**Mémoire | cybersécurité offensive et intelligence artificielle.**

**Version 3.3**

**Date : 2/01/2025**

**Rédacteur : Théophile Crespo Arredondo  
Master : Cyber**

**.00**

Une image contenant texte, Police, Graphique, capture d’écran

Le contenu généré par l’IA peut être incorrect.Contenu du document

Table des matières

[Résumé : 3](#_Toc198142059)

[English summary : 4](#_Toc198142060)

[Mot clés: Phishing, Naïve Bayes, large langage models (LLM), DistillBERT 4](#_Toc198142061)

[Introduction : 5](#_Toc198142062)

[Explication des IA et de Naïve Bayes appliqués à la détection de phishing : 6](#_Toc198142063)

[Collecte et Préparation des Données : 9](#_Toc198142064)

[Implémentation des Modèles et Expérimentation : 13](#_Toc198142065)

[Analyse des Résultats et Comparaison : 14](#_Toc198142066)

[Discussion et Implications : 17](#_Toc198142067)

[Conclusion et Perspectives : 18](#_Toc198142068)

[ATTESTATION PLAGIAT 21](#_Toc198142069)

[Bibliographie : 22](#_Toc198142070)

# Résumé :

Ce travail de recherche étudie l’efficacité des méthodes de détection de phishing en comparant deux méthodes. La première est basée sur l’algorithme naïve bayes et la deuxième quant à elle est basé sur l’utilisation du large langage model DistillBERT. Face à la recrudescence du phishing qui a accompagné l’avènement de l’intelligence artificielle en particulier les larges langages model. Il est primordial d’adapter et d’améliorer nos défenses.

Les deux approches ont été évalué sur un même corpus de mail créer à partir de deux bases de données connues dans le domaine de recherche contre le phishing. J’ai donc créé mon corpus en combinant Enron Email Dataset pour les emails légitimes et le Nazario Phishing Corpus pour les emails de phishing. La première méthode (naïve bayes) qui a beaucoup été utilisé pour la classification de mail frauduleux est simple et peu couteuse tout en ayant de bonnes performances sur des jeux de données équilibrés (F1-score de 97 % (cf. Tab. 1)). Malheureusement elle chute face aux situations réalistes (recall de 53% pour le phishing lorsque le jeu de donnée est déséquilibré (cf. Tab. 3)).

À l’inverse, DistillBERT a fait preuve de robustesse en maintenant un haut niveau de performance sur les jeux de donnée équilibrés comme déséquilibré avec un recall proche de 100% (cf. Tab. 2, Tab. 4).

Ces résultats confirment l’intérêt d’utiliser les larges langages models dans nos systèmes de détection de phishing même s’il faut prendre en compte le cout en temps et en ressource que demande cette approche. Nous pouvons donc dire que la meilleure méthode dépendra du contexte applicatif. Ils devront choisir une solution une fonction de leurs ressources et du niveau de sécurité dont ils ont besoin.

# English summary:

This research studies the effectiveness of phishing detection methods by comparing two methods. The first is based on the naive Bayes algorithm and the second is based on the use of the DistillBERT large language model. Faced with the augmentation of phishing that has accompanied the democratisation of artificial intelligence, in particular large langage model. It is vital to adapt and improve our defences.

The two approaches were evaluated on the same corpus of emails created from two well-known databases in the field of anti-phishing research. I therefore created my own corpus by combining the Enron Email Dataset for legitimate emails and the Nazario Phishing Corpus for phishing emails. The first method (naive bayes), which has been widely used to classify fraudulent emails, is simple and inexpensive while performing well on balanced datasets (F1-score of 97% (cf. Tab. 1)). Unfortunately it falls down in realistic situations (53% recall for phishing when the dataset is unbalanced (cf. Tab. 3)).

Conversely, DistillBERT demonstrated robustness by maintaining a high level of performance on both balanced and unbalanced datasets, with recall close to 100% (cf. Tab. 2, Tab. 4).

These results confirm the value of using large language model in our phishing detection systems, even if we have to take into account the time and resource costs involved in this approach. We can therefore say that the best method will depend on the application context. They will have to choose a solution based on their resources and the level of security they require.

# Mot clés: Phishing, Naïve Bayes, large langage models (LLM), DistillBERT

# Introduction :

De nos jours avec toutes les avancées dans la cybersécurité nous arrivons à faire des systèmes de plus en plus robustes et complexes. Ces avancées ont rendu attaque de plus en plus compliqué à effectuer à tel point que de nos jours la plus grande source de faille est l’humain. Bien que le risque zéro n’existe pas nous évoluons dans un monde ou le moyen le plus simple pour attaquer un système est de chercher les erreurs humaines qu’elles soient accidentelles comme une erreur d’attention, qu’elles soient due à la négligence ou pour des raisons de facilité. C’est pour cette raison que même aujourd’hui avec nos avancées technique, le types d’attaques le plus courant et le plus efficace reste l’ingénierie sociale. Les attaques qui visent à tromper les utilisateurs pour leur soutirer des informations sensibles comme leurs identifiants, leurs mots de passe ou encore leurs coordonnées bancaires. Pour cela l’ingéniosité des attaquants n’a pas de limite et les techniques de manipulations utilisées sont chaque jour plus nombreuses se font passer pour des figures d’autorités ou des personnes de confiances, utilisent des informations volées pour gagner la confiance des victimes tout en leur mettant la pression pour les pousser à commettre des erreurs. Dans ce mémoire nous allons nous étudier une de ces techniques d’ingénierie sociale les plus utilisées. Le phishing ou l’hameçonnage en français est une attaque qui consiste à essayer de piéger la victime en se faisant passer pour un organisme que la victime connait (sa banque, la Caf, etc.) dans un mail piège. Ce mail vous demande généralement de confirmer des informations ou de cliquer sur un lien piégé pour vous extorquer vos informations sous différents prétextes comme la mise à jour d’informations, ou pour régler un incident et le tout urgemment (dans le but de vous mettre sous pression). Selon Heiding et al. (2024), 70% à 80% des cyberattaques utilisent une forme de phishing. Allant du simple mail générique au spear-phishing (le spear-phishing est une forme de phishing ciblé et donc adapté à une personne en particulier ce qui rend l’attaque beaucoup plus dur à détecter pour les victimes). Face à cette menace bien que les méthodes traditionnelles comme les listes noires et les algorithmes probabilistes comme naïve bayes on fait leurs preuves, elles ont aussi montré leurs limites et vulnérabilités. Face à l’évolution de ces attaques nous nous devons de faire évoluer nos techniques défensives également. De nos jours avec l’avènement des modèles d’intelligence artificielle et principalement les larges langages models (LLM) tel que GPT le risque lié au phishing explose. En effet il y a quelques années les mails de phishing étaient relativement facile à détecter car souvent mal fait, avec des fautes d’orthographe et de français. De nos jours ce n’est plus le cas les larges langages models permettent aux attaquants de générer rapidement et facilement de mails de phishing réalistes et sans fautes. Parallèlement à cette augmentation du danger les modèles d’intelligence artificielle pourraient aussi nous permettent de mieux nous défendre face à ces attaques. En effet, les model tel que GPT, BERT, DistilBERT, etc, permettent d’analyser les mails de manière plus fine que les méthodes plus traditionnelles en captant l’intention des mails. Il est donc intéressant d’étudier et de comparer les différentes méthodes de défense afin de pouvoir au mieux se prémunir contre le phishing.

