nous présentons la mise en œuvre du modèle conçu pour surmonter les limites de RoBERTa-base dans la détection du sarcasme et de la double négation.

L'objectif est d'enrichir un modèle pré-entraîné de type RoBERTa en y intégrant deux mécanismes complémentaires :

- **D'une part,** un système de détection du sarcasme fondé sur l'analyse de l'intention implicite, avec une annotation automatique des niveaux de sarcasme générée par un modèle de langage de type LLM.

- **D'autre part,** une prise en compte explicite des structures syntaxiques liées à la double négation.

## 4.2 Présentation générale du modèle

Cette section présente l'architecture générale des deux modèles développés ainsi que leurs objectifs.

Contrairement aux modèles classiques vus dans les chapitres précédents, notre approche combine apprentissage profond et analyse linguistique pour traiter deux phénomènes complexes : le sarcasme, envisagé ici comme une intention implicite difficilement détectable, dont le niveau est automatiquement annoté à l'aide d'un modèle de langage (LLM) ; et la double négation, qui nécessite une compréhension syntaxique approfondie.

## 4.3 Objectifs du modèle

Le but de cette implémentation est de concevoir un modèle capable de :

- Capturer le sarcasme en identifiant l'intention implicite exprimée dans un énoncé, grâce à une annotation automatique du niveau de sarcasme réalisée par un modèle de langage (LLM).

- Interpréter les doubles négations en s'appuyant sur la structure grammaticale implicite des phrases, capturée automatiquement par le modèle à l'aide de représentations contextuelles (RoBERTa), enrichies par un GRU bidirectionnel et un mécanisme d'attention focalisé sur les marqueurs de négation.

- Exploiter conjointement les dimensions sémantique et syntaxique pour affiner la détection du ton et la compréhension du sens implicite des phrases.

## 4.4 Outils et environnement de développement

Le développement du projet a été réalisé dans un environnement accessible et performant, adapté aux exigences du traitement du langage naturel et à l'entraînement de modèles de deep learning.

Dans le cadre de ce projet, plusieurs outils ont été mobilisés afin de faciliter le développement, les tests et l'évaluation des modèles. Cette section présente l'environnement de travail adopté.

### 4.4.1 Environnement de développement[22]

**Google Colab** (abréviation de Google Colaboratory) est une plateforme gratuite proposée par Google, permettant d'écrire et d'exécuter du code Python directement depuis un navigateur. Elle prend en charge les notebooks Jupyter sans nécessiter d'installation locale ni de configuration logicielle, ce qui en fait un environnement accessible et pratique.

Cette solution facilite également l'accès aux bibliothèques couramment utilisées en apprentissage automatique et offre une interface conviviale pour le prototypage de modèles.

Un des avantages majeurs de Google Colab est la mise à disposition gratuite de processeurs graphiques (GPU) et d'unités de traitement de tenseur (TPU), particulièrement utiles pour les tâches computationnelles intensives telles que l'entraînement de modèles de deep learning.

### 4.4.2 Language de programmation [23]

Python est un langage de programmation informatique généraliste. Contrairement à HTML, CSS ou JavaScript, son usage n'est donc pas limité au développement web. Il peut être utilisé pour tout type de programmation et de développement logiciel.

On s'en sert notamment pour le développement back end d'applications web ou mobile, et pour le développement de logiciels et d'applications pour PC. Il permet également d'écrire des scripts système, afin de créer des instructions pour un système informatique.

Par ailleurs, Python est le langage informatique le plus populaire pour le traitement Big Data, l'exécution de calculs mathématiques ou le Machine Learning.

### 4.4.3 Bibliothèques utilisées

#### Pandas [24]

Pandas est une bibliothèque logicielle open-source et est spécifiquement conçue pour la manipulation et l'analyse de données en langage Python. Elle est à la fois performante, flexible et simple d'utilisation. Grâce à Pandas, le langage Python permet enfin de charger, d'aligner, de manipuler ou encore de fusionner des données.

#### Numpy [25]

Le terme NumPy est en fait l'abréviation de « Numerical Python ». Il s'agit d'une bibliothèque Open Source en langage Python. On utilise cet outil pour la programmation scientifique en Python, et notamment pour la programmation en Data Science, pour l'ingénierie, les mathématiques ou la science.

#### PyTorch [26]

PyTorch est une bibliothèque d'IA qui est créée par Meta. Elle est écrite en Python afin de développer le deep learning with PyTorch ou l'apprentissage profond ainsi que le développement des réseaux de neurones artificiels. En partant de plusieurs variables, vous pouvez effectuer des calculs de gradients ou utiliser des tableaux multidimensionnels obtenus à l'aide de tenseurs. PyTorch est disponible en open source.

#### Transformers [27]

Transformers est une bibliothèque de modèles pré-entraînés de texte, de vision par ordinateur, d'audio, de vidéo et modèles multimodaux pour l'inférence et l'entraînement. On utilise les Transformers pour affiner les modèles sur les données, créer des applications d'inférence et des cas d'utilisation d'IA générative sur plusieurs modalités.

#### Scikit-learn [28]

C'est une librairie Python qui donne accès à des versions efficaces d'un grand nombre d'algorithmes courants. Elle offre également une API propre et uniformisée. Par conséquent, un des gros avantages de Scikit-Learn est qu'une fois que vous avez compris l'utilisation et la syntaxe de base de Scikit-Learn pour un type de modèle, le passage à un nouveau modèle ou algorithme est très simple. La librairie ne permet pas seulement de faire de la modélisation, elle peut assurer également des étapes de preprocessing ce que nous verrons dans la suite de l'article.

#### Random [29]

random est un module Python regroupant plusieurs fonctions permettant de travailler avec des valeurs aléatoires. La distribution des nombres aléatoires est réalisée par le générateur de nombres pseudo-aléatoires Mersenne Twister, l'un des générateurs les plus testés et utilisés dans le monde informatique. Le module comprend plusieurs fonctions travaillant chacune avec un type défini de variables. Ces fonctions peuvent être séparées en trois groupes :

- Celles qui travaillent avec des nombres entiers
- Celles qui travaillent avec des nombres réels
- Celles qui travaillent avec des séquences (par exemple des listes).

**Tqdm** [30]

Tqdm est une bibliothèque Python qui permet de créer des barres de progression pour les boucles. Il est conçu pour être facile à utiliser et à intégrer dans votre code Python existant. « tqdm » signifie « taqadum » en arabe, ce qui signifie « progresser » ou « avancer ».

**Matplotlib** [31]

Matplotlib est une bibliothèque Python capable de produire des graphes de qualité. Elle peut être utilisé dans des scripts Python, le shell Python et IPython, le notebook Jupyter, des serveurs d'application web et dans quatre outils d'interface graphique. Vous pouvez générer des graphes, histogrames, des spectres de puissance (lié à la transformée de Fourier), des graphiques à bares, des graphiques d'erreur, des nuages de dispersion, etc... en quelques lignes de code.

**Seaborn** [32]

Seaborn est une bibliothèque de visualisation de données Python basée sur matplotlib . Elle fournit une interface de haut niveau pour créer des graphiques statistiques attrayants et informatifs.

## 4.5 Détection du sarcasme

### 4.5.1 Introduction

Le sarcasme, en tant que figure de style fondée sur l'ironie et la contradiction entre le sens littéral et l'intention réelle, constitue un défi majeur pour les systèmes de traitement automatique du langage naturel (TALN). Il repose souvent sur des implicites, du second degré ou des renversements sémantiques subtils qui échappent aux approches classiques basées sur des mots-clés ou des règles grammaticales.

Dans le contexte des réseaux sociaux et des textes courts en ligne, la détection automatique du sarcasme devient cruciale, notamment pour des applications telles que l'analyse d'opinion, la modération de contenu ou l'assistance conversationnelle. Ce travail s'inscrit dans cette optique, en proposant une approche complète et moderne pour capturer les nuances sarcastiques dans les textes en anglais.

### 4.5.2 Objectif de la section

Comme mentionné précédemment, notre travail est structuré en deux volets : la détection du sarcasme et celle de la double négation. Cette section est consacrée exclusivement au premier volet, en présentant la méthodologie utilisée pour détecter le sarcasme dans les énoncés textuels.

L'objectif principal est de concevoir un système intelligent capable de prédire le niveau de sarcasme d'un énoncé, en s'appuyant sur la compréhension implicite de son contenu. Contrairement aux approches classiques, notre méthode se base sur la puissance des modèles de langage pré-entraînés pour capter l'intention cachée derrière les mots.

Cette démarche repose d'abord sur une annotation automatique des données à l'aide d'un LLM (Flan-T5-large), guidé par un prompt conçu manuellement. L'ensemble du processus est structuré en cinq étapes :

- **Partie 1 :** Annotation automatique du niveau de sarcasme
- **Partie 2 :** Méthodologie de construction du modèle de prédiction du niveau de sarcasme
- **Partie 3 :** Analyse qualitative des prédictions
- **Partie 4 :** Interprétation du sarcasme : vers une meilleure explicabilité
- **Partie 5 :** Évaluation et analyse des résultats

### 4.5.3 Partie 1 : Annotation automatique du niveau de sarcasme

Pour capturer plus finement l'intention implicite présente dans les textes sarcastiques, nous avons choisi d'effectuer une annotation automatique du niveau de sarcasme à l'aide d'un Large Language Model (LLM). L'objectif est de doter chaque exemple d'un label nuancé compris entre : 0 : pas sarcastique, 1 : sarcasme modéré, 2 : sarcasme exagéré.

#### Méthodologie

L'annotation a été réalisée à l'aide du modèle Flan-T5-large, un LLM pré-entraîné par Google, spécialisé dans les tâches de génération et d'inférence textuelle.

Pour améliorer la compréhension du ton employé, chaque exemple annoté ne se limite pas au commentaire seul, mais inclut également le commentaire précédent (le contexte).

En effet, le sarcasme repose souvent sur une contradiction implicite entre une affirmation et son contexte d'origine. En fournissant la paire parent-comment / comment, on permet au modèle de mieux interpréter la nuance sarcastique et l'intention réelle de l'utilisateur.

L'algorithme du prompt généré est le suivant :

**Algorithm 1** Annotation du niveau de sarcasme à partir d'un commentaire et de son contexte

**Entrée** : *parent* (phrase de contexte), *comment* (réponse à évaluer)
**Sortie** : niveau de sarcasme $s \in \{0, 1, 2\}$, ou $-1$ en cas d'échec

Construire le prompt suivant :
  Tu es un expert en détection du sarcasme.
  Attribue un niveau de sarcasme entre 0 et 2 selon la définition :
  0 = pas sarcastique, 1 = un peu sarcastique (modéré), 2 = assez sarcastique (exagéré).
  Contexte : *parent*
  Réponse : *comment*
  Réponds uniquement par le chiffre correspondant.
Envoyer ce prompt au modèle de génération de texte
Récupérer la réponse générée
Rechercher dans la réponse un chiffre entre 0 et 2
**if** un chiffre valide est trouvé **then**
    **return** le dernier chiffre trouvé
**else**
    **return** $-1$ ▷ Si aucun chiffre détecté
**end if**

Ce prompt est dynamique : pour chaque ligne du dataset, les valeurs réelles de parent-comment et comment sont injectées automatiquement avant d'être transmises au modèle.

### Mise en œuvre technique

L'annotation a été automatisée à l'aide de la bibliothèque Hugging Face Transformers et exécutée sur GPU. Le pipeline utilisé est de type **text2text-generation**, ce qui permet de soumettre un prompt personnalisé au modèle et de récupérer directement une réponse textuelle.
Les étapes sont les suivantes :

- Chargement du fichier CSV contenant les colonnes parent-comment et comment.
- Boucle de traitement ligne par ligne avec tqdm pour le suivi.
- Génération d'un prompt complet pour chaque exemple.
- Passage du prompt au modèle Flan-T5-large.
- Extraction du chiffre retourné à l'aide d'expressions régulières.
- Ajout d'une nouvelle colonne sarcasm-level dans le DataFrame.

Ci-joint l'algorithme complet de l'annotation :

**Algorithm 2** Annotation du niveau de sarcasme avec contexte à l'aide de Flan-T5-Large

**Entrée** : un fichier CSV contenant les colonnes `parent_comment` et `comment`
**Sortie** : un fichier CSV enrichi avec une colonne `sarcasm_level`

Charger le dataset depuis `dataset_sarcasme.csv`
Initialiser le pipeline de génération avec le modèle Flan-T5-Large sur GPU

**function** AnnoterSarcasme(*parent*, *comment*)
    Construire le prompt :
    `Tu es un expert en détection du sarcasme.`
    `Attribue un niveau de sarcasme entre 0 et 2 selon la définition :`
    `0 = pas sarcastique, 1 = un peu sarcastique (modéré), 2 = assez sarcastique (exagéré).`
    `Contexte : "parent"`
    `Réponse : "comment"`
    `Réponds uniquement par le chiffre correspondant.`
    Envoyer le prompt au modèle
    Extraire un chiffre entre 0 et 2 de la réponse générée
    **if** un chiffre valide est trouvé **then**
        **return** ce chiffre
    **else**
        **return** $-1$
    **end if**
**end function**

Initialiser une liste vide `sarcasm_levels`
**for** chaque ligne (*parent*, *comment*) dans le dataset **do**
    *level* $\leftarrow$ AnnoterSarcasme(*parent*, *comment*)
    Ajouter *level* à la liste `sarcasm_levels`
**end for**
Ajouter la colonne `sarcasm_level` au dataset
Sauvegarder le fichier sous `nouveau-dataset.csv`
Afficher le message : « Annotation terminée ! »

### Résultat

Le fichier final contient désormais une annotation automatique du niveau de sarcasme pour chaque commentaire, tenant compte du contexte conversationnel.

Ce nouveau dataset nous permet de reformuler le problème comme une classification à 3 classes, bien plus fine et représentative qu'un simple étiquetage binaire.

### 4.5.4 Partie 2 : Méthodologie de construction du modèle de prédiction du niveau de sarcasme

#### 1. Préparation du corpus d'entraînement pour la prédiction du sarcasme

Après avoir automatiquement annoté le corpus en trois niveaux de sarcasme (0, 1 ou 2) à l'aide du modèle Flan-T5-large, nous avons préparé un ensemble de données structuré pour l'entraînement d'un modèle prédictif.

Pour enrichir la représentation du contexte conversationnel, nous avons fusionné chaque commentaire avec le commentaire parent qui le précède directement dans le fil de discussion. Cette concaténation, séparée par le token spécial [SEP], permet au modèle d'apprendre à détecter des indices subtils de sarcasme dépendant du contexte.

Nous avons ensuite nettoyé les annotations pour garantir leur validité, notamment en remplaçant les éventuelles annotations erronées.

Enfin, les textes ont été transformés en représentations numériques à l'aide du tokenizer de RoBERTa avec un **padding et un troncage** à 128 tokens, afin de garantir une uniformité d'entrée dans le modèle de classification.

#### 2. Préparation des données pour l'apprentissage

Une fois les textes concaténés et tokenisés, nous avons préparé les jeux d'entraînement et de validation nécessaires à l'apprentissage supervisé.

**2.1. Équilibrage des classes :** L'échelle de sarcasme allant de 0 à 2 peut introduire un déséquilibre dans la distribution des exemples.

Pour corriger ce biais et éviter qu'un modèle n'apprenne à favoriser la classe majoritaire, nous avons calculé les poids des classes à l'aide de la fonction compute-class-weight de sklearn.C

Ces poids sont intégrés à la fonction de perte afin de mieux pénaliser les erreurs sur les classes sous-représentées.

**2.2. Découpage du dataset :** Nous avons utilisé la fonction train-test-split avec une séparation stratifiée (stratify=y) afin de préserver la répartition des classes dans les deux sous-ensembles. Le dataset a ainsi été scindé comme suit :

- 80% pour l'entraînement.
- 20% pour la validation.

**2.3. Construction des DataLoaders :** Les entrées tokenisées (input-ids et attention-masks) et les étiquettes (labels) ont été converties en objets TensorDataset, puis encapsulées dans des DataLoader.