Quelle approche est la plus efficace pour détecter le phishing ? un algorithmes probabiliste simple et explicable comme Naïve Bayes, ou une intelligence artificielle telle que GPT ?

Ce mémoire a pour objectif

* D’expliquer les fondements de l’algorithme Naïve bayes ainsi que ceux des modèles d’intelligence artificielle.
* Récolter des taux de précision avec une expérimentation des deux méthodes entraînées sur le même corpus de mails
* Evaluer leurs résultats selon les métriques standards

Pour cela nous avons structuré notre mémoire en plusieurs parties

* Explication des concepts de l'intelligence artificielle et de l’algorithme Naïve Bayes appliqués à la détection du phishing. (état de l’art)
* Collecte et Préparation des Données
* Implémentation des Modèles et Expérimentation
* Analyse des Résultats et Comparaison
* Discussion et Implications
* Conclusion et Perspectives

# Explication des concepts de l'intelligence artificielle et de l’algorithme Naïve Bayes appliqués à la détection du phishing :

Dans l’introduction nous avons parlé d’une méthode utilisée pour détecter le phishing, du nom de naïve bayes mais en quoi consiste cette fameuse méthode et comment fonctionne-t-elle pour déterminer si un mail est du phishing ou un mail légitime ? Le modèle naïf bayes est un classifieur probabiliste. Ce qui veut qu’il « range » dans des catégories (classes) ses entrées en calculant la probabilité qu’elles appartiennent à cette classe en se grâce à la probabilité qu’elles fassent partie de cette classe sachant les caractéristiques du message (mot, lien, etc,) (probabilités qu’il calcule pendant son entrainement). Ce modèle est fondé sur le théorème de bayes. Il suppose l’indépendance conditionnelle des données d’entrées du modèle. Cela signifie que chaque variable est indépendante si on connaît déjà la classe (Exemple : une fois qu’on sait que le mail est du phishing la présence du mot « urgent » est indépendant de celle du mot « cliquer »). Malgré cette simplification qui ne reflète pas vraiment la réalité ce modèle à démontrer son efficacité dans la classification textuelle, ce qui en a fait une de méthode phare de la détection de spam et de phishing (Greco et al., 2024). Quand on y regarde bien la détection de spam ou de phishing est un problème de classification binaire mail légitime ou phishing.

Pour calculer la probabilité qu’un mail soit du phishing il utilise le théorème de bayes dont la formule est :

**Figure 1**

Une image contenant Police, texte, blanc, nombre

Le contenu généré par l’IA peut être incorrect.  
*https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me\_de\_Bayes*

A étant la classe (phishing/légitime) et B les caractéristiques (mots, lien, etc.).(cf. Fig. 1)

Cette méthode a plusieurs avantages comme la simplicité d’implémentation et la vitesse d’entrainement et de traitement. C’est une méthode transparente nous pouvons donc facilement savoir pourquoi elle a classé un mail dans une certaine classe. Je reste mitigé car cet avantage étant donné qu’il même si le classement est transparent nous n’avons pas vraiment d’explication, juste des passages suspects parce qu’ils se retrouvent souvent dans des mails de phishing. Un des autres avantages et pas des moindres est que cette méthode est efficace même avec peu de donnée. Mais ce n’est pas tout il est aussi peu gourmand en termes de ressources. Bien que cette méthode ait de gros avantage, elle a aussi ses limites. Nous en avons déjà parlé mais il suppose une indépendance qui est rarement vrai. À cela s’ajoute qu’il est vulnérable à l'empoisonnement bayésien. Cela résulte du fait que c’est une méthode connue, les attaquants on apprit ses limites. L’empoisonnement bayésien est une manipulation syntaxique ou l’attaquant utilise sa connaissance de naïve bayes pour utiliser des mots que l’algorithme considère comme des indicateurs de légitimité (la probabilité que le mail soit légitime sachant qu’il contient ce mot est élevé) pour induire le modèle en erreur. Bien que ce modèle ait ses limite sa transparence, sa robustesse et simplicité sur des jeux de données bien préparés font qu’il reste un modèle de référence souvent utilisé comme base de référence pour les études de classification textuelle.

Nous allons maintenant parler des modèles d’intelligence artificielle. Avec l’avancée des larges languages models (LLM) et des circuits neuronaux, nous entendons parler d’intelligence artificielle tout le temps on l’utilise partout mais comment cela fonctionne. Tout le monde utilise les intelligences artificielles en voyant cela un peu comme parler à un génie qui donnerait les réponses par magie. Sans vraiment comprendre comment cela fonctionne. Les méthodes classiques de machine learning supervisé comme Randome forests (ensemble d’arbres de décision), SVM (support vector Machine est un modèle basé sur la géométrie. Il représent ses entrées sur un hyperplan et détermine les regroupements / frontières qui sépare les classes) utilisent des vecteurs de caractéristique (comme pour naïve bayes : lien, mots clés, etc.) pour ranger les mails dans les classes légitime / phishing. Ces modèles sont efficaces mais comme naïve bayes ils sont sensibles à la qualité de leurs entrainements que ce soit les données ou le choix des caractéristiques à surveillé.

Les méthodes plus récentes utilisent le deep learning et des modèles préentraînés comme GTP, BERT, etc. ces modèles on la capacité de comprendre le sens du texte ainsi que le ton et donc de détecter plus facilement les mails frauduleux en captant par exemple le ton d’urgence ou une fausse identité. En effet, leur analyse permet de relever les techniques de manipulation comme le montre Takashi Koide et al. (2024) avec ChatSpamDetector un système qui utilise des large language models pour détecter le phishing qui est chargé de détecter les techniques de manipulation présentent dans les mails. Mais comment est-ce que cela fonctionne. il lit le texte et le transforme en vecteurs numérique (qu’on appelle embeddings) à l’aide de réseau neurones pour capturer le sens des mots. Cet entrainement permet aux réseaux de neurones de produire des vecteurs aux valeurs proches pour les mots dont la signification est similaire et ainsi de représenter le sens du mot pour la machine. Après il utilise des matrices et des opérations pour enrichir ces vecteurs numériques avec des données de contexte comme la position. Ensuite il utilise le mécanisme de self-attention présenté par Ashish Vaswani et al.(2023) pour calculer l’important de chaque mot les uns par rapports aux autres et enrichir encore les vecteurs numériques. Cela permet de comprendre les relations dans la phrase et d’avoir une compréhension beaucoup plus fine du texte. Chaque vecteur est ensuite donné à un réseau de neurones (feed-forward) qui les affine encore. Ce processus d’affinage est répété à travers plusieurs couches (appelé couches de transformer) pour obtenir la compréhension la précise possible du texte. Après être passé dans toutes les couches du transformer les embedding finaux sont enfin prêt à être utilisé. En utilisant son vocabulaire le model va calculer la probabilité pour chaque mot d’être le mot suivant et va produire ainsi mot après mot une réponse au texte (prompte) que nous lui avons soumis.

Nous avons vu que plusieurs méthodes et techniques ont été développé pour lutter contre le phishing et que l’émergence de l’intelligence artificielle avait exacerbé le danger que représentait le phishing. Alors quelle méthode utilisée pour se protéger efficacement. L’utilisation du modèle naïf bayes pour classer détecter le phishing a été beaucoup étudié au cours des années. Par exemple en le combinant avec du traitement du langage naturel (pour transformer les mails en représentation exploitable par le model) l’approche novatrice de Omar Abdelaziz a et al. (2020) a obtenu des résultats d’une grande précisions. Dans son étude nous voyons qu’ils arrivent à obtenir une précision de 96.03% pour les ensembles équilibrés (même proportion des différentes classes) et de 97.21% pour les ensembles déséquilibrés. Nous pouvons donc constater une grande précision malgré les limites que nous avons vu. Bien que nous puissions obtenir de très bons résultats avec cet algorithme il y a bien un piège dans le phishing que naïve bayes a du mal à détecter, les liens frauduleux. Faire la différence entre un lien suspect et un lien légitime est un part essentiel surtout quand on sait que les attaquants peuvent utiliser l’empoisonnement bayésien pour ne laisser que le lien suspect. Malheureusement comme nous le montre l’étude de Tosin Ige (2024) naïve bayes est très mauvais pour détecter le phishing en se basant sur des url. L’étude révèle même que c’est une des pire classifieurs de phishing obtenant une moyenne de 80.4% ce qui est inférieure à tous les autres (random forest, decision tree, SVM, etc.). Ces variations de précision montre bien l’impacte qu’a le contexte sur les performances de naïve bayes.

Bien que naîve bayes ait ses limitations, il reste performant dans bien des cas. Est-­ce que l’intelligence artificielle peut faire mieux et est-ce que ça vaut la quantité de ressource en plus que demande l’intelligence artificielle. Plusieurs études démontrent que les modèles de deep learning peuvent atteindre des niveaux de précision que naïve bayes ne peux atteindre. Par exemple dans une étude mené par Francesco Greco(2024) un modèle de réseau neuronal sans apprentissage par transfert a obtenu 99.78%, le svm eu 99.2% et la régression logistique à obtenue 99.03%. Ce ne sont pas les seuls à avoir obtenue ce genre de résultats les chercheurs Mohammad Amaz Uddin et Iqbal H. Sarker (2024) on eut aussi montré des performances allant jusqu’à 99% avec des techniques de deep learning et de traitement du langage naturel (NLP) plus récemment les Larges languages models (LLM) sont directement utilisés pour la détection de phishing. En effet, des systèmes basés sur des large langage model comme ChatSpamDetector qui utilise GPT-4 ont vu le jour et on démontre une précision impressionnante. Takashi Koide et al.(2024) ont réussi à obtenir une précision de 99.7% avec ChatSpamDetector. L’un des avantages d’utilisé des large langage model (LLM) est qu’ils peuvent à la fois détecter le phishing et donner des explications sur pourquoi ils ont classé le mail comme étant du phishing et donner des conseils ce qui peux en plus de défendre contre le phishing instruire les usagés sur les risques de ce genre d’attaque. Une des choses que nous a appris Takashi Koide et al.(2024) avec ChatSpamDetector est qu’on peut pousser les largues langages models (LLM) à suivre un certain schéma de penser en les faisant réfléchir étape par étape et que cela fourni de meilleurs résultat. Les études de Heiding et al(2024) et celle de Desolda(2024) montre que les résultats des larges langages model (LLM) varie fort en fonction des indications et demande (prompt). Même si leurs performances d’analyse textuelle très impressionnantes elles ne sont pas infaillibles, elles ont leurs limites et il est important de la savoir pour ne pas leurs faire une confiance aveugle et risquée de devenir une victime de phishing pour un faux négatif. Comme le montre Jing Hua(2024) dans son étude ou les models Gemini Advanced and ChatGPT-4 n’ont pas pu détecter le lien frauduleux dans un mail venant d’une source légitime sans mot clé suspect. Bien que les modèles d’intelligence artificielle avancé comme les larges langages model soit très précis ils demandent aussi beaucoup plus de ressources de calcul et peuvent être plus lent. Un des points sur lequel s’accorde une grande partie des études sont que les larges langages model ont augmenter le risque et l’efficacité du phishing. Mazal Bethany et al (2025) nous ont prouvé que les mails générés par des large langage models (LLM) peuvent être plus efficace que des mails écrit par des professionnels de la communication. Ce qui montre évolution de risque que représente le phishing et donc du besoin de faire évoluer nos méthodes de défense.