Ces derniers permettent une gestion efficace du chargement des mini-batchs :

- RandomSampler pour l'entraînement, afin de mélanger les données à chaque epoch.
- SequentialSampler pour la validation, pour un traitement cohérent.

**3. Architecture du modèle**

Pour la tâche de classification multi-classes (3 classes), nous avons utilisé le modèle RoBERTa-base pré-entraîné par Hugging Face. Plus précisément, nous avons utilisé la classe RobertaForSequenceClassification, qui ajoute une couche de classification linéaire au-dessus du modèle RoBERTa, adaptée aux tâches de classification de séquences.

Nous avons initialisé ce modèle avec l'argument num-labels=3, correspondant aux trois classes de sarcasme : non sarcastique (0), modéré (1) et très sarcastique (2). Pour accélérer l'entraînement, l'ensemble du modèle a été déplacé sur le GPU.

Nous avons ensuite procédé au fine-tuning du modèle, c'est-à-dire à l'adaptation de ce modèle pré-entraîné à notre tâche spécifique. RoBERTa-base, entraîné sur un vaste corpus textuel par Facebook AI, possède déjà une compréhension linguistique générale. Le fine-tuning permet de spécialiser cette compréhension aux nuances propres au sarcasme, en poursuivant l'apprentissage sur notre jeu de données annoté.

Concrètement, tous les poids du modèle, y compris ceux de la couche de classification ajoutée, sont mis à jour durant l'entraînement. Il s'agit donc d'un fine-tuning complet, visant à affiner la capacité du modèle à reconnaître et distinguer les degrés de sarcasme présents dans les énoncés.

### 3.1. Optimisation et stratégie d'entraînement

**Fonction de perte :** Nous avons utilisé la fonction **CrossEntropyLoss**, en lui passant les poids de classes calculés précédemment pour équilibrer l'impact des erreurs entre les niveaux de sarcasme.

**Optimiseur et scheduler :** L'optimiseur utilisé est **AdamW**, recommandé pour les modèles Transformers, avec :

- un taux d'apprentissage (lr) de **2e-5** ;
- un epsilon (eps) de **1e-8**.

Nous avons également utilisé un scheduler linéaire avec **warm-up** à zéro pour ajuster dynamiquement le taux d'apprentissage au fil des itérations. Le nombre total d'étapes d'entraînement est défini par : (nombre de batches par epoch) × (nombre d'epochs).

**4. Entraînement du modèle**

Le modèle a été entraîné sur 5 époques. À chaque epoch, le processus suivant est exécuté : Phase d'entraînement :
**Phase d'entraînement :**

- Le modèle est mis en mode entraînement (model.train()).
- Pour chaque batch, les gradients sont remis à zéro, la prédiction est effectuée, la perte est calculée et les gradients sont rétropropagés.
- Le gradient est ensuite clippé à une norme maximale de 1.0 pour éviter les explosions de gradients.
- L'optimiseur et le scheduler sont appelés à chaque itération.

**Phase de validation :**

- Le modèle passe en mode évaluation (model.eval()).
- Les prédictions sont faites sans calcul de gradient.
- La moyenne de la perte de validation est calculée à la fin de chaque epoch.

**Early Stopping :**

- Un mécanisme d'early stopping a été mis en place pour éviter le surapprentissage. Si la perte de validation ne s'améliore pas pendant 2 époques consécutives (patience=2), l'entraînement s'arrête automatiquement. Chaque fois qu'une meilleure performance est observée sur les données de validation, l'état du modèle est sauvegardé dans un fichier best-model.pt.

**Résumé des paramètres d'entraînement**

| **Paramètre** | **Valeur** |
|---|---|
| Modèle | `RoBERTa-base fine-tuné` |
| Nombre de classes | 3 (sarcasme : 0, 1, 2) |
| Token max par texte | 128 |
| Taille de batch | 16 |
| Nombre d'époch s | 5 |
| Optimiseur | `AdamW` |
| Fonction de perte | `CrossEntropyLoss` (avec poids) |
| Scheduler | Linéaire |
| Early Stopping | Oui (patience = 2) |

TABLE 4.1 – Résumé des paramètres utilisés pour l'entraînement du modèle RoBERTa

L'algorithme est le suivant :

**Algorithm 3** Entraînement de RoBERTa pour la détection de sarcasme

**Entrée** : un fichier CSV contenant les colonnes `parent_comment`, `comment` et `sarcasm_level`
**Sortie** : un modèle RoBERTa entraîné et sauvegardé

Définir une graine aléatoire pour la reproductibilité
Détecter l'appareil (GPU)
Charger le fichier `nouveau-dataset.csv`
Créer la colonne `full_text` en concaténant `parent_comment` + [SEP] + `comment`
Nettoyer les espaces dans `full_text`

Initialiser le tokenizer RoBERTa-base
Tokenizer les textes avec padding, troncature, max_length=128
Extraire `input_ids`, `attention_mask` et convertir les labels en tenseurs

Diviser les données en ensembles d'entraînement et validation (stratification)
Créer les `TensorDataset` pour chaque ensemble

Calculer les poids de classes équilibrés
Initialiser le modèle RoBERTa avec 3 classes
Définir la fonction de perte pondérée `CrossEntropyLoss`
Initialiser l'optimiseur `AdamW` et le scheduler linéaire

Fixer le `batch_size = 16`
Créer les `DataLoader` pour entraînement (aléatoire) et validation (séquentiel)

Initialiser : `best_val_loss` $\leftarrow \infty$, `patience` $\leftarrow 2$, `counter` $\leftarrow 0$
**for** chaque epoch de 1 à 5 **do**
    Passer en mode entraînement
    **for** chaque batch dans les données d'entraînement **do**
        Transférer les données sur l'appareil
        Mettre les gradients à zéro
        Obtenir les prédictions du modèle
        Calculer la perte
        Backpropagation
        Clipper les gradients à 1.0
        Mettre à jour les poids avec AdamW
        Mettre à jour le scheduler
    **end for**
    Calculer la perte moyenne d'entraînement
    Passer en mode évaluation
    Calculer la perte moyenne de validation
    **if** val_loss ¡ best_val_loss **then**
        Mettre à jour `best_val_loss`
        Réinitialiser `counter`
        Sauvegarder le modèle dans `best_model.pt`
    **else**
        Incrémenter `counter`
        **if** `counter` $\geq$ `patience` **then**
            **break** ▷ Early stopping
        **end if**
    **end if**
**end for**

### 4.5.5 Partie 3 : Analyse qualitative des prédictions

**Validation qualitative sur des exemples internes**

Afin de vérifier la cohérence des prédictions et d'évaluer la qualité des annotations, une analyse qualitative a été menée sur un échantillon de 10 exemples issus de l'ensemble de validation.

Chaque exemple est constitué d'une paire parent-commentaire, et est accompagné de l'annotation manuelle ainsi que de la prédiction générée par le modèle fine-tuné.

Les résultats obtenus sont les suivants :

- 7 exemples sur 10 ont été correctement classés, ce qui montre une bonne capacité du modèle à identifier les niveaux de sarcasme.

- 3 exemples présentent une incohérence entre l'annotation et la prédiction, traduisant une certaine difficulté à distinguer des cas aux limites floues entre les niveaux.

Voici quelques cas illustratifs :

- **Texte 3 :** L'annotation "sarcasme modéré" est classée comme "neutre" par le modèle. Cela peut s'expliquer par une formulation polie et ambigüe du commentaire, difficile à interpréter sans signaux sarcastiques explicites.

- **Texte 6 :** Le modèle prédit "sarcasme modéré" alors que le niveau annoté est "sarcasme exagéré". Cette confusion suggère une limite dans la sensibilité du modèle à détecter l'intensité du sarcasme, notamment en l'absence de marques emphatiques.

- **Texte 10 :** L'expression "Good luck with school", perçue comme neutre, est interprétée comme hautement sarcastique. Cette erreur peut être due à une surinterprétation du ton implicite, ce qui reflète les limites du modèle à faire la part entre cordialité sincère et sarcasme implicite.

Globalement, l'analyse qualitative met en évidence une bonne cohérence globale, mais souligne la subtilité et la subjectivité de la tâche, en particulier lorsqu'il s'agit de distinguer le sarcasme modéré du sarcasme exagéré.

Ci joint l'algorithme de cette étape :

**Algorithm 4** Analyse qualitative des prédictions sur l'ensemble de validation

**Entrée** : modèle RoBERTa entraîné, `val_dataloader`, `val_inputs`, dataframe original `df`
**Sortie** : affichage de 10 exemples avec prédictions et commentaires

Charger le modèle sauvegardé `best_model.pt` dans `model`
Passer le modèle en mode évaluation
Définir le dictionnaire `id2label` : 0 → neutre, 1 → sarcasme modéré, 2 → sarcasme exagéré
Fixer le nombre d'exemples à afficher à 10
Initialiser le compteur ← 0
Transférer `val_inputs` sur le même appareil que le modèle

**for** chaque batch dans `val_dataloader` **do**
    Transférer `b_input_ids`, `b_input_mask`, `b_labels` sur l'appareil
    Obtenir les sorties du modèle sur les entrées
    Prédire les classes avec `argmax` sur les logits
    **for** chaque exemple $i$ dans le batch **do**
        **if** compteur $\geq 10$ **then**
            **break**
        **end if**
        Trouver l'indice de l'exemple courant dans `val_inputs` via comparaison des tenseurs
        Récupérer le texte original depuis le dataframe `df`
        Récupérer le vrai label et le label prédit via `id2label`
        **if** le vrai label est différent du label prédit **then**
            Commentaire ← "Incohérence détectée"
        **else**
            Commentaire ← "Conforme"
        **end if**
        Afficher le texte, l'annotation, la prédiction et le commentaire
        Incrémenter le compteur
    **end for**
    **if** compteur $\geq 10$ **then**
        **break**
    **end if**
**end for**

### Validation qualitative sur des exemples externes

En complément de l'évaluation sur l'ensemble de validation, une analyse qualitative manuelle a été menée à l'aide d'un outil interactif permettant de prédire le niveau de sarcasme pour des phrases externes au dataset.

L'objectif était de tester la robustesse du modèle sur des formulations variées issues du langage courant.

Voici trois exemples testés :

| **Phrase analysée** | **Prédiction** |
|---|---|
| I have a meeting at 10 a.m., so I need to leave early. | Pas sarcastique (niveau 0) |
| Oh great, another Monday. Just what I needed. | Sarcasme modéré (niveau 1) |
| Wow, your brilliant plan to fix the internet by hitting the router was genius — I'm sure NASA will call you. | Très sarcastique (niveau 2) |

TABLE 4.2 – Exemples de prédictions du modèle sur des phrases externes au dataset

Toutes les prédictions ont été pertinentes et cohérentes avec le ton implicite des phrases proposées. Cette étape confirme la bonne capacité du modèle à s'adapter à des contextes nouveaux et à identifier le sarcasme avec un niveau de précision satisfaisant, même sans dépendre du contexte conversationnel initial.

Ci joint l'algorithme de ce test qualitatif :

**Algorithm 5** Prédiction interactive du niveau de sarcasme d'un texte externe

**function** PREDICTSARCASMLEVEL(texte)
    Passer le modèle en mode évaluation
    Tokeniser le texte avec `tokenizer`, en appliquant le padding, le troncage et la limite à 128 tokens
    Transférer `input_ids` et `attention_mask` vers l'appareil utilisé (GPU ou CPU)
    **Sans gradient** :
        Calculer les logits avec le modèle
        Obtenir la classe prédite avec `argmax`
    Associer la classe prédite à un label :
    — 0 → "Pas sarcastique"
    — 1 → "Sarcasme modéré"
    — 2 → "Très sarcastique"
    **return** le niveau de sarcasme et le label associé
**end function**

**while** vrai **do**
    Demander à l'utilisateur une phrase à analyser
    **if** l'utilisateur entre "stop" **then**
        **break**
    **end if**
    Appeler PREDICTSARCASMLEVEL(phrase)
    Afficher le label et le niveau de sarcasme prédits
**end while**

### 4.5.6 Partie 4 : Interprétation du sarcasme : vers une meilleure explicabilité

L'interprétation des décisions d'un modèle d'apprentissage profond est essentielle, notamment lorsqu'il s'agit de traiter des phénomènes subjectifs et subtils comme le sarcasme.

En effet, bien que les performances quantitatives soient importantes, il est tout aussi crucial de comprendre comment et pourquoi le modèle parvient à ses prédictions.

Dans cette optique, nous avons eu recours à trois méthodes complémentaires d'explicabilité :

- **SHAP (SHapley Additive exPlanations)** est une méthode fondée sur la théorie des jeux, qui attribue à chaque mot du texte une valeur d'importance (positive ou négative) selon sa contribution à la prédiction. Cette approche permet de visualiser précisément quels tokens influencent le plus la décision du modèle.

- **LIME (Local Interpretable Model-agnostic Explanations)** approxime le comportement du modèle à proximité d'une prédiction donnée, à l'aide d'un classifieur simple, afin d'en déduire les variables locales les plus explicatives.

- Les **poids d'attention** du modèle **RoBERTa**, permettant d'observer les tokens sur lesquels le modèle « focalise » son attention durant l'inférence.

Chacune de ces techniques offre un angle d'analyse différent, mais leur complémentarité permet de construire une vision plus complète et plus fiable du raisonnement du modèle face à des contenus sarcastiques.

### Interprétation avec SHAP

Pour mieux comprendre les décisions prises par notre modèle , nous avons eu recours à l'outil d'interprétation SHAP (SHapley Additive exPlanations). SHAP permet d'identifier les mots ayant le plus contribué à une prédiction, en attribuant une valeur d'influence (positive ou négative) à chaque token du texte.

Nous avons sélectionné trois exemples de phrases sarcastiques, et avons utilisé une version adaptée de SHAP pour les modèles de type transformers.La méthode s'appuie sur une fonction de prédiction personnalisée retournant les probabilités de chaque classe, et utilise un masque de type Text fourni par la bibliothèque SHAP pour analyser les textes mot à mot.

Ci joint l'algorithme :

**Algorithm 6** Explication du modèle avec SHAP

**Initialiser** SHAP avec `shap.initjs()`
**function** F_SHAP(liste de textes)
    Tokeniser les textes avec le tokenizer RoBERTa (troncage, padding, max 128)
    Transférer les `input_ids` et `attention_mask` vers l'appareil (GPU/CPU)
    **Sans gradient** :
        Obtenir les `logits` à l'aide du modèle
        Calculer les probabilités avec la fonction `softmax`
    **return** les probabilités sous forme de tableau NumPy
**end function**
Sélectionner quelques exemples de texte pertinents pour l'explication
Créer un `text_masker` avec le tokenizer et le token de masquage "`unk`"
Initialiser un `SHAP Explainer` avec :
— la fonction `f_shap`
— le `text_masker`
Calculer les valeurs de SHAP sur les exemples de texte
Visualiser les contributions des tokens avec `shap.plots.text`

Voici un exemple d'interprétation obtenue :

**Phrase :** "Oh great, another Monday morning full of meetings."

**Prédiction du modèle :** Sarcasme modéré (niveau 1)

**Interprétation SHAP :**

- Les mots "Oh great", "Monday", et "full of meetings" ont eu un poids positif important dans la prédiction sarcastique.

- En revanche, des mots comme "morning" ou "another" ont eu une influence plus neutre.

L'illustration générée par SHAP montre les tokens colorés selon leur impact : plus ils sont rouges, plus ils ont favorisé la prédiction sarcastique ; plus ils sont bleus, plus ils y ont contribué négativement.

Cette visualisation permet non seulement de valider les choix du modèle, mais aussi d'identifier des cas où le sarcasme peut être mal interprété à cause de formulations ambiguës.