Collecte et Préparation des Données :

Pour cette étude, j’ai choisi deux corpus différents pour avoir un ensemble de données représentatif de mails légitimes et de mails frauduleux (phishing).

Pour le corpus de mails légitimes j’ai utilisé la base de données Enron Email Dataset (une référence largement utilisé dans ce domaine de recherche) ce corpus contient des communications réelles des employés de l’entreprise Enron avant sa faillite. Nous pouvons trouvez cette base de données qui sous le format csv garde un grand nombre de communication légitime. Cette base de données a pour avantage d’avoir souvent été utilisé dans ce domaine de recherche. Ce qui m’a permis de comparer mes résultats à ceux d’autres études portant sur le même sujet comme celle de Koide et al. (2024) ou celle de Nahmias et al. (2024) qui ont tous utilisé ce corpus afin de former un jeu de donnée pour leurs études.

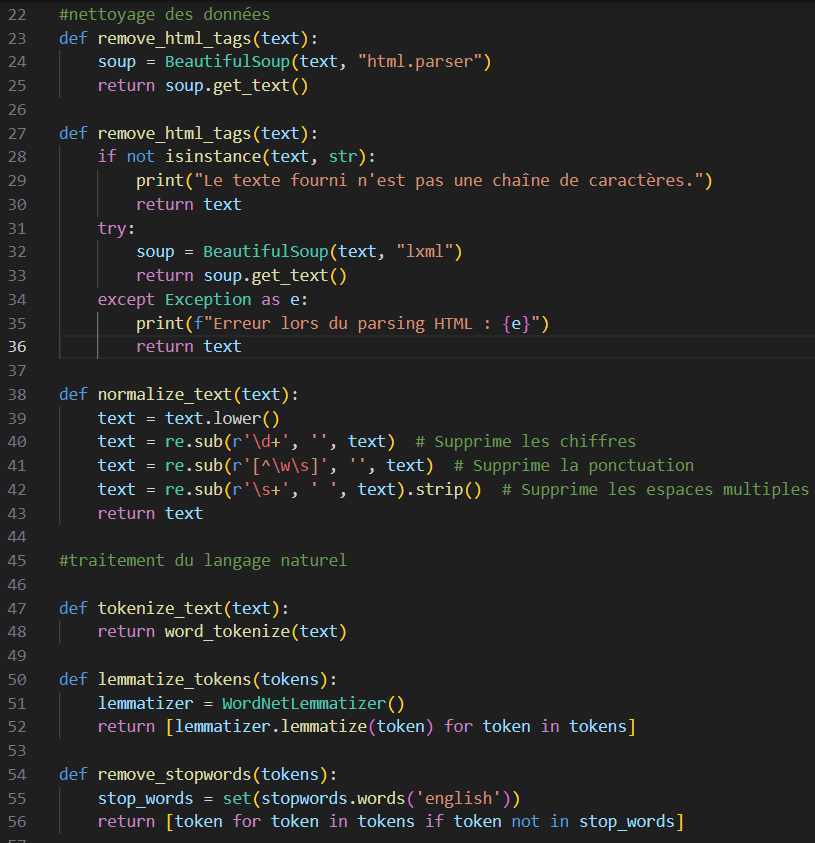
Pour avoir des mails frauduleux (phishing) dans mes jeux de donnée j’ai utilisé le Nazario Phishing Corpus. Une autre base de données connue contenant des recueils de campagnes de phishing sous le format mbox. Comme pour la base de données Enron Email Dataset ce corpus à l’avantage d’être connue et très utilisé pour les études portant sur la détection de phishing me permant ici aussi de comparé mes résultats aux études consulté pendant ce travail de recherche. Par exemple nous avons Yasin, Abuhasan (2016) qui a utilisé ce corpus pour entrainer sont classificateur naîve bayes et obtenue une précision de 94%. Nous avons également déjà abordé plusieurs études qui avaient utilisé ce corpus. Lors de notre état de l’art nous avons parlé des études de Greco (2024) mais c’est loin d’être le seul. Etant un sujet d’actualité nous avons pleins d’étude récentes sur le sujet qui profite de se recueille de mail pour faire leurs jeux de donnée comme Afane (2024) ou Desolda (2024). C’est données ont été combiné pour former les corpus que j’ai utilisé pour entrainé et testé les modèles naîve bayes et Bert.