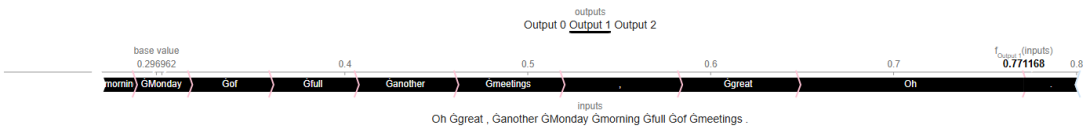


FIGURE 4.1 – Visualisation avec shape

### Analyse via les poids d'attention

Une autre manière d'interpréter le comportement du modèle consiste à examiner les poids d'attention générés par le transformeur RoBERTa.

Dans les modèles à base d'attention, chaque token apprend à « porter attention » à d'autres tokens lors du traitement de la séquence. Ces poids permettent donc d'identifier les relations d'importance entre les mots dans une phrase donnée.Ci joint l'algorithme de cette étape :

**Algorithm 7** Visualisation des poids d'attention de RoBERTa

1: **Entrée** : une phrase d'exemple
2: **Sortie** : une carte thermique représentant les poids d'attention de la dernière couche
3: Charger le modèle `RoBERTa-base` avec `output_attentions=True`
4: Initialiser le tokenizer associé
5: Définir la phrase d'exemple (e.g., "Oh great, another Monday morning full of meetings.")
6: Tokeniser la phrase et générer les tenseurs d'entrée
7: Appliquer le modèle sur les entrées pour récupérer les poids d'attention
8: Extraire les poids d'attention de la **dernière couche**, **tête 0**
9: Convertir les identifiants de tokens en chaînes de caractères
10: Afficher une carte thermique (heatmap) avec Seaborn
11: Ajouter un titre à la figure : "Visualisation des poids d'attention (dernière couche, tête 0)"

Pour illustrer cela, nous avons utilisé la phrase :**Oh great, another Monday morning full of meetings.**

Nous avons extrait les poids d'attention de la dernière couche du modèle RoBERTa, en nous concentrant sur la tête 0. Ces poids ont ensuite été visualisés sous forme de matrice de chaleur (heatmap), où chaque cellule représente l'intensité avec laquelle un mot influence un autre. Ci joint la matrice de l'attention afin de mieux comprendre :

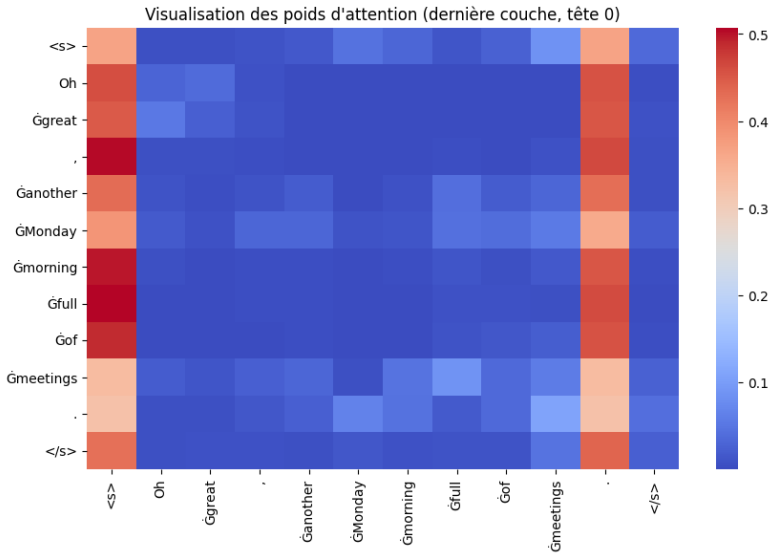


FIGURE 4.2 – Matrice des poids.

Cette visualisation met en évidence les mots-clés que le modèle considère comme importants dans le contexte du sarcasme. Par exemple, dans cette phrase, le mot « *great* » ainsi que « *oh* » reçoivent une attention particulière en lien avec des termes comme « *Monday* » ou « *meetings* », renforçant ainsi l'idée d'un contraste ironique.

Cette méthode permet de mieux comprendre la structure interne du modèle et les liens qu'il établit implicitement entre les mots pour détecter le sarcasme.

**Analyse locale avec LIME**

Pour compléter notre démarche d'interprétation du modèle, nous avons eu recours à LIME (Local Interpretable Model-agnostic Explanations), une méthode d'explicabilité locale.

Contrairement à SHAP, qui s'appuie sur une approche globale inspirée de la théorie des jeux, LIME génère une interprétation spécifique à chaque prédiction en construisant un modèle linéaire localement fidèle autour de l'instance analysée.

Ci joint l'algorithme de cette étape :

**Algorithm 8** Explication locale d'une prédiction avec LIME

1: **Entrée** : une phrase à expliquer
2: **Sortie** : visualisation des poids des mots influents selon LIME
3: Définir le nom des classes : `['Non sarcastique', 'Sarcasme modéré', 'Sarcasme fort']`
4: Initialiser l'explainer LIME avec ces classes
5: Choisir une phrase à analyser (ex. : "I just love when the code crashes after 3 hours of training.")
6: Définir une fonction prédictive `predict_proba` :
   — Tokeniser les phrases avec RoBERTa
   — Appliquer le modèle entraîné
   — Calculer les probabilités via la softmax
   — Retourner les scores sous forme de tableau NumPy
7: Appeler `explain_instance` avec le texte cible et la fonction `predict_proba`
8: Visualiser l'explication dans le notebook (ou exporter pour intégration graphique)

Dans notre cas, nous avons analysé la phrase : **I just love when the code crashes after 3 hours of training.**

Cette phrase a été soumise au modèle pour prédiction, et LIME a généré les probabilités suivantes :

- Sarcasme modéré : 48 %.
- Non sarcastique : 34 %.
- Sarcasme fort : 18 %.

La prédiction finale correspond donc à "Sarcasme modéré", ce qui est cohérent avec l'interprétation humaine.L'illustration suivante certifie les résultats obtenus :

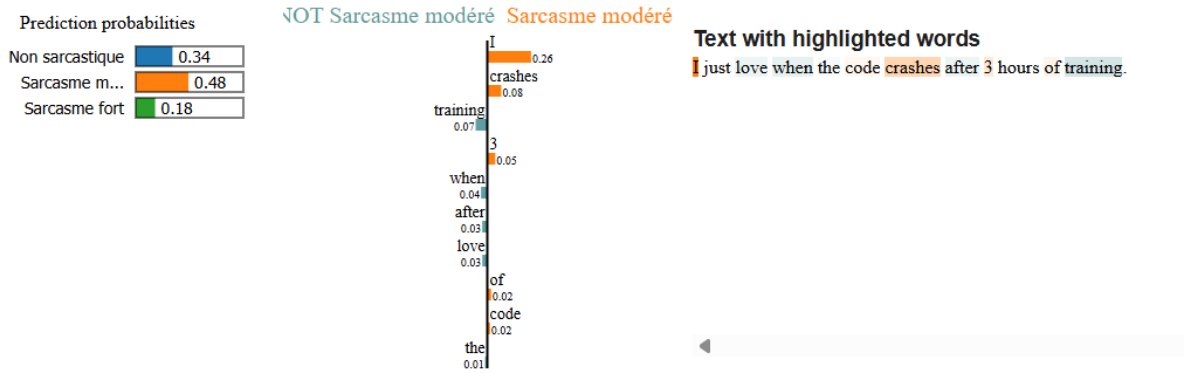


FIGURE 4.3 – Visualisation avec Lime

LIME met en évidence les mots qui influencent le plus la décision du modèle, parmi lesquels :

« **I** », « **crashes** », « **training** », « **when** », « **love** », tous ces termes ont contribué à l'activation de la classe sarcastique, notamment en raison du contraste entre des expressions positives comme love et des contextes négatifs comme code crashes.

L'utilisation de LIME permet de valider localement la prédiction du modèle et d'identifier les indices linguistiques qui influencent la détection du sarcasme, ce qui renforce la transparence et la confiance dans le modèle développé.

### 4.5.7 Partie 5 : Évaluation et analyse des résultats

Afin de mesurer l'efficacité de notre modèle de classification du sarcasme à trois niveaux (pas sarcastique, sarcasme modéré, sarcasme fort), nous avons mené une double évaluation :

- Une évaluation spécifique, mesurant la précision, le rappel et le score F1 pour chaque classe individuellement (Pas sarcastique, Sarcasme modéré, Très sarcastique), accompagnée d'une matrice de confusion.

- Une évaluation globale, offrant une vue d'ensemble à travers des métriques agrégées pondérées selon la répartition du dataset.

Ci joint l'algorithme de l'étape de l'évaluation :

**Algorithm 9** Évaluation du modèle sur l'ensemble de validation

**Entrée** : modèle entraîné, ensemble de validation
**Sortie** : rapport de classification, matrice de confusion, scores globaux
Passer le modèle en mode `eval()`
Initialiser deux listes vides : `predictions`, `true_labels`
**for** chaque batch dans `val_dataloader` **do**
    Transférer les données sur le GPU
    Désactiver le gradient avec `no_grad()`
    Obtenir les prédictions du modèle
    Appliquer `argmax` pour obtenir les classes prédites
    Ajouter les prédictions et labels à leurs listes respectives
**end for**
Générer un rapport de classification avec `classification_report`
Afficher la matrice de confusion avec `confusion_matrix`
**Scores globaux** :
— `Accuracy` : score global
— `Precision` : moyenne pondérée
— `Recall` : moyenne pondérée
— `F1-score` : moyenne pondérée
Afficher les scores avec 4 chiffres après la virgule

### Évaluation spécifique par classe

Le tableau suivant présente les résultats détaillés obtenus pour chaque classe :

| Classe | Précision | Rappel | F1-score | Effectif |
|---|---|---|---|---|
| Pas sarcastique (0) | 94% | 91% | 93% | 817 |
| Sarcasme modéré (1) | 85% | 85% | 85% | 2843 |
| Très sarcastique (2) | 64% | 66% | 65% | 1085 |
| **Moyenne pondérée** | 82% | 82% | 82% | 4745 |
| **Moyenne macro** | 81% | 81% | 81% | – |

TABLE 4.3 – Résultats détaillés par classe du modèle de détection de sarcasme

**Interprétation :**
Le modèle est particulièrement performant pour détecter les textes non sarcastiques (classe 0), avec un F1-score élevé de 93%. Cela indique qu'il fait peu d'erreurs sur cette classe.

Pour la classe sarcasme modéré (1), qui est la plus représentée, les résultats sont également bons (précision et rappel à 85%), montrant une bonne stabilité.

En revanche, la classe très sarcastique (2) est plus difficile à détecter. Le modèle obtient un F1-score de 65%, ce qui reflète une confusion fréquente avec le sarcasme modéré.

### Matrice de confusion

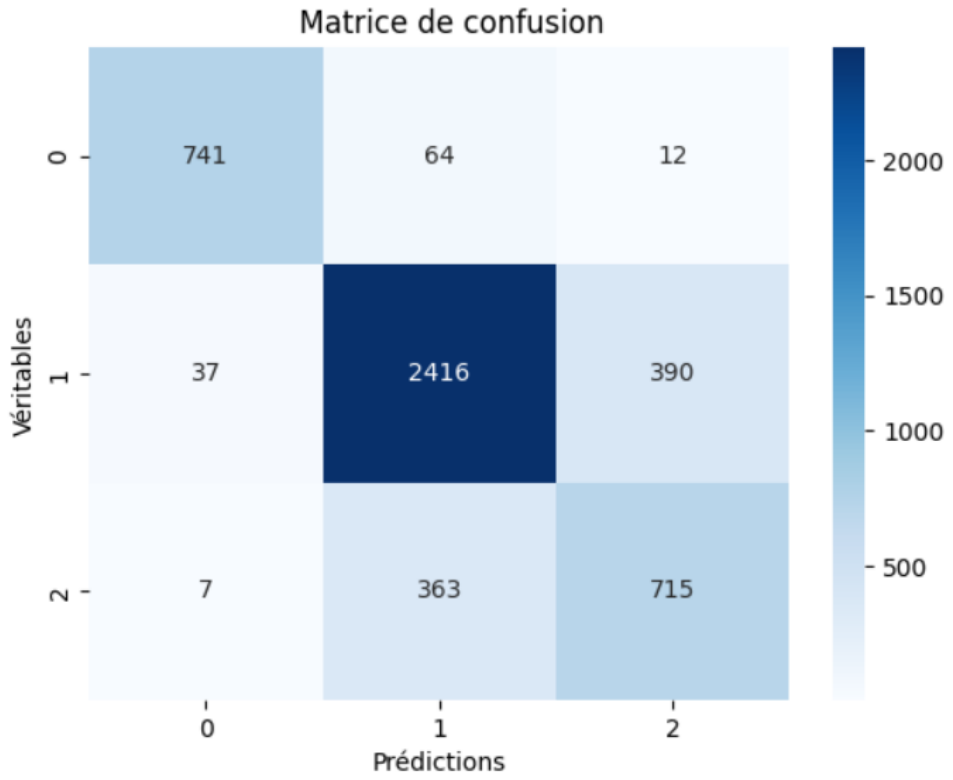


FIGURE 4.4 – Matrice de confusion du modèle de détection du sarcasme

**Analyse :** Sur les 817 vrais exemples "Pas sarcastique", 741 sont bien classés, mais 64 sont confondus avec du sarcasme modéré, montrant une certaine ambiguïté dans les formulations neutres.

Pour les 2843 exemples "Sarcasme modéré", 2416 sont correctement classés, mais 390 sont confondus avec du sarcasme fort, ce qui confirme que le modèle a du mal à distinguer les niveaux de sarcasme élevés.

Enfin, parmi les 1085 phrases "Très sarcastiques", seulement 715 sont bien reconnues. 363 sont mal classées comme sarcasme modéré, ce qui représente un tiers des erreurs sur cette classe.

### Évaluation globale

L'évaluation générale, qui donne une vue d'ensemble des performances, fournit les scores suivants :

| Métrique | Score |
|---|---|
| Exactitude (Accuracy) | 81,60% |
| Précision pondérée | 81,81% |
| Rappel pondéré | 81,60% |
| F1-score pondéré | 81,69% |

TABLE 4.4 – Évaluation globale des performances du modèle

**Interprétation :** Ces résultats indiquent que le modèle est globalement fiable, avec plus de 81% d'exactitude. L'équilibre entre précision et rappel montre qu'il parvient à bien classifier l'ensemble des classes sans déséquilibre majeur, malgré les difficultés liées au sarcasme très fort.

## 4.6 Détection de la double négation

La double négation est une structure linguitique dans laquelle deux expressions négatives se trouvent dans une même phrase. Elle est capable de :

- Renforcer la négation, par exemple : Je n'ai jamais rien dit
- Inverser la polarité de la phrase, exprimant une affirmation implicite, par exemple : Ce n'est pas impossible = C'est possible

Dans le cadre de l'analyse des sentiments, cette structure rend l'interprétation plus compliquée, étant donné que la signification véritable ne correspond pas toujours à la somme des mots utilisés. C'est donc un défi sémantique, souvent mal abordé par les modèles classiques de classification de texte.

### 4.6.1 Objectif de cette section

L'objectif de cette section est de :

- Détecter automatiquement l'existence ou l'absence d'une double négation dans une phrase.
- Améliorer RoBERTa-base en proposant un modèle hybride, capable d'extraire des indices syntaxiques et sémantiques utiles pour ce phénomène.

### 4.6.2 Implémentation

1. **Prétraitement :**

   Dans le but de préparer les données à l'apprentissage, un pipeline de prétraitement rigoureux a été implémenté :

   - **Tokenisation :** nous avons utilisé RobertaTokenizer de la bibliothèque HuggingFace en fixant une longueur maximale à 128 tokens. Chaque phrase est soit coupée, soit complétée par du padding (remplissage) afin de garantir une

cohérence de dimension à l'entrée du modèle.

- **Labellisation :** Chaque exemple de texte est étiqueté à la main avec une étiquette binaire :

  ✓ 0 : absence de négation ou présence d'une négation simple.
  ✓ 1 : existence d'une double négation (par exemple : « Il ne peut pas ne pas y aller »).

- Nettoyage et équilibrage

**Algorithm 10** Encodage d'un texte pour le modèle

1: **Entrée** : un texte brut `texte`, une étiquette `etiquette`, un tokenizer `self.token`, une longueur maximale `self.longmax`
2: **Sortie** : un dictionnaire contenant les entrées encodées pour le modèle
3: Encoder le texte avec `encode_plus`, en activant :
   — la troncature
   — le padding en mode `longmax`
   — la longueur maximale définie par `self.longmax`
   — le retour au format `PyTorch tensor`
4: Extraire les éléments suivants :
   — `entrees` : séquence encodée, compressée avec `squeeze(0)`
   — `attention_mask` : masque d'attention, compressé avec `squeeze(0)`
   — `etiquette` : convertie en tenseur de type `torch.long`
5: **return** un dictionnaire contenant : `entrees`, `attention_mask`, `etiquette`

2. **Dataset PyTorch personnalisé :** une classe Dataset personnalisée a été mise en place en PyTorch, permettant d'intégrer directement le tokenizer RoBERTa dans la phase d'extraction des échantillons grâce à la méthode _getitem_. Cette classe encode des phrases et retourne les tensores nécessaires à l'entraînement : input_ids, attention_mask et labels.

**Algorithm 11** Construction du dataset `DatasetDoubleNegation`

**Entrée** : listes `textes`, `etiquettes`, tokenizer `token`, longueur maximale `longmax`
**Sortie** : objet de type `Dataset` compatible avec PyTorch
**Initialisation** :
- Stocker les `textes` et `etiquettes`
- Enregistrer le `tokenizer`
- Fixer la `longueur maximale`

**Méthode** `__len__` :
Retourner la taille totale du dataset : `len(textes)`
**Méthode** `__getitem__(idx)` :
- Extraire le texte à l'indice `idx`
- Extraire l'étiquette correspondante
- Encoder le texte avec `token.encode_plus` :
  - troncature activée
  - padding = `longmax`
  - retour en tenseur PyTorch
  - max_length = `longmax`
- Retourner un dictionnaire :
  - `entrees` : séquence encodée (`squeeze(0)`)
  - `attention_mask` : masque d'attention (`squeeze(0)`)
  - `etiquette` : étiquette convertie en `torch.tensor(long)`

3. **Modèle hybride :** Le modele hybride proposé a été conçue pour traiter plus efficacement les structures linguistiques complexes, comme la double négation, que le modèle RoBERTa traditionnel rencontre des difficultés à interpréter.

   Voici les différentes composantes du modèle :

   - *RoBERTa-base :* Utilisé en tant que générateur de caractéristiques profondes. Il encode chaque token d'entrée en un vecteur à 768 dimensions, intégrant un contexte bidirectionnel à chaque phase de la phrase.

   - *GRU bidirectionnel (BiGRU) :* La sortie de RoBERTa est envoyée à une couche BiGRU, qui traite la séquence en parallèle dans les deux sens (de l'avant et de l'arrière). Cette phase a pour but de saisir les dépendances syntaxiques et sémantiques longues, indispensables pour l'interprétation de structures telles que « Il ne peut pas ne pas y aller ».

   - *Mécanisme d'attention :* Un module d'attention est appliqué sur la sortie du BiGRU pour attribuer une pondération dynamique aux états cachés les plus utiles pour la prédiction finale. Ce mécanisme permet au modèle d'accorder une attention particulière aux segments qui contiennent des indicateurs de négation (comme « pas », « jamais », « rien », « aucun ») et à leurs relations grammaticales.

   - *Fully Connected Layer (Couche dense) :* La représentation contextuelle finale, influencée par l'attention, est ensuite transmise à un niveau dense (entièrement connecté) qui utilise une fonction d'activation ReLU, jouant le rôle de transfert

vers l'espace de sortie.

- *Prédiction binaire :* Enfin, une couche de sortie composée de deux neurones utilisant l'activation Softmax est utilisée pour générer une distribution de probabilité sur les deux classes possibles :
  ✓ Classe 0 : phrase sans double négation,
  ✓ Classe 1 : phrase contenant une double négation.

**Algorithm 12** Architecture du modèle `DoubleNegation`

**Entrée** : `entrees`, `attention_mask`
**Sortie** : prédiction binaire de la polarité (0 ou 1)
**Étape 1 : Encodage contextuel**
Initialiser le modèle `RoBERTa-base` préentraîné
Geler tous les paramètres de RoBERTa
Débloquer uniquement les paramètres des 4 dernières couches du modèle
**Étape 2 : Extraction des dépendances séquentielles**
Ajouter un GRU bidirectionnel avec :
— `input_size = 768` (taille des sorties de RoBERTa)
— `hidden_size = 256`
— traitement de séquences bidirectionnel
**Étape 3 : Mécanisme d'attention**
Ajouter une couche `Attention(gru_hidden)` pour pondérer les sorties du GRU
**Étape 4 : Classification**
Définir une couche dense composée de :
— `Linear(512, 128)` suivi de `ReLU` et `Dropout`
— `Linear(128, 2)` pour prédire les 2 classes (négation simple ou double)
**function** FORWARD(entrees, attention_mask)
    Passer les données dans RoBERTa (`no_grad`)
    Récupérer les représentations contextuelles (`last_hidden_state`)
    Passer les vecteurs dans le `BiGRU`
    Appliquer le mécanisme d'attention sur les sorties du GRU
    Passer le vecteur de contexte dans la couche de classification
    **return** les logits de prédiction
**end function**

**Algorithm 13** Mécanisme d'attention additive

**Entrée** : `sortiegruputs` $\in R^{(batch, seq\_len, 2\times hidden\_size)}$
**Sortie** : vecteur de contexte pondéré $\in R^{(batch, 2\times hidden\_size)}$
**Initialisation** :
Définir une couche linéaire `couche_score` de dimension $(2 \times hidden\_size) \to 1$
**function** FORWARD(sortiegruputs)
    Calculer les poids d'attention :
    `poids` $\leftarrow$ `softmax(couche_score(sortiegruputs), dim=1)`
    Calculer le vecteur de contexte :
    `contexte` $\leftarrow \sum$ (`poids` $\times$ `sortiegruputs`) sur la dimension temporelle
    **return** `contexte`
**end function**

L'architecture complète offre donc au modèle la capacité de saisir les liaisons complexes entre les mots, tout en lui facilitant d'interpréter correctement le sens fondamentale des doubles négations, en les différenciant des simples négations.

Le schéma ci-dessous illustre le fonctionnement du système :

**Algorithm 14** Entraînement du modèle

**Entrée** : `modele`, `charger_donnees`, `optimiseur`, `planifieur`, `perte_fn`, `appareil`
**Sortie** : perte moyenne sur l'ensemble d'entraînement
Passer le modèle en mode `train()`
Initialiser `perte_totale` $\leftarrow 0$
**for** chaque `lot` dans `charger_donnees` **do**
    Transférer les données sur `appareil` :
    — `entrees` $\leftarrow$ `lot[''entrees'']`
    — `attention_mask` $\leftarrow$ `lot[''attention_mask'']`
    — `etiquettes` $\leftarrow$ `lot[''etiquette'']`
    Obtenir les `sorties` du modèle
    Calculer la perte : `perte` $\leftarrow$ `perte_fn(sorties, etiquettes)`
    Réinitialiser les gradients : `optimiseur.zero_grad()`
    Propager les gradients : `perte.backward()`
    Mise à jour des poids : `optimiseur.step()`
    Mise à jour du taux d'apprentissage : `planifieur.step()`
    Ajouter `perte.item()` à `perte_totale`
**end for**
**return** `perte_totale` divisée par le nombre de lots

### 4.6.3 Entraînement du modèle

L'entraînement du modèle a été effectué sur GPU afin d'accélérer sa convergence. Les paramètres sélectionnés sont les suivants :

- **Optimiseur :** AdamW (adapté aux architectures Transformers).
- **Taux d'apprentissage :** 2e-5 (valeur standard pour fine-tuning).
- **Fonction de perte :** CrossEntropyLoss (classification binaire).
- **Nombre d'époques :** 5.
- **Batch size :** 16.
- **Stratégie d'arrêt anticipé** (early stopping) sur la perte de validation.

**Algorithm 15** Entraînement du modèle

**Entrée** : `modele`, `charger_donnees`, `optimiseur`, `planifieur`, `perte_fn`, `appareil`
**Sortie** : perte moyenne sur l'ensemble d'entraînement
Passer le modèle en mode `train()`
Initialiser `perte_totale` $\leftarrow 0$
**for** chaque `lot` dans `charger_donnees` **do**
    Transférer les données sur `appareil` :
    — `entrees` $\leftarrow$ `lot[''entrees'']`
    — `attention_mask` $\leftarrow$ `lot[''attention_mask'']`
    — `etiquettes` $\leftarrow$ `lot[''etiquette'']`
    Obtenir les `sorties` du modèle
    Calculer la perte : `perte` $\leftarrow$ `perte_fn(sorties, etiquettes)`
    Réinitialiser les gradients : `optimiseur.zero_grad()`
    Propager les gradients : `perte.backward()`
    Mise à jour des poids : `optimiseur.step()`
    Mise à jour du taux d'apprentissage : `planifieur.step()`
    Ajouter `perte.item()` à `perte_totale`
**end for**
**return** `perte_totale` divisée par le nombre de lots

### 4.6.4 Résultats obtenus

L'évaluation du modèle démontre une bonne performance dans la détection de la double négation. Le système est capable de saisir à la fois la structure grammaticale des phrases et les relations de dépendance entre les éléments linguistiques. Cette capacité est essentielle pour traiter les cas complexes où la négation ne se limite pas à des marqueurs explicites, mais s'inscrit dans des formulations syntaxiques plus subtiles ou ambigües.

**Algorithm 16** Évaluation du modèle sur l'ensemble de validation

**Entrée** : `modele`, `charger_donnees`, `appareil`
**Sortie** : affichage du rapport de classification et de la matrice de confusion
Passer le modèle en mode `eval()`
Initialiser deux listes vides : `vraie_etiquette`, `predictions`
Désactiver le calcul des gradients : `with no_grad()`
**for** chaque `lot` dans `charger_donnees` **do**
    Transférer `entrees`, `attention_mask`, `etiquette` sur `appareil`
    Obtenir les `sorties` du modèle
    Extraire les classes prédites avec `argmax` sur les logits
    Ajouter les `etiquettes` vraies et prédites aux listes correspondantes
**end for**
Afficher le rapport de classification avec `classification_report`
Calculer la matrice de confusion avec `confusion_matrix`
Afficher la matrice avec `ConfusionMatrixDisplay.plot()` en bleu

Afin d'illustrer ces performances, le tableau ci-dessous présente les métriques d'évaluation calculées sur le jeu de test pour chaque classe :

| Métrique | Exactitude (%) | Précision (%) | Rappel (%) | F1-score (%) |
|---|---|---|---|---|
| Double Négation | 96.5 | 96.5 | 94.8 | 95.7 |

TABLE 4.5 – Résultats du modèle pour la détection de la double négation

**Matrice de confusion :**

Afin de compléter l'évaluation du modèle, nous avons généré une matrice de confusion qui permet de visualiser la distribution des prédictions correctes et fausses pour chaque catégorie.

Comme illustré dans la figure ci-dessous, le modèle réussit à classer correctement la plupart des exemples, avec juste quelques erreurs :

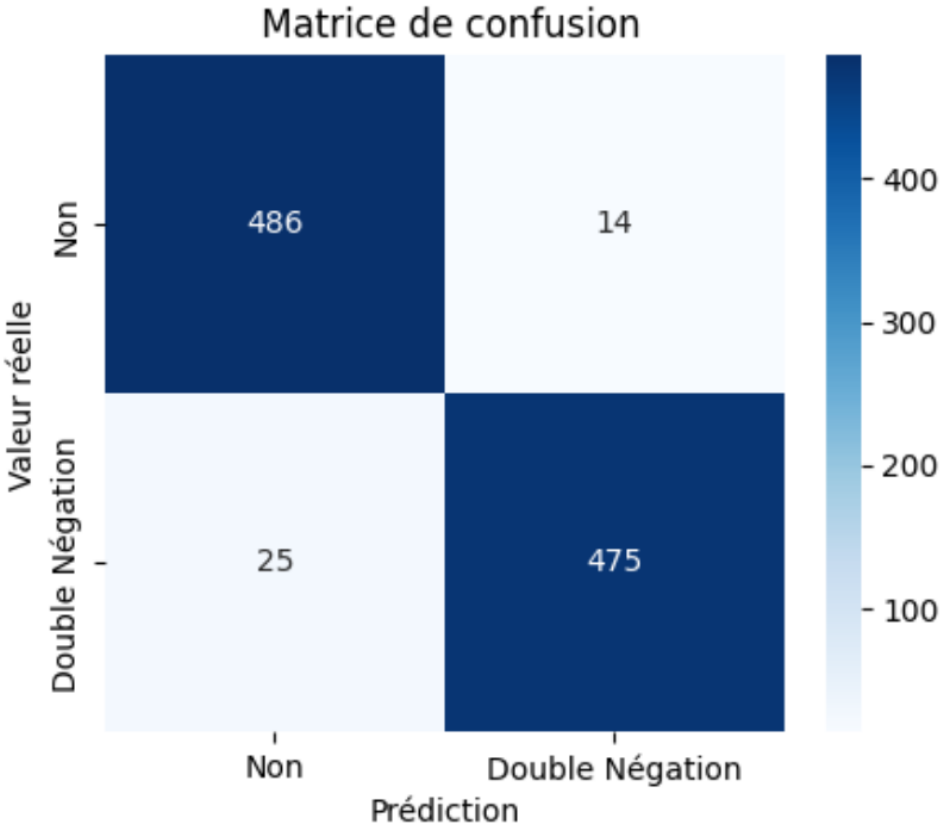


FIGURE 4.5 – Matrice de confusion du modèle de détection de la double négation

Ces résultats confirment les scores élevés observés dans le rapport de classification, avec un F1-score équilibré de 95,7% pour les deux classes. Les erreurs de classification restent limitées et peuvent être dues à des formulations ambiguës ou à la présence d'expressions idiomatiques complexes.

**Test Qualitatif**

Pour analyser plus en détail la fiabilité du modèle proposé, un test qualitatif a été réalisé sur un ensemble de phrases comportant de la double négation implicite, complexe à identifier sans une compréhension contextuelle approfondie.

Voici les résultats issus :

| **Phrase** | **Classe réelle** | **Classe prédite** |
|---|---|---|
| I don't think it's not worth trying again. | Double Négation | Double Négation |
| She likes to read books every evening. | Non | Non |
| It's not like I don't care about this. | Double Négation | Double Négation |
| We followed the steps carefully and succeeded. | Non | Non |

TABLE 4.6 – Résultats du test qualitatif effectué sur le modèle proposé

Ce test qualitatif confirme les résultats quantitatifs obtenus précédemment : l'architecture proposée présente une meilleure compréhension syntaxique et logique, et s'adapte plus efficacement aux expressions ambiguës ou implicites.

Notre modèle est choisi de préférence pour les situations complexes réelles, telles que celles souvent rencontrées sur les réseaux sociaux ou dans les échanges humains naturels, grâce à sa capacité de généralisation.

### 4.6.5 Discussion et analyse comparative

L'analyse qualitative effectuée sur des expressions ambiguës comprenant de la double négation et d'autres non, a mené à une comparaison entre les performances du modèle RoBERTa-base et celles de notre modèle.