Une fois que j’ai télécharger ces deux bases de données respectivement sur :

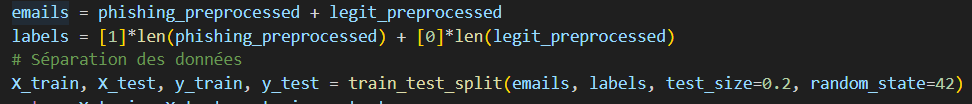
* Enron Email Dataset : <https://www.kaggle.com/datasets/wcukierski/enron-email-dataset>
* Nazario Phishing Corpus: <https://monkey.org/~jose/phishing/>

J’étais en possession d’un fichier csv avec les mails légitimes et une dizaine de fichier .mbox contenant les mails frauduleux. Il me fallait donc les prétraiter pour pouvoir les utiliser de manière efficace avec mes modèles. C’est une étape importante pour ne pas gaspiller des ressources dans le traitement des données mais aussi pour garantir des données de qualitées, épuré des tout ce qui est superflue pour les modèles. Pour cela j’ai procédé en plusieurs étapes. J’ai commencé par extraire de ces fichiers les mails sous la forme de liste de texte brut. La première étape de ce traitement fut de retirer les balises HTLM qui ne servent pas à mes model et qui est surtout utile pour l’humain pour rendre les mails plus faciles à lire. En effet, je ne veux garder que les informations textuelles et enlever toutes les informations non pertinentes qui pourraient biaiser l’entrainement des modèles. Toujours afin de nettoyer mes données de toutes les informations superflues J’ai ensuite « normalisé » le texte. Ce qui veux dire que j’ai mis le texte en minuscules et que j’ai supprimer les chiffres, la ponctuation et les espaces multiples.

Une fois mes données nettoyées j’ai entrepris de segmenter les textes par mot. Pour cela j’ai utilisé la bibliothèque nltk qui à transformer chaque mot en token. Maintenant que j’ai un texte nettoyé et brut ce qui est parfait pour un large langage model comme BERT qui a été pré-entrainé sur du texte brut. Je passe donc au traitement pour pouvoir entrainé naîve bayes au mieux. En effet naive bayes ne comprends pas la grammaire et ne se base que sur la fréquence des mots. J’ai donc entrepris de continuer d’approfondir le traitement pour optimiser ses performances. J’ai donc lemmatisé chaque token grâce à WordNetLemmatizer. Ce qui signifie que j’ai transfomé chaque mot en sa forme canonique. à ce moment j’avais retiré presque toutes les informations non textuelles et ramener les mots à leur forme canonique pour n’avoir que les données pertinentes du texte. Comme naïve bayes est un algorithme probabiliste qui se base sur la fréquence des mots il y a des mots qui tellement utilisé qu’il est inutile de les inclure dans nos données. C’est ce qu’on appel des mot vide ou stop word (le, la, de, du, ce, etc.). je les ai retiré en utilisant stopwords de la bibliothèque nltk.

**Figure 2**  


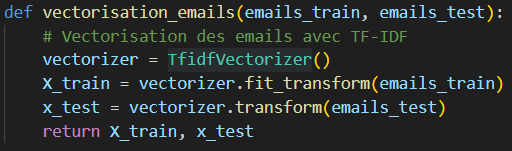
Maintenant que les données sont nettoyées nous devons former nos jeux de données. Pour cela j’ai commencé par fusionner mes deux corpus de mail et stocké la classe de chaque mails (dans la variable labels) afin d’avoir un corpus ayant à la fois des mails légitime et des mails frauduleux. Ensuite nous allons séparer nos données en deux jeux de donnée, un pour l’entrainement et un pour l’évaluation (test). Nous avons séparé le corpus grâce à la fonction train\_test\_split de la bibliothèque sklearn.(cf. Fig 3)

**Figure 3**  


Les données ont été découpé en deux jeu de donné dont la répartition est de 80% pour l’entrainement des modèles et 20% pour les tests. (cf. Fig 3)

Nous pouvons maintenant passer à la deuxième étape du pré-traitement des données. La vectorisation des mails. C’est dans cette partie que nous mettons en forme les données pour qu’elles puissent être interpréter par nos modèles. Pour cela nous allons utiliser une méthode différente pour nos deux modèles. Pour naïve bayes nous allons utiliser la méthode TF-IDF (Term Frequency-Inverse Document Frequency) grâce à TfidfVectorizer de la bibliothèque sklearn (cf. Fig. 4). C’est une méthode qui permet de transformer mon corpus de mail en matrice qui permet de mettre en avant les mots qui aide à classer le texte. Pour réussir ce tour de force elle utilise deux informations importantes. La première est la fréquence dans le texte (Term Frequency). Et la deuxième est la rareté du mot dans l’ensemble du corpus. La formule de Term Frequency-Inverse Document Frequency :

TF-IDF(t,d)=TF(t,d)×IDF(t)  
IDF(t)=log(N /nt​)  
 d = le texte et t = fréquence du mot dans le texte  
N = nombre de texte et nt= nombre de texte ayant le mot t

**Figure 3**  


Pour Bert la méthode est dans transfomer les mots en Embeddings. Cette approche est assez simple. Comme nous utilisons un large languge model déjà entrainé que nous ré-entrainons, il nous de lui donner le texte brut et il s’occupera de la vectorisation. (cf.Fig.5)

Implémentation des Modèles et Expérimentation :

Pour mon expérience j’ai utilisé le corpus de mail que j’ai obtenue en fusionnant en pré-traitant les mail légitime de la base de données Enron Email Dataset de mail frauduleux provenant du Nazario Phishing Corpus. Ensuite lors de mon expérimentation pour naïve bayes j’ai segmenté mon corpus en deux partie (pour l’entrainement 80% et l’évalution/test 20%) que j’ai vectorisés à l’aide de la méthode Term Frequency-Inverse Document Frequency. J’ai implémenté l’algorithme Naïve bayes avec la classe la classe MultinomialNB de la bibliothèque Scikit-Learn. Pour l’entrainement j’ai donné le jeu de donnée d’entrainement et ses labels (tableau contenant la classe de chaque mail dans l’ordre) a la fonction fit mon modèle de naïve bayes. Une fois mon modèle entrainé, je l’ai évalué sur le jeu de donnée d’entrainement grâce à la fonction predict du modèle