Le tableau suivant résume les résultats de cette comparaison :

| **Phrase** | **Classe réelle** | **Prédiction de RoBERTa-Base** | **Prédiction de notre modèle** |
|---|---|---|---|
| Oh great, my favorite thing working late on a Friday ! | Pas de Double Négation | Double Négation | Pas de Double Négation |
| I totally needed my coffee to spill all over my laptop this morning. | Pas de Double Négation | Double Négation | Pas de Double Négation |
| Wow, this app crashing for the tenth time is really amazing. | Pas de Double Négation | Double Négation | Pas de Double Négation |
| Yeah, because staying up all night is so healthy. | Pas de Double Négation | Double Négation | Double Négation |
| He's not completely unaware of the consequences. | Double Négation | Double Négation | Double Négation |
| I don't think it's not working. | Double Négation | Double Négation | Double Négation |
| That's not entirely untrue. | Double Négation | Double Négation | Double Négation |

Table 4.7 – Comparaison entres les tests qualitatifs

Le tableau précédent montre que :

- RoBERTa-base identifie incorrectement la double négation dans des phrases où elle n'est pas présente.
- Notre modèle démontre une meilleure précision générale, en détectant correctement les structures grammaticales à double négation.

Cette différence de performance s'explique principalement par l'architecture spécifique de notre modèle. Le BiGRU, associé au mécanisme d'attention, facilite une meilleure détection des relations séquentielles complexes entre les éléments de la phrase, particulièrement lorsque les indicateurs de négation sont distants l'un de l'autre.

Contrairement à RoBERTa-base, qui analyse le texte de manière plus générale sans se focaliser spécifiquement sur ces structures, notre approche souligne les expressions principales qui altèrent le sens, ce qui améliore considérablement l'identification des doubles négations.

Bien que notre modèle présente encore certaines imperfections, surtout sur des phrases ambiguës, il reste plus fiable pour cette tâche spécifique. Cet avantage est essentiellement significatif dans des situations où il est essentiel de saisir les particularités linguistiques, telles que l'analyse des sentiments ou le traitement automatique du langage naturel, car une mauvaise compréhension pourrait conduire à des résultats erronés.

Par conséquent, l'approche suggérée présente une correspondance supérieure pour la détection et la gestion des doubles négations par rapport aux modèles de langage traditionnels tels que RoBERTa-base.

### 4.6.6 Pistes futures

Une approche ambitieuse pourrait être d'intégrer un élément dédié à la modélisation de la syntaxe profonde des phrases, en particulier via un parseur syntaxique neuronal (par exemple, basé sur les dépendances syntaxiques).

L'idée serait de former un sous-module capable d'identifier de manière spécifique les nœuds de négation dans l'arbre des dépendances d'une phrase, ainsi que les liens qu'ils maintiennent avec les autres termes.

Ce composant pourrait par la suite être intégré (en utilisant un mécanisme d'attention croisée ou d'alignement) à un encodeur de style transformer tel qu'ELECTRA ou DeBERTa, dans le but d'orienter la représentation contextuelle vers une meilleure réactivité face aux constructions de "double négation".

Cette approche permettrait de :

- Détailler les structures complexes comme les triples négations ou les négations interrompues par des adverbes ;
- Détermination des portées de la négation (identification des scopes), un problème fréquemment rencontré dans les textes non formels ;
- Offrir une base solide pour établir une détection causale entre polarité réelle et structure grammaticale, spécifiquement dans les contextes de fausse information ou d'ironie sous-entendue.

L'entraînement pourrait se faire en multi-task learning, en combinant des objectifs syntaxiques (parsing, tagging de négation) et sémantiques (classification sentiment, opinion, etc.), ce qui offrirait une représentation détaillée, compréhensible et adaptable à d'autres tâches linguistiques liées à la négation.

## 4.7 Conclusion

Ce chapitre a présenté deux modules complémentaires dédiés à l'analyse automatique de formulations complexes : l'un centré sur la détection du sarcasme à différents niveaux d'intensité, l'autre sur l'identification précise de la double négation dans des énoncés ambigus. Ces deux phénomènes, bien que différents, partagent une forte dépendance au contexte

discursif et échappent souvent aux approches classiques.

Pour y répondre, nous avons conçu des architectures hybrides combinant la puissance de RoBERTa avec des réseaux séquentiels (BiGRU) et un mécanisme d'attention. Cette combinaison a permis d'améliorer sensiblement la sensibilité aux nuances linguistiques, en particulier dans les cas où le sens réel ne peut être déduit que par une lecture contextuelle approfondie.

Les résultats obtenus sont encourageants : un F1-score de 81,69% pour la classification du sarcasme, et 95,7% pour la détection de la double négation, avec une réduction notable des erreurs d'interprétation. L'apport des techniques d'interprétabilité (SHAP, attention, LIME) a en outre renforcé la transparence et la compréhension fine des décisions prises par les modèles.

Dans l'ensemble, les expérimentations menées confirment l'efficacité des solutions proposées dans des contextes linguistiques exigeants. Elles posent les bases de systèmes plus robustes, capables de mieux capter les subtilités de la communication humaine, avec des applications concrètes en analyse d'opinion, modération intelligente ou agents conversationnels. Des améliorations futures pourraient encore être envisagées à travers l'ajout de composants syntaxiques spécialisés ou de stratégies d'apprentissage multitâche.

# Chapitre 5

# Nouveau modèle pour la détection du sarcasme

## 5.1 Introduction

Ce chapitre vise à concevoir un système de reconnaissance automatique du sarcasme dans des commentaires issus des médias sociaux.

Dans le chapitre précédent, nous avons proposé une architecture hybride (RoBERTa + Bi-GRU + attention) pour détecter la double négation et un système qui détecte le sarcasme via un fine-tuning de RoBERTa-base . Bien que les résultats aient montré un certain niveau de performance (F1-score global de 81,69 % pour le sarcasme), l'analyse qualitative a mis en évidence plusieurs limitations, notamment une confusion entre les niveaux de sarcasme modéré et exagéré, ainsi qu'une sensibilité insuffisante aux constructions linguistiques subtiles typiques des échanges informels.

Ces constats nous ont amenés à concevoir un modèle distinct et plus spécialisé, dédié exclusivement à la détection du sarcasme. Contrairement aux approches classiques qui binaire les classifications (sarcastique / non sarcastique), notre méthode adopte une granularité plus fine, avec trois niveaux :

- 0 : non sarcastique,
- 1 : sarcasme modéré,
- 2 : sarcasme exagéré.

Pour répondre à ces objectifs, nous avons conçu un modèle personnalisé de type Transformer, inspiré de BERT mais entraîné depuis zéro sur un corpus adapté, dans le but d'améliorer la compréhension de constructions linguistiques complexes dans des échanges informels.

## 5.2 Motivation du choix de la construction de ce modèle personnalisé

Le sarcasme constitue un phénomène linguistique particulièrement complexe à modéliser, en raison du décalage implicite entre le sens littéral et l'intention réelle du locuteur. Bien que l'architecture conçue dans le chapitre 4, appliquée à la fois à la détection du sarcasme et à celle de la double négation a permis d'obtenir des résultats encourageants sur les deux tâches, les performances observées pour la détection du sarcasme sont restées limitées, en particulier dans la distinction entre les degrés modéré et exagéré.

Cela indique que la gestion simultanée du sarcasme et de la double négation dans un seul modèle compromet la spécialisation nécessaire pour saisir les nuances sémantiques spécifiques au discours sarcastique. De plus, des modèles préentraînés standard comme BERT ou RoBERTa, qui sont performants pour les tâches générales, ont du mal à repérer les formes implicites et contextuelles du sarcasme sans une adaptation spécifique.

Pour ces raisons, nous avons décidé de construire un modèle entièrement dédié au sarcasme, basé sur une architecture BERT-like personnalisée.

Ce choix nous permet de concentrer l'apprentissage sur des signaux linguistiques plus fins, spécifiques au sarcasme, en nous appuyant sur :

- un corpus annoté manuellement selon trois niveaux de sarcasme (0, 1, 2).
- un tokenizer adapté au langage informel issu des réseaux sociaux.
- un entraînement depuis zéro pour apprendre des représentations mieux contextualisées.

Cette démarche nous a permis de :

- concevoir une architecture spécialisée, libérée des contraintes de la double tâche.
- tirer parti d'un corpus ciblé pour améliorer la sensibilité aux indices sarcastiques.
- optimiser chaque étape du pipeline pour la compréhension d'énoncés ambigus.
- améliorer la capacité du modèle à reconnaître les intentions implicites et les inversions de sens typiques du sarcasme.

## 5.3 Traitement du sarcasme

### 5.3.1 Préparation du corpus pour la détection du sarcasme

Afin de construire un modèle performant et adapté à la détection du sarcasme, nous avons constitué un corpus que nous avons annoté à l'aide d'un *prompt* conçu par nous même, exécuté via le modèle **Flan-T5 Large**. Bien que l'annotation ait été automatisée, elle a été encadrée, supervisée et validée manuellement tout au long du processus. Les données

utilisées proviennent de de deux datasets, connus pour leur richesse en expressions sarcastiques.

Chaque échantillon comprend :

- un commentaire ;
- son contexte conversationnel (le commentaire précédent) ;
- et un niveau de sarcasme, défini selon la classification suivante :
  - 0 : Non sarcastique ;
  - 1 : Sarcasme modéré ;
  - 2 : Sarcasme exagéré.

Cette structuration contextuelle vise à capturer la dimension pragmatique du sarcasme, souvent dépendante de l'interaction avec un énoncé antérieur.

Les données ont été préalablement nettoyées pour supprimer les doublons et les lignes incomplètes, puis structurées dans un fichier `train.txt`, selon le format suivant :

```
[CONTEXT] texte_du_contexte [COMMENT] texte_du_commentaire
```

L'ajout explicite des balises `[CONTEXT]` et `[COMMENT]` permet d'ancrer clairement la séparation entre le contexte et le commentaire, ce qui sera exploité plus tard lors de la tokenisation.

Le corpus final contient plus de 23 000 exemples, et constitue la base d'entraînement pour la phase de préentraînement de notre propre modèle.

Ci-joint l'algorithme correspondant à la première étape de notre pipeline (constitution du corpus d'entraînement) :

**Algorithm 17** Préparation du corpus d'entraînement à partir du fichier annoté

**Entrée** :
- `dataset_nettoye.csv` contenant les colonnes `comment`, `parent_comment`

**Sortie** : fichier texte `train.txt` prêt pour l'entraînement du tokenizer

Charger le fichier CSV avec pandas
Supprimer les lignes contenant des valeurs manquantes dans `comment` ou `parent_comment`
**for all** lignes du dataset **do**
    Créer une nouvelle chaîne de la forme :
    `[CONTEXT] parent_comment [COMMENT] comment`
**end for**
Enregistrer chaque ligne générée dans le fichier `train.txt`, une par ligne

### 5.3.2 Construction du tokenizer personnalisé

Afin de disposer d'un modèle entièrement construit depuis zéro, nous avons également conçu notre propre tokenizer personnalisé en s'appuyant sur l'algorithme **Byte-Pair Encoding (BPE)**, qui permet d'obtenir une représentation optimisée du vocabulaire spécifique à notre corpus.
La création du tokenizer s'est déroulée en plusieurs étapes clés :

**Normalisation :** Tous les textes ont été :

- convertis en minuscules (*lowercase*) ;
- décomposés selon la forme NFD pour gérer les accents ;
- nettoyés de leurs accents (*strip accents*).

**Pré-tokenisation :** Une séparation basique a été appliquée à l'aide du pré-tokeniseur `Whitespace`, afin de découper le texte en unités lexicales en fonction des espaces.

**Définition des tokens spéciaux :** Nous avons ajouté les tokens suivants, essentiels au bon fonctionnement de notre futur modèle `BERT` :

- `[PAD]`, `[UNK]`, `[CLS]`, `[SEP]`, `[MASK]` : tokens standards utilisés dans BERT ;
- `[CONTEXT]`, `[COMMENT]` : tokens personnalisés pour marquer explicitement la structure du dialogue dans notre corpus.

**Entraînement du tokenizer :** Le tokenizer a été entraîné sur le fichier `train.txt` contenant l'ensemble de nos données formatées, avec un vocabulaire limité à **16 000 tokens**. Cette taille permet d'équilibrer couverture lexicale et efficacité mémoire.

**Conversion au format Hugging Face :** Une fois entraîné, le tokenizer a été converti au format `PreTrainedTokenizerFast`, le rendant compatible avec l'architecture `BertModel` de la bibliothèque `Transformers`, tout en conservant les tokens spéciaux et les spécificités définies.
L'ensemble du tokenizer a été sauvegardé dans le dossier `tokenizer_hf`, prêt à être utilisé pour la suite de la création du modèle.
Ci-joint l'algorithme correspondant à la deuxième étape de notre pipeline (construction du tokenizer personnalisé) :

**Algorithm 18** Construction d'un tokenizer personnalisé avec Byte Pair Encoding (BPE)

**Entrée** : fichier `train.txt` contenant les lignes `[CONTEXT] ... [COMMENT] ...`
**Sortie** : tokenizer entraîné et sauvegardé au format `tokenizer_hf/tokenizer.json`

Initialiser un tokenizer avec le modèle BPE
Appliquer les normalisations suivantes :
— Conversion en minuscules
— Suppression des accents (normalisation NFD + StripAccents)
Utiliser une pré-tokenisation par séparation sur les espaces (`Whitespace`)
Définir les tokens spéciaux :
— `[PAD]`, `[UNK]`, `[CLS]`, `[SEP]`, `[MASK]`, `[CONTEXT]`, `[COMMENT]`
Configurer l'entraîneur BPE avec un vocabulaire de taille 16 000
Entraîner le tokenizer sur le fichier `train.txt`
Sauvegarder le tokenizer entraîné dans le dossier `tokenizer_hf`

### 5.3.3 Conception du modèle sur basé BERT personnalisé

Après la construction du tokenizer adapté à notre corpus, nous avons entrepris la conception d'un modèle BERT personnalisé, entraîné entièrement depuis zéro.

L'objectif était de disposer d'une architecture ajustée à notre tâche spécifique de reconnaissance du sarcasme à trois niveaux (non sarcastique, sarcastique modéré, sarcastique exagéré), tout en garantissant une bonne capacité de généralisation.

**Choix des paramètres de l'architecture :** Nous nous sommes appuyés sur la configuration de `BERT-base` comme point de départ, tout en apportant des ajustements ciblés pour adapter le modèle à la nature de notre corpus :

| **Paramètre** | **Valeur choisie** | **Justification** |
|---|---|---|
| Taille du vocabulaire | 16 000 | Correspond à la taille du tokenizer entraîné sur le corpus. |
| Taille des embeddings | 768 | Taille classique de BERT-base, adaptée à la richesse syntaxique attendue. |
| Nombre de couches (layers) | 8 | Un compromis entre profondeur et vitesse d'entraînement. |
| Têtes d'attention | 12 | Reprise de BERT-base pour conserver un bon parallélisme de traitement. |
| Dimension FFN | 3 072 | Taille intermédiaire classique dans les architectures Transformer. |
| Taille max des séquences | 512 | Permet le traitement de paires contextuelles longues. |
| Type de vocabulaire | 2 | Permet la distinction entre les tokens `[CONTEXT]` et `[COMMENT]`. |
| Dropout | 0.1 | Pour régulariser l'apprentissage et limiter l'overfitting. |

TABLE 5.1 – Configuration du modèle BERT personnalisé

Ce modèle a été implémenté à l'aide de la classe `BertConfig` de la bibliothèque `Transformers`, et initialisé avec `BertForMaskedLM` afin d'effectuer un pré-entraînement en *Language Modeling* (MLM) sur notre propre corpus. Voici l'algorithme pour la conception et le pré-entraînement du modèle BERT personnalisé via le masquage de mots (MLM) :

**Algorithm 19** Conception du modèle BERT personnalisé et entraînement par masquage de mots (MLM)

1: **Entrée** : tokenizer personnalisé (`tokenizer_hf`), données tokenisées à partir de `train.txt`
2: **Sortie** : modèle BERT préentraîné sauvegardé dans `bert_siham_pretrained`

3: Définir une configuration BERT (`BertConfig`) :
   — Taille du vocabulaire = taille du tokenizer
   — `hidden_size` = 768, `num_layers` = 8, `num_attention_heads` = 12
   — Taille du feed-forward = 3072
   — Dropout = 0.1
   — `type_vocab_size` = 2 (pour segmenter CONTEXT et COMMENT)
4: Instancier un modèle `BertForMaskedLM` avec cette configuration
5: Créer un dataset de type MLM avec masquage aléatoire (probabilité = 15%)
6: Initialiser un `Trainer` avec :
   — `batch_size` = 16, `epochs` = 5
   — `data_collator` adapté au MLM
   — `tokenizer` et données tokenisées
7: Entraîner le modèle sur les données non étiquetées
8: Sauvegarder le modèle entraîné dans `bert_pretrained`

### 5.3.4 Passage au modèle de classification

Une fois notre modèle `BERT` pré-entraîné sur le corpus via la tâche de *Masked Language Modeling (MLM)*, nous avons adapté cette architecture à une tâche de classification supervisée à trois classes, correspondant aux trois niveaux de sarcasme :

- 0 : Non sarcastique ;
- 1 : Sarcastique modéré ;
- 2 : Sarcastique exagéré.

**Adaptation de l'architecture :** Nous avons utilisé la classe `BertForSequenceClassification` de la bibliothèque `Transformers` pour transformer notre modèle pré-entraîné en classificateur. Cette classe ajoute une couche linéaire de sortie sur le vecteur `[CLS]`, avec un nombre de neurones égal au nombre de classes (`num_labels=3`).

Le modèle a été initialisé à partir des poids du modèle MLM sauvegardé dans le répertoire `./bert_pretrained`. Cela nous a permis de réutiliser les représentations linguistiques apprises lors du pré-entraînement pour améliorer la performance de la classification.

**Préparation des données** Le dataset original a été retravaillé pour inclure uniquement les colonnes nécessaires à la classification :

- `comment` ;
- `parent_comment` ;
- `sarcasm_level` (la vraie étiquette cible, renommée en `label`).

Nous avons ensuite réalisé une division stratifiée du corpus en deux ensembles :

- Entraînement : 80 % ;
- Test : 20 %.

Cette répartition permettait de garantir une représentation équilibrée des trois classes dans chaque sous-ensemble.

**Construction du dataset PyTorch** Pour la phase de *fine-tuning*, nous avons construit une classe personnalisée `SarcasmDataset`, chargée de :

- concaténer les champs `parent_comment` et `comment` selon le format :

  `[CONTEXT] ... [COMMENT] ...`
- appliquer la tokenisation via notre tokenizer personnalisé ;
- retourner des tenseurs `PyTorch` : `input_ids`, `attention_mask`, `labels`.

Voici l'algorithme 20 concernant le passage au modèle de classification à partir du modèle BERT préentraîné :

**Algorithm 20** Passage du modèle BERT préentraîné au modèle de classification

1: **Entrée** : modèle préentraîné `bert_pretrained`, tokenizer `tokenizer_hf`, dataset annoté avec labels {0, 1, 2}
2: **Sortie** : modèle BERT fine-tuné pour la classification à 3 classes

3: Charger le modèle `BertForSequenceClassification` à partir de `bert_pretrained`
4: Spécifier `num_labels = 3` pour la classification en trois niveaux de sarcasme
5: Préparer les données d'entrée en concaténant les paires :
   — Texte = `[CONTEXT] parent [COMMENT] comment`
   — Tokenisation avec padding/troncature à longueur fixe
6: Construire une classe `SarcasmDataset` retournant les `input_ids`, `attention_mask` et le `label`
7: Initialiser un nouvel objet `Trainer` avec :
   — Le modèle de classification
   — Le dataset tokenisé
   — Le tokenizer personnalisé
   — Les paramètres d'apprentissage (`batch_size` = 16, `epochs` = 5)
8: Lancer l'entraînement supervisé avec les étiquettes `sarcasm_level`
9: Sauvegarder le modèle fine-tuné dans `bert_finetuned`

### 5.3.5 Entraînement du modèle de classification

Une fois le modèle `BertForSequenceClassification` construit à partir des poids préentraînés obtenus via la tâche de masquage de mots (*Masked Language Modeling*), nous avons procédé à l'entraînement supervisé sur notre corpus annoté en trois niveaux de sarcasme.

L'entraînement a été réalisé à l'aide du framework `Transformers` de Hugging Face. Les principales caractéristiques de cette phase sont les suivantes :

- **Durée** : 5 époques d'entraînement ;
- **Taille de lot (batch size)** : 16 exemples par lot ;
- **Objectif** : apprentissage de la correspondance entre un couple `(contexte, commentaire)` et l'un des trois niveaux de sarcasme : *non sarcastique*, *modéré* ou *exagéré*.

L'entraînement a permis d'adapter notre modèle préentraîné à la tâche spécifique de classification fine du sarcasme, tout en tenant compte de la structure contextuelle des données, grâce à l'utilisation explicite des balises `[CONTEXT]` et `[COMMENT]` introduites dès la phase de préparation des données.

Voici l'algorithme correspondant à l'entraînement du modèle de classification :

**Algorithm 21** Entraînement du modèle BERT pour la classification du sarcasme

1: **Entrée** :
   — `train_dataset` : données d'entraînement étiquetées
   — `model` : instance de `BertForSequenceClassification`
   — `tokenizer` : tokenizer personnalisé
2: **Sortie** : modèle BERT fine-tuné

3: Définir les paramètres d'entraînement :
   — Nombre d'époques : 5
   — Taille de batch : 16
   — Sauvegarde automatique toutes les 300 étapes
   — Désactiver les rapports externes (ex. : `wandb`)
4: Initialiser l'objet `Trainer` avec :
   — Le modèle
   — Les arguments d'entraînement
   — Le dataset d'entraînement
   — Le tokenizer
5: Lancer la méthode `train()` pour démarrer l'apprentissage supervisé
6: À la fin, sauvegarder le modèle fine-tuné dans `bert_finetuned`

### 5.3.6 Évaluation et analyse des résultats

Afin de mesurer l'efficacité de notre modèle de classification du sarcasme à trois niveaux (non sarcastique, sarcasme modéré, sarcasme exagéré), nous avons mené une double évaluation :

- **Une évaluation spécifique**, mesurant la précision, le rappel et le score F1 pour chaque classe individuellement, accompagnée d'une matrice de confusion ;
- **Une évaluation globale**, offrant une vue d'ensemble à travers des métriques agrégées pondérées selon la répartition du dataset.

### Évaluation spécifique par classe

Le tableau suivant présente les résultats détaillés obtenus pour chaque classe :

| **Classe** | **Précision** | **Rappel** | **F1-score** | **Effectif** |
|---|---|---|---|---|
| Pas sarcastique (0) | 99.73 % | 90.58 % | 94.93 % | 817 |
| Sarcasme modéré (1) | 93.95 % | 98.84 % | 96.33 % | 2843 |
| Sarcasme exagéré (2) | 96.64 % | 90.14 % | 93.28 % | 1085 |
| **Moyenne pondérée** | 95.56 % | 95.43 % | 95.39 % | 4745 |
| **Moyenne macro** | 96.77 % | 93.18 % | 94.85 % | – |

TABLE 5.2 – Résultats détaillés par classe du modèle de détection du sarcasme

### Interprétation

Le modèle est particulièrement performant pour détecter les textes non sarcastiques (classe 0), avec un F1-score élevé de 94.93 %. Cela signifie qu'il fait très peu d'erreurs sur cette classe, même si quelques confusions subsistent avec le sarcasme modéré.

Pour la classe *sarcasme modéré* (1), qui est la plus représentée dans le jeu de données, les performances sont excellentes, avec une précision de 93.95 % et un rappel de 98.84 %. Le modèle réussit donc à reconnaître cette classe avec une grande fiabilité.

En ce qui concerne le *sarcasme exagéré* (2), les résultats restent solides avec un F1-score de 93.28 %, mais une légère baisse du rappel (90.14 %) laisse supposer une confusion avec le sarcasme modéré dans certains cas.

### Matrice de confusion

Voici la matrice de confusion associée au modèle de classification :

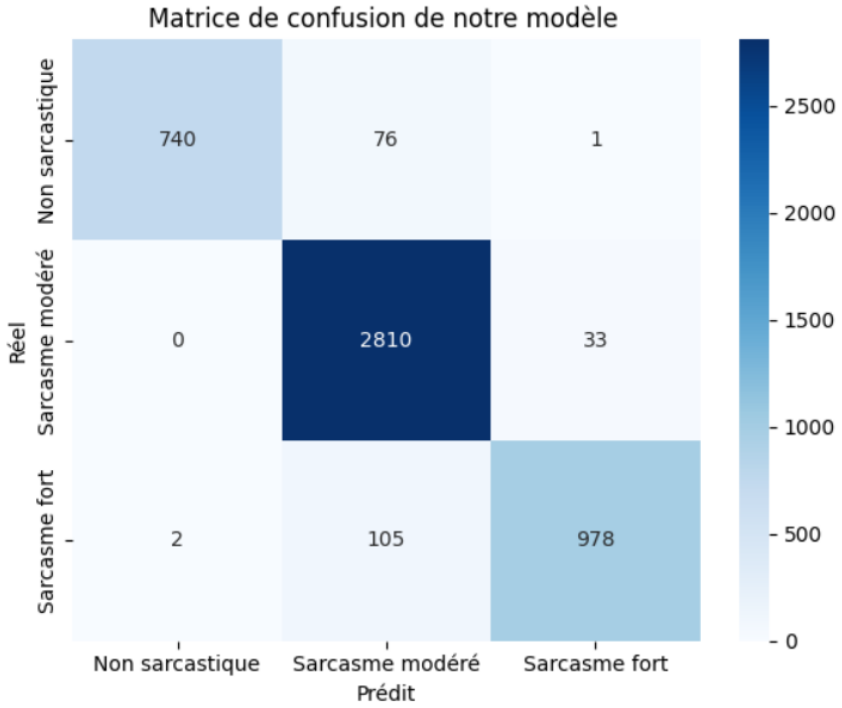


FIGURE 5.1 – Matrice de confusion de notre modèle de détection du sarcasme

**Analyse :**

- Sur les 817 exemples *non sarcastiques*, 740 sont bien classés. Cependant, 76 sont confondus avec le sarcasme modéré, ce qui révèle une certaine ambiguïté dans les formulations neutres.
- Parmi les 2843 commentaires *sarcastiques modérés*, le modèle en identifie correctement 2810, mais en confond 33 avec du sarcasme exagéré.
- Pour les 1085 exemples *sarcasme exagéré*, 978 sont bien reconnus, tandis que 105 sont classés comme sarcasme modéré, confirmant une légère difficulté à distinguer les niveaux de sarcasme les plus élevés.

### Évaluation globale

L'évaluation générale, qui donne une vue d'ensemble des performances, fournit les scores suivants :

| **Métrique** | **Score** |
|---|---|
| Exactitude (Accuracy) | 95.43 % |
| Précision pondérée | 95.56 % |
| Rappel pondéré | 95.43 % |
| F1-score pondéré | 95.39 % |

TABLE 5.3 – Évaluation globale des performances du modèle

**Interprétation :** Ces résultats indiquent que le modèle est globalement très fiable, avec une exactitude de plus de 95 %. L'équilibre entre la précision, le rappel et le F1-score

montre que le modèle parvient à bien classifier l'ensemble des classes sans déséquilibre majeur, même si le sarcasme exagéré reste légèrement plus difficile à détecter que les autres catégories.

Ci-joint l'algorithme de l'évaluation quantitative du modèle de classification :

**Algorithm 22** Évaluation quantitative du modèle BERT sur des données test

**Entrée** :
- `model` : modèle BERT fine-tuné
- `tokenizer` : tokenizer utilisé lors du fine-tuning
- `sarcasm_test.csv` : fichier CSV contenant les exemples de test

**Sortie** : rapport de classification et matrice de confusion

Charger le fichier `sarcasm_test.csv`
Pour chaque ligne :
- Combiner le contexte et le commentaire sous la forme `[CONTEXT] ... [COMMENT] ...`
- Tokeniser le texte avec le `tokenizer`
- Ajouter le label associé (0, 1 ou 2)

Mettre le modèle en mode `eval()`
Pour chaque exemple du test :
- Appliquer le modèle pour obtenir les `logits`
- Prendre la classe prédite via `argmax`
- Sauvegarder la prédiction et le label réel

Calculer et afficher :
- La précision, le rappel, le F1-score par classe
- La matrice de confusion
- Les moyennes pondérées et macro

### Évaluation qualitative

En complément de l'évaluation quantitative, nous avons réalisé une évaluation qualitative afin d'observer le comportement du modèle sur des exemples externes, non présents dans le jeu d'entraînement.

L'objectif était de vérifier sa capacité à détecter le sarcasme de manière contextuelle, dans des situations variées et réalistes.

Nous avons sélectionné un ensemble de 12 paires contextuelles représentant les trois niveaux de sarcasme :

- **Pas sarcastique (classe 0)** : commentaires neutres ou sérieux ;
- **Sarcasme modéré (classe 1)** : ironie subtile, souvent ambigüe ;
- **Sarcasme exagéré (classe 2)** : ironie très évidente, formulation outrancière.

Chaque exemple analysé comprend :

- **Le contexte** : une phrase ou situation antérieure ;
- **Le commentaire à évaluer** ;
- **Le label réel** : 0, 1 ou 2 ;
- **La prédiction du modèle** ;
- **Un indicateur d'exactitude** : ou .

Sur les 12 exemples testés, **9 ont été correctement classés** par le modèle, et **3 ont été mal prédits**.

**Observations**

- Le modèle reconnaît bien les cas évidents de *sarcasme exagéré*, notamment lorsque le commentaire contient des expressions fortes et ironiques telles que *"Best user experience ever"* ou *"Awesome ! I always wanted to walk"*.
- Pour la classe *non sarcastique*, quelques erreurs sont notables. Certaines phrases neutres comme *"That's great news !"* sont interprétées à tort comme sarcastiques, ce qui suggère une tendance du modèle à **surdétecter le sarcasme** dans certaines formulations positives.
- Les confusions entre *sarcasme modéré* et *sarcasme exagéré* sont les plus fréquentes. Cela montre que, même si le modèle distingue bien le sarcasme du non-sarcasme, il peut avoir des difficultés à évaluer **l'intensité** du sarcasme.

Ci-joint l'algorithme de l'évaluation qualitative du modèle de détection du sarcasme :

**Algorithm 23** Évaluation qualitative du modèle sur des exemples externes

**Entrée** :
— `model` : modèle BERT fine-tuné
— `tokenizer` : tokenizer utilisé lors de l'entraînement
— `examples` : liste de paires (`context`, `comment`) avec leur label réel
**Sortie** : comparaison entre prédiction et classe réelle

**for** `(context, comment, true_label)` dans `examples` **do**
    Construire l'entrée sous la forme : `[CONTEXT] context [COMMENT] comment`
    Tokeniser le texte avec le `tokenizer`
    Appliquer le modèle en mode `eval()` pour obtenir les logits
    Récupérer la classe prédite via `argmax(logits)`
    Afficher :
— le contexte
— le commentaire
— le label réel (0, 1 ou 2)
— la prédiction du modèle
**end for**

## 5.4 Comparaison avec les modèles standards

Afin de mieux cerner les limites des modèles standards dans le cadre spécifique de la détection du sarcasme à plusieurs niveaux, nous avons réalisé une comparaison approfondie entre notre modèle basé sur BERT personnalisé et plusieurs modèles préentraînés de référence, notamment RoBERTa, DeBERTa et XLNet.