**Figure 4**  
Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

Pour l’implémentation du large langage model j’ai utilisé un modèle DistBert pré-entrainé que j’ai entrainé afin de détecter le phishing. Afin de pouvoir comparer les deux modèles je l’ai entrainé sur les mêmes jeux de donnée que naïve bayes. Mais Contrairement à pour lui Bert n’a pas besoin de pré-traitement du style Term Frequency-Inverse Document Frequency car il est capable d’interpréter le texte brut en le transformant lui-même en embedding. Je luis passe donc directement mails découper en token. (nous ne faisons donc pas de lemmatisation et lui laissons les mots vide qui lui servent à bien saisir le contexte du texte) dans des dictionnaires contenants et les mails et les labels de ces mails. L’entrainement a été configuré en utilisant la classe Trainer et TrainingArguments de la bibliothèque transformers. Les paramètres importants que j’ai configurés sont le nombre d’époques (passage du modèle sur les données d’entrainement) et la taille de batch (nombre d’élément traité avant de mettre à jours les poids). Un fois que le trainer est configuré avec le model, les arguments d’entrainement et les jeux de données (dataset), j’ré-entraine (fine-tuner) mon modèle avec la fonction train du trainer. Une fois entrainé à l’instar d’avec naïve bayes je l’évalue sur le jeu de donnée de test avec la fonction prédict du modèle

**Figure 5**  


Analyse des Résultats et Comparaison :

Une fois les prédictions obtenues il me suffit de les comparé aux réponses attendues (les labels du jeu de test) pour évaluer les performances de mes modèles. Pour obtenir cette évaluation de mes modèles j’ai utilisé la fonction classification\_report de scikit-learn qui m’a calculé les métriques standards utilisées pour juger de la qualité d’un modèle de classification. Le retour de cette fonction nous donne un tableau contenant les métriques suivantes :

* Précision : pourcentage des prédiction positives correctes parmi les prédictions positives
* Rappel : pourcentage de prédiction positives correctes parmi les cas positifs
* Score F1 : la moyenne harmonique de la précision et le rappel (métrique très utilisé dans le domaine ou avec proportionnalité inverses)
* Support ; nombre d’élément de chaque classe
* Accuracy : pourcentage de prédiction correcte sur l’ensemble du corpus
* Macro avg : (macro avrage) moyenne des métriques de chaque classe
* Weighted avg : moyenne pondérée de chaque classe (prends en compte le nombre d’élément pour chaque classe)

Suite à nos expérimentations nous avons des tableaux sur un jeu de donnée équilibré (jeu ayant la même proportion de chaque classe) et sur un jeu de donnée déséquilibré (jeu ayant des proportions différentes de chaque classe ce qui est plus réaliste) pour pouvoir comparer les performances des chaque modèles.

Nous allons donc analyser les résultats que nous avons eu pour chacun des jeux de données. Nous allons donc étudier les métriques de chaque évaluation et comparer les moyennes métriques de chaque classe (macro avrage). Après l’évaluation du jeu de donnée équilibré, nous avons obtenu ces résultats grâce à la fonction classification\_report :

**Tableau 1**  
Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Nous pouvons voir que sur un jeu de donnée équilibré naïve bayes à une bonne précision malgré un petit jeux de donné d’entrainement (1000 mails). Nous pouvons constater qu’il est très fiable dans sa détection. En effet il a détecté presque l’intégralité des mails de phishing (phishing recall à 99%). Malheureusement ces prédictions ne sont pas parfaites. En effet il a tendance à trop vite catégorisé un mail comme du phishing ce qui fait qu’il a des faux positifs. Bien que 4% (légitime recall à 96%) puisse paraitre anodin mais à grande échelle cela classe comme du phishing énormément de mail potentiellement important. (cf. Tab. 1)

**Tableau 2**  
Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Nous pouvons tout de suite constater de l’efficacité impressionnant de bert qui parvient presque toujours détecter le phishing (phishing recall a 99%) et à ne se tromper que très rarement (phishing précision à 98%). Mais ce n’est pas la seule catégorie ou il excelle. Bien que dans une moindre mesure (légitime précision à 99% et recall à 98%). Il s’impose donc comme un modèle d’une performance extrême comme le montre son rapport précision/recall (légitime Score F1 à 99% et à 98% pour le phishing). (cf. Tab. 2)

Nous allons maintenant étudier les résultats obtenus avec le jeu de donnée déséquilibré.

**Tableau 3**  
Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Nous voyons tout de suite que les performances ont énormément chuté (Score F1 à 0.69). Nous voyons qu’il n’a pas su détecter presque la moitié des mails de phishing (phishing recall à 53%) malgré que nous lui ayons fourni un plus grand corpus. Nous voyons donc bien les limites de de nayes qui en condition réel (ou les données déséquilibré) obtient de bien moins bonne performance. (cf. Tab.3)

**Tableau 4**  
Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Nous voyons ici que bert est robuste. En effet même contrairement à Naïve bayes il a gardé de très bonne performance une détection presque parfait pour les mail légitime et sa capacité à détecter du phishing n’en a pas été impacté malgré le déséquilibre des classes. On contraire il a presque un score parfait (Le score peut sembler impossible mais ce n’est que que parce nous arrondissont à deux décimale). (cf. Tab. 4)

Nous voulions savoir qu’elle était la meilleure méthode il est donc normal de comparer les deux model. Pour cela nous allons comparer leurs performance grâce moyenne tout en gardant en tête ce que nous a appris l’étude de chacun des résultats obtenue grâce à classification\_report.