Cette comparaison, présentée dans le tableau ci-dessous, met en évidence les spécificités de notre approche : entraînement sur un corpus annoté sur mesure, architecture adaptée, vocabulaire personnalisé, et classification en trois niveaux de sarcasme. Contrairement aux modèles existants souvent conçus pour des tâches générales de classification binaire, notre modèle est entièrement orienté vers la reconnaissance fine du sarcasme, ce qui se traduit par une meilleure sensibilité aux nuances et une performance globale plus élevée.

| **Caractéristique** | **Modèle personnalisé (BERT)** | **RoBERTa-base** | **DeBERTa-base** | **XLNet-base** |
|---|---|---|---|---|
| Type d'architecture | BERT from scratch | BERT optimisé | Variante BERT avec disentanglement | Transformer auto-régressif permutationnel |
| Objectif préentraînement | MLM (Masked LM) | MLM dynamique | MLM avec disentangled attention | PLM (Permutation LM) |
| Taille du vocabulaire | 16 000 (BPE personnalisé) | 50 265 (BPE) | 128 000 (WordPiece) | 32 000 (SentencePiece) |
| Tokenisation | BPE entraîné sur le dataset sarcasme | BPE sur large corpus | WordPiece multi-lingue | SentencePiece |
| Nombre de couches (transformers) | 8 | 12 | 12 | 12 |
| Dimension des couches cachées | 768 | 768 | 768 | 768 |
| Têtes d'attention | 12 | 12 | 12 | 12 |
| Taille du feedforward (intermédiaire) | 3072 | 3072 | 3072 | 3072 |
| Dropout | 0.1 | 0.1 | 0.1 | 0.1 |
| Position embeddings | Absolus | Absolus + sinus | Disentangled (séparés) | Relatifs |
| Entraîné avec contexte `[CONTEXT]-[COMMENT]` | Oui (via tokens spéciaux) | Non | Non | Non |
| Tâche cible fine-tuning | 3 classes sarcasme (0/1/2) | Classif. binaire (souvent) | Classif. binaire (souvent) | Classif. binaire (souvent) |

Table 5.4 – Comparaison architecturale entre notre modèle BERT personnalisé et les modèles préentraînés RoBERTa-base, DeBERTa-base et XLNet-base.

## 5.5 Conclusion

Dans ce projet, nous avons conçu de bout en bout un modèle personnalisé de détection du sarcasme, structuré autour d'une classification à trois niveaux : non sarcastique, sarcasme modéré et sarcasme exagéré. Contrairement à une simple adaptation de modèles préexistants, nous avons fait le choix de construire notre propre pipeline, depuis la création du tokenizer jusqu'à l'entraînement d'un modèle BERT sur mesure, afin d'exploiter au mieux les spécificités du langage sarcastique.

Notre démarche s'est appuyée sur une série d'étapes rigoureuses :

- constitution et annotation d'un corpus spécialisé via un prompt appliqué avec le modèle `FLAN-T5-Large` ;
- construction d'un tokenizer BPE personnalisé, intégrant des balises contextuelles comme `[CONTEXT]` et `[COMMENT]` ;
- préentraînement d'un modèle BERT via une tâche de masquage de mots (*Masked Language Modeling*) ;
- spécialisation du modèle par *fine-tuning* pour la classification, sur la base du champ `sarcasm_level` ;
- évaluation quantitative (F1-score global de 95.39 %) et qualitative (tests sur données externes), révélant à la fois la robustesse du modèle et ses limites.‘

Les résultats obtenus sont très encourageants, notamment sur les sarcasmes modérés et exagérés. Le modèle a montré une bonne capacité à capter les nuances ironiques dans les textes, même si certaines confusions subsistent entre les différentes intensités.

Notre approche démontre qu'il est possible, à partir d'un dataset annoté manuellement et d'un entraînement ciblé, de bâtir un modèle efficace capable d'aller au-delà du simple repérage binaire du sarcasme, en intégrant une dimension d'intensité.

Enfin, ce travail ouvre de nombreuses perspectives : l'intégration de signaux multimodaux (émoticônes, ponctuation, tonalité vocale), l'amélioration de la robustesse en contexte multiculturel, ou encore l'exploration de modèles plus légers pour une utilisation en production.

# Chapitre 6

# Analyse des erreurs des systèmes

## 6.1 Introduction

Dans toute démarche d'apprentissage automatique, l'analyse des erreurs constitue une étape clé permettant de dépasser les simples performances chiffrées pour mieux comprendre les limites réelles d'un système. Même lorsque les résultats globaux affichent des scores élevés en précision ou en F1-score, il est crucial d'identifier et de caractériser les cas spécifiques où les modèles échouent. Cette compréhension fine permet non seulement d'améliorer les performances futures, mais aussi de renforcer l'interprétabilité et la robustesse des systèmes, deux aspects essentiels dans des tâches sensibles comme l'analyse de sentiments.

Dans le contexte de ce mémoire, l'analyse porte principalement sur deux phénomènes linguistiques particulièrement complexes : le sarcasme et la double négation. Ces formes de langage figuré ou inversé posent des défis uniques aux modèles, même les plus avancés, car elles impliquent des degrés d'implicite, de contexte culturel ou encore de subjectivité qui échappent souvent aux représentations classiques basées uniquement sur les séquences de mots.

Ainsi, ce chapitre vise à décortiquer les erreurs produites par les trois principaux systèmes que nous avons développés. Au-delà de la simple classification correcte ou incorrecte, nous cherchons à établir une typologie des erreurs, à en analyser les causes possibles (limites du corpus, biais du modèle, ambiguïtés syntaxiques ou pragmatiques) et à proposer des pistes concrètes d'amélioration. Cette analyse constituera une base de réflexion pour les futurs travaux cherchant à modéliser des dimensions subtiles du langage humain.

Les trois modèles analysés dans ce chapitre sont :

- **RoBERTa + BiGRU (sarcasme)** : appliqué à un dataset standard pour évaluer la détection du sarcasme.
- **RoBERTa + BiGRU (négation)** : entraîné sur un corpus annoté autour des structures de négation.
- **Modèle personnalisé BERT (sarcasme)** : conçu spécifiquement pour capter des formes complexes de sarcasme, notamment implicite.

## 6.2 Méthodologie de l'analyse des erreurs

L'analyse des erreurs a reposé sur une approche méthodologique combinant deux volets complémentaires :

- une évaluation quantitative fondée sur des métriques classiques de classification ;
- une évaluation qualitative reposant sur une inspection manuelle des prédictions erronées.

Du point de vue quantitatif, nous avons exploité les indicateurs standards tels que la précision, le rappel et le score F1, ainsi que les matrices de confusion générées pour chaque modèle. Ces outils nous ont permis d'identifier les classes les plus sujettes à confusion, en particulier dans les cas de sarcasme modéré ou implicite.

Les résultats présentés dans les chapitres précédents mettent notamment en évidence un F1-score élevé de 95,7 % pour la détection de la double négation, contrastant fortement avec les 65 % obtenus pour le sarcasme fort. Cette disparité témoigne d'une performance hétérogène selon les phénomènes linguistiques abordés.

Du point de vue qualitatif, une analyse manuelle a été menée sur un sous-ensemble d'exemples mal classés. L'objectif était d'identifier des schémas d'erreurs récurrents, liés à des facteurs tels que la structure syntaxique des phrases, le lexique employé, l'implicite contextuel ou encore la présence d'ambiguïtés pragmatiques et culturelles. Cette démarche a permis de mieux cerner les situations dans lesquelles les modèles échouent, malgré des performances globales satisfaisantes.

Cette double approche avait pour finalité de dégager une typologie structurée des erreurs critiques, d'en analyser les causes profondes, et de proposer des pistes d'amélioration concrètes, fondées à la fois sur les résultats statistiques et sur une lecture fine des données.

## 6.3 Typologie des erreurs identifiées

Malgré les performances globalement satisfaisantes de nos modèles spécialisés avec une précision de 96,5 % pour la détection de la double négation, et un F1-score pondéré de 81,69 % pour la classification du sarcasme ainsi qu'une précision pondérée de 95,56 % pour notre modèle basé sur BERT, plusieurs catégories d'erreurs récurrentes ont été observées.

Ces erreurs se concentrent principalement dans des cas où l'interprétation linguistique nécessite une compréhension implicite, une inférence contextuelle ou une prise en compte de la pragmatique discursive.

La typologie suivante propose une classification qualitative des difficultés les plus représentatives, chacune illustrée par un exemple d'erreur commise par les modèles.

### 6.3.1 Erreurs liées au sarcasme implicite

Le sarcasme implicite constitue un véritable défi pour les modèles de classification. Il repose sur un décalage subtil entre le contenu littéral de l'énoncé et l'intention réelle du locuteur.

Contrairement au sarcasme explicite, ce type de formulation n'utilise ni ponctuation ironique (points d'exclamation répétés, majuscules), ni emojis, ni hyperboles manifestes. L'ironie est ici suggérée par la situation ou par un contraste implicite avec les attentes du lecteur.

En l'absence d'éléments saillants, le modèle adopte une lecture sémantique superficielle, basée sur la polarité lexicale apparente, ce qui conduit à une sous-détection du sarcasme.

**Exemple :** *So happy I got to spend three hours in traffic today.*

- Prédiction (Modèle sarcasme) : Non sarcastique
- Réel : Sarcastique

**Remarque :** Ce type d'erreur pourrait être atténué par l'intégration de mécanismes de détection de l'incongruité ou par l'analyse des corpus adjacents (contextualisation conversationnelle).

### 6.3.2 Erreurs dans l'interprétation de la double négation

Les énoncés contenant deux (ou plus) négations nécessitent un traitement syntaxique et logique rigoureux. La double négation ne correspond pas toujours à une simple affirmation déguisée ; elle peut être utilisée pour atténuer un propos, exprimer une réserve, ou au contraire renforcer une position.

Le modèle, en l'absence d'un module de raisonnement logique ou de parsers syntaxiques avancés, applique une lecture linéaire des unités négatives. Cela engendre souvent des erreurs dans l'interprétation globale.

**Exemple :** *It's not that I don't like it, but...*

- Prédiction : 0 (absence de double négation)
- Réel : 1

**Autre exemple :** *I'm not saying it's not bad.*

→ Une mauvaise segmentation logique conduit souvent à une inversion du sens réel.

### 6.3.3 Erreurs sur les structures contrastives intra-phrastiques

Certains énoncés présentent une évolution interne du ton ou de la position argumentative, passant d'une appréciation positive à une nuance ou une critique. Ces constructions

demandent une lecture dynamique, capable de détecter le retournement discursif.

Or, les modèles actuels tendent à surévaluer le début de phrase, souvent plus positif lexicalement, au détriment du contraste final.

**Exemple :** *It was great... until the last part.*

- Erreur : L'évaluation positive est retenue, la nuance finale est ignorée.

### 6.3.4 Biais de prédiction sur les formulations modalisées ou neutres

Les expressions atténuées ou à polarité floue posent également problème. Formulations comme « not too bad », « kinda nice », « not great » expriment une opinion nuancée, parfois ambiguë ou ironique selon le ton ou le contexte.

Le modèle interprète souvent ces formulations comme neutres, en ignorant leur portée implicite.

**Exemple :** *Not too bad...*

- Prédiction : Non sarcastique
- Réel : Sarcastique implicite

### 6.3.5 Erreurs dues aux expressions idiomatiques et régionales

Les expressions idiomatiques, familières ou régionales échappent souvent à l'analyse car leur signification ne peut être déduite de manière compositionnelle. Leur interprétation exige une connaissance socio-linguistique ou culturelle, rarement représentée dans les corpus d'entraînement.

**Exemple :** *Not bad at all !*

- Prédiction : Non sarcastique
- Réel : Sarcastique (dans l'usage courant)

**Autre exemple :** *That was a piece of cake !*

→ Selon le contexte, cette expression peut être interprétée littéralement ou de manière sarcastique.

### 6.3.6 Limites du modèle personnalisé dans la détection fine du sarcasme

Notre modèle personnalisé BERT classifie le sarcasme en trois niveaux : 0 (non sarcastique), 1 (modéré), et 2 (fort). Il a obtenu de très bonnes performances (F1-score pondéré

de 95,39 %) et surpasse RoBERTa+BiGRU.

Cependant, deux types d'erreurs subsistent :

- Confusion entre sarcasme modéré et fort, notamment quand le ton est ambigu.
- Sur-interprétation : des critiques franches sont à tort perçues comme sarcastiques.

**Illustration :** *You really nailed it this time. Bravo.*

→ Le contexte ou l'intonation peut en faire une critique ou un sarcasme appuyé.

**Conclusion partielle :** Ces erreurs illustrent les limites actuelles du modèle face aux subtilités du langage humain. Elles appellent à intégrer davantage de signaux contextuels, pragmatiques ou même multimodaux pour affiner la détection.

## 6.4 Synthèse visuelle des erreurs observées

Pour compléter l'analyse qualitative et quantitative précédemment menée, nous proposons une représentation graphique synthétique des types d'erreurs identifiés, en fonction de leur fréquence d'apparition.

Cette visualisation permet non seulement d'apprécier la répartition des difficultés rencontrées par les modèles, mais aussi d'identifier les sources d'erreurs les plus critiques à prioriser dans les ajustements futurs.

Elle sert ainsi de support complémentaire aux recommandations formulées dans la section suivante.

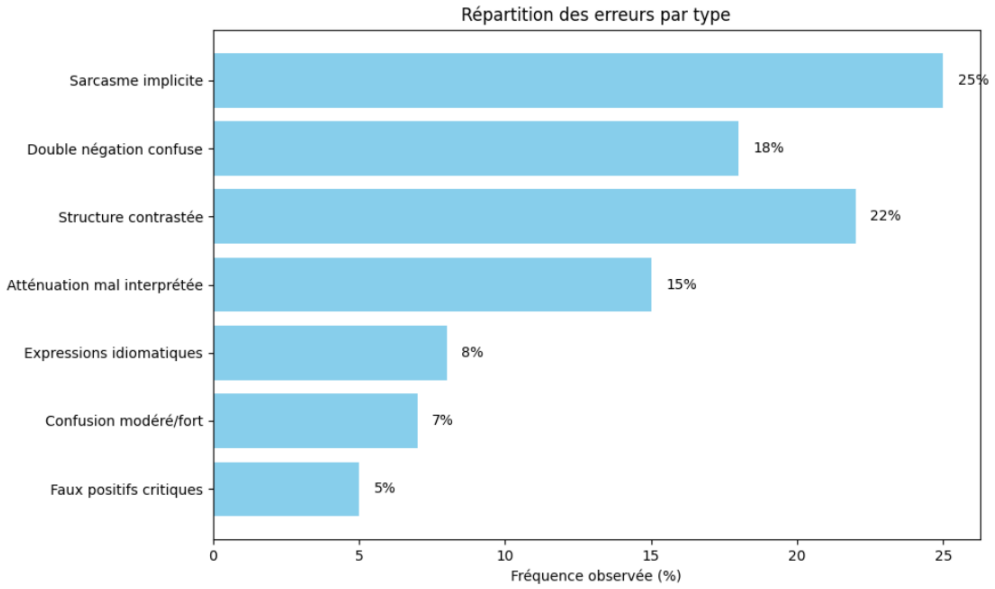


FIGURE 6.1 – Distribution des erreurs selon leur nature linguistique

## 6.5 Analyse quantitative

En complément de l'analyse qualitative, cette section propose une synthèse quantitative des principales erreurs observées. Le tableau ci-dessous dresse un panorama des types d'erreurs identifiés, en les associant à un exemple représentatif, au(x) modèle(s) concerné(s), ainsi qu'à leur fréquence ou impact observé.

Cette représentation tabulaire permet non seulement de mettre en évidence les faiblesses spécifiques de chaque architecture, mais aussi de repérer les motifs récurrents d'erreurs, facilitant ainsi la formulation de pistes d'amélioration futures.

| **Type d'erreur** | **Exemple typique (EN)** | **Modèle concerné** | **Fréquence / Gravité** |
|---|---|---|---|
| Sarcasme implicite | *So thrilled to have missed my job interview.* | RoBERTa + BiGRU (sarcasm) | High |
| Double négation confuse | *It's not like it doesn't matter...* | RoBERTa + BiGRU (negation) | Medium |
| Structure contrastée | *I used to enjoy it... until the last episode.* | All models | High |
| Atténuation mal interprétée | *Not bad at all.* | RoBERTa + BiGRU (sarcasm) | Medium |
| Expression idiomatique / régionale | *Could've been worse.* | RoBERTa + BiGRU / Custom BERT | Low |
| Confusion sarcasme modéré/fort | *Thanks for absolutely nothing.* | Custom BERT (sarcasm levels) | Low–Medium |
| Faux positifs sur critique directe | *He would've been more useful staying home.* | Custom BERT | Rare |

TABLE 6.1 – Typologie quantitative des erreurs par modèle et par situation linguistique

L'analyse du tableau 6.1 révèle que :

- Le modèle **RoBERTa + BiGRU** est particulièrement sensible aux formulations implicites ou atténuées, ce qui se traduit par une forte proportion d'erreurs sur les sarcasmes non explicites, les double négations subtiles ou les constructions atténuées.