Nous avons donc les performances suivantes ;

**Jeu de Données Déséquilibré**

**Tableau 5**

| **Modèle** | **Précision (Macro)** | **Recall (Macro)** | **F1-score (Macro)** | **Accuracy** |
| --- | --- | --- | --- | --- |
| Naïve Bayes | 0.98 | 0.76 | 0.83 | 0.96 |
| BERT (LLM) | 0.99 | 1.00 | 0.99 | 1.00 |

**Jeu de Données Équilibré**

**Tableau 6**

| **Modèle** | **Précision (Macro)** | **Recall (Macro)** | **F1-score (Macro)** | **Accuracy** |
| --- | --- | --- | --- | --- |
| Naïve Bayes | 0.97 | 0.98 | 0.97 | 0.97 |
| BERT (LLM) | 0.98 | 0.99 | 0.98 | 0.98 |

L’expérimentation a révélé que dans dans le jeu de donnée équilibré, les deux modèles affiche de très bonne performance. Mais Bert garde un léger avantage. C’est dans le jeu de donnée déséquilibre et donc plus représentatif que Bert se démarque vraiment de par sa robustesse. Il continue d’afficher haute niveau de performances que n’arrive pas à égaler naïve bayes (différence de près de 25% de recall). (cf. Tab. 1, Tab. 3)

Discussion et Implications :

Ce travail de recherche avait pour but de déterminer quelle méthode était la plus efficace pour détecter le phishing entre un algorithme classique de type Naïve bayes avec une vectorisation TF-IDF (Term Frequency-Inverse Document Frequency) des mails et une méthode basée sur un large langage model préentrainé et ajusté (fine-tuning) comme DistillBERT. Les résultats de nos expérimentations on mit en évidence les mêmes tendances que celles des articles que nous avons étudié lors de notre état de l’art. mais ce n’est pas tout en étudiant les résultats nous avons pu apporter un regard nouveau l’impact de la distribution des données.

Comme le confirme l’étude du Omar Abdelaziz et al (2020) les modèles basés sur naïve bayes sont performant dans la détection de phisghing quand les jeux de données sont équilibrés. Nos résultats nous ont démontrer la même chose avec un F1-score et un recall respectivement de 97% et 98% (cf. tab. 6) mais nous aussi démontré que cette performance chutait drastiquement dans une situation plus réaliste avec un jeu de donné déséquilibré. Le recall qui était de 98% sur le jeu de donnés équilibré tombe à 53% (cf. Tab. 1, Tab. 3). La sensibilité a la distribution de l’algorithme naïve bayes avaient déjà été souligné par Mohammad Amaz Uddin et Iqbal H. Sarker qui avaient eu aussi constaté une baisse de performance lors de leur passage d’un jeu de donnée équilibré à un jeu de donnée déséquilibré.

A l’inverse, les modèles basés sur des larges langage model comme BERT démontre une robustesse impressionnante même face à un jeu de donnée déséquilibré tout en gardant un haut niveau de performance. Nos résultats confirment ceux obtenue par Mohammad Amaz Uddin et Iqbal H. Sarker qui on démontrer que les larges langages models pré-entrainé sur de grand corpus sont meilleurs à capter la subtilité du langage et les schémas caractéristiques du phishing. Ces modèles permettent des détecter des tentatives de phishing même avec un jeu de donnée déséquilibré n’ayant qu’une petite proportion de mail de phishing.

Cette étude à l’instar des études présentées dans l’état de l’art s’accorde sur un point, la capacité des larges langages models à dépasser les anciennes méthodes en capturant le contexte des mots dans leurs représentation vectoriel (embedding). En effet, l’expérimentation que mené pour ce travail de recherche a démontré certaines des limites des méthodes traditionnelle. Comme la sensibilité à la distribution. Cependant, elle a aussi fait ressortir les limitations bien connues de cette nouvelle méthode, la complexité de son déploiement et sont cout élevé que ce soit en termes de temps ou de ressource pour l’entrainement.

Les résultats obtenus renforcent les observations établis par les chercheurs qui ont mené les études présentées tout au long de ce travail de recherche. Tout en apportant une lumière nouvelle sur l’impact concret de la distribution des données. Ils permettent de choisir un la meilleure méthode de détection de phishing en s’adaptant au besoin applicatif.

Conclusion et Perspectives :

Ce travail de recherche m’a permis de démontrer que parmi les deux méthodes que nous avons étudiées, La méthode basé sur l’utilisation d’un large langage model comme bert est plus performante pour détecter de phishing. Sa capacité a capté le contexte et de le stock dans des vecteurs (embeddings) permet d’avoir une compréhension fine du contexte et de détecter avec plus de précision les tentatives de phishing et ce même dans des situations complique avec un petit échantillon déséquilibré. Cependant, Ce travail de recherche m’a aussi appris que même si la méthode de détection la plus efficace pour détecter le phishing est d’utilise un large langage model comme bert, que chaque méthode avait ses avantages et ses désavantages. Le choix de la meilleure méthode dépend surtout du contexte applicatif. Naïve bayes est rapide, simple et peu couteux et tandis que bert est plus performant mais demande plus de temps et de ressource (surtout pendant l’entrainement). Donc si vous avez besoin de protéger un environnement peu sensible avec peu de ressource et de donnée (un environnement à faible risque) alors naïve bayes convient parfaitement mais si vous avez un environnement sensible qui a un besoin de haut niveau de sécurité alors un large langage model comme bert est indispensable malgré son prix et ses demandes en ressource et en temps.

Après avoir analysé mes résultats, j’ai plusieurs voies d’amélioration et d’approfondissement possible qui pourrait améliorer l’efficacité de nos méthodes de détection de phishing. La première consiste à étudier d’autre approcher et surtout des approches hybride pour essayer d’obtenir de meilleure performance. Pendant cette étude nous avons vu que chacun des méthodes avec ces points fort et ses désavantages. Nous pourrions essayer de tirer de meilleure performance en s’appuyant sur un système hybride plutôt que sur un modèle unique. Nous pourrions par exemple utiliser naïve bayes comme un premier filtre rapide et utilisé un large langage model que pour les cas ambigus. Ce qui pourrait nous permettre d’avoir le meilleur des deux monde en étant rapide et économe pour la plupart des cas grâce à naïve bayes et précis même sur les cas ambigus grâce au large langage model.

Un des points importants dans l’études des différentes approches serait d’étudier les performances des modèles spécialisé dans la cybersécurité à la place d’utilisé un large langage modèles généralisé et ajusté (fine-tuning).

Un autre axe d’amélioration important serait l’implémentation et l’étude de l’online learning (apprentissage en temps réel en ligne). En effet avec l’évolution constante des techniques des cybercriminel il est primordial que nos méthodes évolue elle aussi avec les attaques qu’elles doivent endiguer. Afin de permettre à nos méthodes d’évolue il serait intéressant que nos d’utilisé des modèles qui évoluerai en continue de s’affiner grâce à leurs observations des nouvelles données.