- Le **modèle personnalisé BERT**, bien que plus performant, montre des limites dans la détection de la *graduation* du sarcasme. Il a notamment tendance à surévaluer certaines critiques directes en les classant comme sarcastiques modérés.

- Les erreurs liées aux **structures contrastées** (phrases à retournement) affectent tous les modèles de manière marquée, suggérant une faiblesse commune dans la

modélisation du raisonnement discursif.

- Enfin, les erreurs dues aux **expressions idiomatiques ou régionales**, bien que moins fréquentes, révèlent un manque de robustesse face à des variations linguistiques non standardisées.

Cette synthèse met en lumière la nécessité d'intégrer davantage d'informations contextuelles, logiques et culturelles pour affiner la robustesse des modèles face à la complexité du langage naturel.

## 6.6 Analyse des matrices de confusion

Pour approfondir l'analyse des performances, nous avons examiné les matrices de confusion des modèles évalués. Ces matrices révèlent les classes les plus sujettes à confusion, en particulier entre sarcasme modéré (classe 1) et sarcasme fort (classe 2), ou encore entre absence et présence de double négation.

- Pour le modèle RoBERTa + BiGRU (sarcasme), les erreurs sont principalement des confusions entre classes 0 et 1, traduisant une difficulté à détecter les signaux faibles du sarcasme modéré.

- Le modèle BERT personnalisé, quant à lui, tend à confondre les classes 1 et 2, ce qui suggère un besoin d'encodage plus fin de l'intensité sarcastique.

Ces observations confirment la nécessité de modéliser non seulement la présence du sarcasme, mais aussi ses degrés d'intensité.

## 6.7 Recommandations pour corriger les biais

À la lumière des erreurs identifiées lors de l'analyse qualitative et quantitative, plusieurs pistes d'amélioration peuvent être envisagées afin de renforcer la robustesse des modèles face aux structures linguistiques complexes ou implicites.

Les recommandations suivantes visent à atténuer les biais observés et à améliorer la capacité de généralisation des systèmes :

- **Diversification ciblée du corpus d'entraînement :** Il est essentiel d'enrichir le dataset initial avec des exemples représentatifs de cas-limites ou rarement couverts par les données standards. Cela inclut notamment des phrases contenant du sarcasme implicite, des doubles négations non triviales, ou des expressions idiomatiques. Cette augmentation peut être réalisée soit par collecte manuelle, soit par génération automatique via des techniques de paraphrasage contrôlé ou de data augmentation guidée.

- **Prétraitement linguistique enrichi :** L'ajout d'une étape d'analyse syntaxique approfondie (par exemple via des parseurs de dépendances) permettrait de mieux modéliser les relations grammaticales complexes, notamment celles liées aux structures négatives imbriquées ou aux revirements de ton. Cela faciliterait la désambiguïsa-

tion de certaines constructions qui échappent à une lecture purement linéaire.

- **Intégration de lexiques et ressources spécialisées :** L'utilisation de lexiques d'expressions idiomatiques, de formulations ironiques ou de tournures régionales peut améliorer la compréhension de segments à forte connotation contextuelle ou culturelle. Ces ressources peuvent être injectées sous forme de features linguistiques supplémentaires ou intégrées à un module d'attention spécialisé.

- **Atténuation des biais de classification :** Pour réduire les déséquilibres dans la distribution des prédictions (ex. tendance à la sous-détection du sarcasme implicite), il est recommandé d'ajuster dynamiquement les poids de classes durant l'entraînement ou d'utiliser des techniques de rééchantillonnage adaptatif. Des stratégies comme le *focal loss* ou l'entraînement multi-objectifs peuvent également aider à mieux différencier les cas subtils.

- **Ajout de modules pragmatiques complémentaires :** L'inclusion d'un composant spécialisé dans l'analyse de la subjectivité, de l'ironie ou du ton discursif (ex. analyse de sentiment fine-grainée, détection d'intention) peut venir compenser les limites des encodeurs actuels sur les aspects pragmatiques du langage. Ce module peut être conçu sous la forme d'une couche attentionnelle auxiliaire ou d'un classifieur multi-niveaux.

Ces recommandations visent à accroître la sensibilité des modèles aux signaux faibles du langage naturel, en particulier ceux qui ne sont ni explicites ni systématiquement marqués dans le texte. Leur mise en œuvre ouvre des perspectives prometteuses pour la détection plus fiable du sarcasme et des structures négatives complexes dans des contextes variés.

## 6.8 Limites de l'analyse

Bien que cette étude propose une analyse approfondie des erreurs de classification, certaines limites doivent être soulignées :

- L'analyse qualitative repose sur un sous-ensemble limité d'exemples, sélectionnés manuellement, ce qui peut introduire un biais d'interprétation.

- Le contexte conversationnel ou multimodal n'a pas été pris en compte, alors qu'il influence fortement la perception du sarcasme.

- Certaines erreurs peuvent provenir d'ambiguïtés réelles du langage, y compris pour des annotateurs humains, ce qui relativise la notion d'« erreur ».

Ces limites suggèrent des pistes futures d'amélioration, notamment en intégrant des contextes étendus ou des évaluations humaines comparatives.

## 6.9 Conclusion

Cette étude a mis en lumière les limites persistantes des trois modèles spécialisés face à certains phénomènes linguistiques complexes, tels que le sarcasme implicite, les doubles

négations imbriquées et les expressions idiomatiques ou contextuelles. Ces difficultés révèlent les insuffisances des représentations contextuelles actuelles, même lorsqu'elles reposent sur des architectures avancées de type transformeur.

L'analyse fine des erreurs rencontrées ne constitue pas seulement un diagnostic technique, mais représente un levier stratégique pour guider l'amélioration des systèmes de traitement du langage naturel. Elle a permis d'identifier des vulnérabilités spécifiques en particulier dans la compréhension pragmatique et discursive que les modèles peinent encore à modéliser de manière satisfaisante.

Les recommandations formulées ouvrent des perspectives concrètes pour accroître la robustesse et la finesse d'analyse de ces modèles. Qu'il s'agisse d'enrichir les données d'entraînement, d'intégrer des connaissances linguistiques externes ou de coupler l'apprentissage profond avec des modules d'analyse pragmatique, ces pistes dessinent une approche plus hybride et plus sensible aux subtilités du langage.

En définitive, cette étude souligne l'importance de combiner rigueur algorithmique et compréhension linguistique fine pour traiter efficacement les ambiguïtés du langage naturel dans des contextes réels et variés.

# Conclusion Générale et perspectives

Ce travail de fin d'études nous a permis d'explorer en profondeur les défis liés à l'analyse automatique des sentiments exprimés sur les réseaux sociaux, en ciblant deux constructions linguistiques particulièrement délicates : la double négation et le sarcasme implicite. Au fil des expérimentations, nous avons mesuré la complexité réelle que ces phénomènes posent, même aux modèles les plus avancés du traitement automatique du langage naturel.

Loin de nous limiter à une simple application de modèles existants, nous avons choisi d'adopter une approche progressive et critique. Après avoir testé des architectures préentraînées sur un corpus standard, nous avons construit notre propre dataset, réfléchi à la nature des annotations, amélioré les performances via des adaptations structurelles (GRU, attention), et conçu un modèle entièrement personnalisé pour la détection du sarcasme. Cette démarche nous a permis de passer d'une phase de reproduction à une réelle capacité de conception de solutions adaptées.

Les résultats obtenus, mais surtout les difficultés rencontrées, nous ont appris que la qualité d'un système d'analyse de sentiments ne repose pas uniquement sur les performances numériques. La finesse d'interprétation, la robustesse face à des cas ambigus, et l'aptitude à généraliser à des données non vues constituent des enjeux tout aussi essentiels.

D'un point de vue personnel et académique, ce projet nous a offert une expérience complète : construction de corpus, annotation semi-manuelle, entraînement de modèles, analyse d'erreurs, évaluation qualitative... Il nous a permis de mieux comprendre les subtilités du langage humain, tout en consolidant nos compétences en intelligence artificielle, en modélisation de données textuelles, et en esprit critique face aux résultats produits par les algorithmes.

Enfin, plusieurs perspectives se dégagent naturellement de ce travail :

- ✓ Approfondir la qualité des corpus par un enrichissement en données issues de contextes variés (avis clients, forums, dialogues, SMS) ;
- ✓ Coupler les modèles de classification avec des analyses syntaxiques et pragmatiques pour renforcer la compréhension du langage implicite ;
- ✓ Développer des systèmes hybrides, capables de combiner plusieurs modules (sarcasme, double négation, émotion) selon les besoins de l'analyse ;
- ✓ Mettre en œuvre une application web permettant à des utilisateurs non spécialistes de tester et explorer ces modèles en temps réel.

Ce mémoire ne clôt pas un sujet : il ouvre la voie à des recherches futures, où la sensibilité linguistique rencontrera l'intelligence computationnelle pour offrir des outils toujours plus pertinents et humains.

# Bibliographie


1 Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies.
Consulté le 05 février 2025.

2 Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval.
Consulté le 07 février 2025.

3 Plutchik, R. (2001). *The Nature of Emotions*. American Scientist.
Consulté le 10 février 2025.

4 Cambria, E., & Hussain, A. (2012). *Sentic Computing : Techniques, Tools, and Applications*. Springer.
Consulté le 11 février 2025.

5 Qualtrics — https ://www.qualtrics.com/fr/gestion-de-l-experience/etude-marche/analyse-sentiment/
Consulté le 20 février 2025.

6 Elastic — https ://www.elastic.co/fr/what-is/sentiment-analysis
Consulté le 27 février 2025.

7 Innovatiana — https ://www.innovatiana.com/post/sentiment-analysis-powered-by-ai
Consulté le 5 mars 2025.

8 Voxco — https ://www.voxco.com/fr/blog/analyse-des-sentiments-definition-types-signification-et-exemples/
Consulté le 9 mars 2025.

9 Elastic — https ://www.elastic.co/fr/what-is/sentiment-analysis
Consulté le 13 mars 2025.

10 Université de Biskra — http ://thesis.univ-biskra.dz/5581/1/these%20doc.pdf
Consulté le 18 mars 2025.

11 Amazon Web Services — https ://aws.amazon.com/fr/what-is/recurrent-neural-network/
Consulté le 22 mars 2025.

12 Devoteam — https ://france.devoteam.com/paroles-dexperts/aller-plus-loin-en-deep-learning-avec-les-reseaux-de-neurones-recurrents-rnns/
Consulté le 27 mars 2025.

13 Intelligence Artificielle School — https ://www.intelligence-artificielle-school.com/ecole/technologies/gated-recurrent-unit/
Consulté le 1er avril 2025.

14 Intelligence Artificielle School — https ://www.intelligence-artificielle-school.com/ecole/technologies/abert-modele-nlp/
Consulté le 5 avril 2025.

15 Amazon Web Services — https ://aws.amazon.com/fr/what-is/gpt/
Consulté le 9 avril 2025.

16 Hugging Face — https ://huggingface.co/docs/transformers/model$_doc/xlnet$
$Consultéle12avril2025.$

17 UQAM — https ://archipel.uqam.ca/16421/1/M17567.pdf
Consulté le 15 avril 2025.

18 Isidoros Perikos et Athanasios Diamantopoulos (2024), Explainable Aspect-Based Sentiment Analysis Using Transformer Models
Consulté le 18 avril 2025.

19 Erik-Robert Kovacs, Liviu-Adrian Cotfas et Camelia Delcea (2024), January 6th on Twitter : measuring social media attitudes towards the Capitol riot through unhealthy online conversation and sentiment analysis
Consulté le 22 avril 2025.

20 Merinda Lestandy , Abdurrahim, Amrul Faruq et Muhammad Irfan (2025), A comparative analysis of transfer learning models on suicide and non-suicide textual data
Consulté le 25 avril 2025.

21 Labellerr — https ://www.labellerr.com/blog/roberta-a-robustly-optimized-bert-pretraining-approach/
Consulté le 30 avril 2025.

22 DataScientest — https ://datascientest.com/google-colab-tout-savoir
Consulté le 3 mai 2025.

23 DataScientest — https ://datascientest.com/python-tout-savoir
Consulté le 6 mai 2025.

24 DataScientest — https ://datascientest.com/pandas-python-data-science
Consulté le 9 mai 2025.

25 DataScientest — https ://datascientest.com/numpy
Consulté le 12 mai 2025.

26 Datarockstars — https ://www.datarockstars.ai/pytorch-la-bibliotheque-de-deep-learning-de-meta/
Consulté le 16 mai 2025.

27 PyPI — https ://pypi.org/project/transformers/
Consulté le 20 mai 2025.

28 DataScientest — https ://datascientest.com/tout-savoir-sur-scikit-learn
Consulté le 23 mai 2025.

29 HE-ARC — https ://he-arc.github.io/livre-python/random/index.html
Consulté le 26 mai 2025.

30 Analytics.fr — https ://analytics.fr/definitions/tqdm/
Consulté le 30 mai 2025.

31 HE-ARC — https ://he-arc.github.io/livre-python/matplotlib/index.html
Consulté le 2 juin 2025.

32 Seaborn — https ://seaborn.pydata.org/
Consulté le 5 juin 2025.

33 SlideTeam — https ://www.slideteam.net/blog/top-10-des-modeles-powerpoint-positifs-et-negatifs-avec-des-echantillons-et-des-exemples ?lang=French
Consulté le 10 juin 2025.

34 Yao Fu, Biao Huang, Yujun Wen et Pengzhou Zhang (2024), FDR-MSA : Enhancing multimodal sentiment analysis through feature disentanglement and reconstruction.
consulté le 24 juin 2025.

# Résumé

Ce mémoire porte sur l'analyse des sentiments sur les réseaux sociaux, en ciblant deux phénomènes linguistiques implicites : le **sarcasme** et la **double négation**, difficiles à traiter même pour les modèles avancés de traitement du langage.

Deux systèmes fondés sur RoBERTa-base ont été développés. Le premier système détecte le sarcasme via un fine-tuning de RoBERTa-base (F1-score global : 81,69%).Le second reconnaît la double négation à l'aide d'un modèle hybride combinant RoBERTa, GRU bidirectionnel et mécanisme d'attention, atteignant une précision de 96,5%.

Un troisième modèle, entièrement personnalisé, a été conçu pour le sarcasme : architecture sur mesure, tokenizer dédié et entraînement optimisé. Il atteint 95,43 % d'exactitude et un F1-score pondéré de 95,39 %, surpassant largement les modèles préentraînés.

Ces résultats montrent l'intérêt de modèles spécialisés pour mieux capter des structures linguistiques complexes, au-delà des limites des approches génériques.



**Mots-clés :**
*Analyse de sentiments, Sarcasme, Double négation, TALN, Apprentissage profond.*



# Abstract

This thesis focuses on sentiment analysis on social media, targeting two implicit linguistic phenomena : **sarcasm** and **double negation**, which challenge even advanced NLP models.

Two systems based on RoBERTa-base were developed. The first system detects sarcasm via a fine-tuning of RoBERTa-base (overall F1-score : 81.69%). The second identifies double negation using a hybrid model combining RoBERTa, bidirectional GRU, and attention mechanism, achieving an accuracy of 96.5%.

A fully customized model was also built for sarcasm detection, with a tailored architecture, tokenizer, and optimized training. It achieved 95.43% accuracy and a 95.39% weighted F1-score, outperforming pretrained models.

The results highlight the value of specialized architectures to address complex linguistic structures beyond generic models.



**Keywords :**
*Sentiment analysis, Sarcasm, Double negation, NLP, Deep learning.*