Même si ce n’impacte pas directement les performances l’un des axes d’amélioration que je voudrais aborder est la transparence. L’un des désavantages à utiliser un modèle d’intelligence artificielle comme Bert, c’est que c’est un peu des boites noirs même si on sait de quoi ils sont composés, comment ils arrivent à un résonnement en particulier reste souvent obscure. Alors afin d’augmenter la confiance des usagers tout en leurs prodiguant des explications et des conseils, il serait intéressant de nous inspirer de Francesco Greco et al.(2024) et d’utiliser des techniques d’ia explicable (XAI) post-hoc (technique ou le large langage model justifie son choix par un texte après son raisonnement) pour rendre nos modèles plus compréhensibles comme utilisé LIME (Local Interpretable Model-agnostic Explanations test le système en modifiant le texte donnée à ton système pour déterminer son raisonnement) ou SHAP (SHapley Additive exPlanations qui calcule l’importance de chaque caractéristique).

Pour conclure, Bien que les larges langages models ont fait preuve de performance impressionnante dans la détection du phishing. L’avenir de la lutte contre le phishing repose certainement sur des solutions hybrides qui allierais la précisions et rapidité tout en restant économe en ressources.

# ATTESTATION PLAGIAT

Je soussigné, « Nom et Prénom », étudiant à la FGES, durant l’année universitaire « 2024/2025 »

certifie que le(a) présente livrable du travail de recherche « titre du document », est strictement le

fruit de mon travail personnel, de synthèse et d’analyse.

Toute citation (articles, livres, mémoires, documents d’entreprises, sources Internet, …) est

formellement notée comme telle, explicitée et référencée dans le corps du texte et en bibliographie.

Tout tableau ou modèle (photos et illustrations diverses) est dûment cité s’il est emprunté à un auteur

ou cité en source s’il est adapté.

Tout manquement à cette Charte de non-plagiat entraînera la suspension de l’évaluation du livrable,

une notation égale à 0, et la convocation devant le conseil de discipline de l’école.

Fait, à \_\_\_\_\_\_ le \_\_\_\_\_\_\_\_

Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Bibliographie :

Abdelaziz O., Deb S., Hodhod R., Ray L. (2020), A Novel Phishing Email Detection Algorithm based on Multinomial Naive Bayes Classifier and Natural Language Processing, Scitepress.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. (2017), Attention Is All You Need, Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), pp. 5998–6008.

Chataut R., Gyawali P., Usman Y. (2024), Can AI Keep You Safe? A Study of Large Language Models for Phishing Detection, Computing and Communication Workshop and Conference.

Sadasivan V.S., Kumar A., Balasubramanian S., Wang W., Feizi S. (2023), Can AI-Generated Text Be Reliably Detected?, arXiv.org.

Koide T., Fukushi N., Nakano H., Chiba D. (2024), ChatSpamDetector: Leveraging Large Language Models for Effective Phishing Email Detection, arXiv.org.

Desolda G., Greco F., Viganò L. (2024), APOLLO: A GPT-based Tool to Detect Phishing Emails and Generate Explanations that Warn Users, arXiv.org.

Heiding F., Schneier B., Vishwanath A., Bernstein J., Park P.S. (2024), Devising and Detecting Phishing Emails Using Large Language Models, IEEE Access.

Greco F., Desolda G., Carelli A., Esposito A. (2024), David versus Goliath: Can Machine Learning Detect LLM-Generated Text? A Case Study in the Detection of Phishing Emails, Italian Conference on Cybersecurity.

Patel H., Rehman U., Iqbal F. (2024), Evaluating the Efficacy of Large Language Models in Identifying Phishing Attempts, arXiv.org.

Nanayakkara P., Kim H., Wu Y., Sarvghad A., Mahyar N., Miklau G., Hullman J. (2024), From Chatbots to Phishbots?: Phishing Scam Generation in Commercial Large Language Models, IEEE Symposium on Security and Privacy.

Hua J., Wang P., Lutchkus P. (2024), How Effective Are Large Language Models in Detecting Phishing Emails?, Issues in Information Systems.

Ige T., Kiekintveld C., Piplai A., Wagler A., Kolade O., Matti B.H. (2024), An Investigation into the Performances of the Current State-of-the-Art Naïve Bayes, Non-Bayesian and Deep Learning-Based Classifier for Phishing Detection: A Survey, arXiv.org.

Nahmias D., Engelberg G., Klein D., Shabtai A. (2024), Prompted Contextual Vectors for Spear-Phishing Detection, arXiv.org.

Misra K., Rayz J. (2022), LMs go Phishing: Adapting Pre-trained Language Models to Detect Phishing Emails, 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT).

Al-Fayoumi M.A., Odeh A., Keshta I., Aboshgifa A., Alhajahjeh T., Abdulraheem R. (2022), Email Phishing Detection Based on Naïve Bayes, Random Forests, and SVM Classifications: A Comparative Study, ResearchGate.

Bethany M., Galiopoulos A., Bethany E., Bahrami Karkevandi M., Vishwamitra N., Najafirad P. (2024), Large Language Model Lateral Spear Phishing: A Comparative Study in Large-Scale Organizational Settings, arXiv.org.

Uddin M.A., Sarker I.H. (2024), An Explainable Transformer-Based Model for Phishing Email Detection: A Large Language Model Approach, arXiv.org.

Afane K., Wei W., Mao Y., Farooq J., Chen J. (2024), Next-Generation Phishing: How LLM Agents Empower Cyber Attackers, BigData Congress [Services Society].

Bethany M., Galiopoulos A., Bethany E., Bahrami Karkevandi M., Beebe N., Vishwamitra N., Najafirad P. (2025), Lateral Phishing with Large Language Models: A Large Organization Comparative Study.

Yasin A., Abuhasan A. (2016), An Intelligent Classification Model for Phishing Email Detection, arXiv preprint arXiv:1608.02196.