

IDENTIFICATION DE PARAPHRASES

PROJET FINAL

ETIENNE COLLIN

20237904

ETIENNE.COLLIN@UMONTREAL.CA

GUILLAUME GENOIS

20248507

GUILLAUME.GENOIS@UMONTREAL.CA

FONDEMENTS THÉORIQUES EN SCIENCE DES DONNÉES -
STT3795

SECTION A

ENSEIGNANTS STEFAN HOROI & GUILLAUME HUGUET

UNIVERSITÉ DE MONTRÉAL

Hiver 2024

À remettre le 29 Avril 2024 à 23:59



Table des matières

1. Introduction	2
2. Objectifs	2
3. Analyse exploratoire des données	2
3.1. Source des données	2
3.2. Statistiques descriptives	3
3.3. Résultats	5
3.3.a. QQP	5
3.3.b. MRPC	5
3.3.c. Analyse	6
4. Prétraitement des données	7
5. Méthodologie d'analyse	8
5.1. Transformation des phrases en vecteurs (encodeurs)	8
5.1.a. Doc2Vec	8
5.1.b. CountVectorizer	8
5.1.c. TF-IDF	9
5.1.d. Universal Sentence Encoder	9
5.2. Métriques de distance	9
5.3. MDS, ISOMAP, PCA pour visualisation des vecteurs	10
5.4. Métriques d'évaluation	10
5.5. Modèles d'apprentissage	10
5.5.a. Naïve Bayes	10
5.5.b. Arbre de décision	11
5.5.c. Forêt aléatoire	11
5.5.d. SVM	11
6. Résultats & Comparaison des modèles	12
6.1. Projections des données	12
6.2. Comparaison des modèles	13
7. Conclusion & Discussion	16
8. Contributions des membres de l'équipe	16
9. Annexes	17
9.1. Colaboratory	17
9.2. Types de mots	17
9.3. Métriques d'évaluation	17
9.4. Embeddings	19
9.5. Graphiques de tous les modèles selon leurs métriques d'évaluation	21
Bibliographie	23

1. Introduction

Les tâches de similarité en traitement du langage naturel (NLP) visent à déterminer le degré de ressemblance sémantique entre différents éléments de langage. Ces éléments peuvent être des mots, des phrases, des paragraphes, ou même des documents entiers.

Ce type de tâche est particulièrement utile pour une variété d'applications telle que la recherche de documents ou de sites web selon une requête, la détection de plagiat en identifiant les contenus similaires à d'autres sources, la traduction en différentes langues en identifiant les phrases et les expressions similaires dans les langues source et cible, et les chatbots tels que ChatGPT qui utilisent ce genre de tâches pour comprendre les nuances du langage naturel.

2. Objectifs

Ce projet évalue et compare divers systèmes de classification textuelle capables d'identifier les paraphrases entre phrases, en testant la robustesse des modèles sur les jeux de données QQP et MRPC du corpus GLUE. Des détails sur ces jeux de données sont fournis dans une section ultérieure. De plus, le projet explore si un entraînement biaisé peut mener à des performances biaisées et expérimente avec divers encodeurs pour identifier les plus performants et généralisables.

3. Analyse exploratoire des données

Les données utilisées proviennent du corpus fiable de données GLUE, un benchmark reconnu et maintes fois utilisé en recherche pour évaluer les modèles de langue [1]. Plus spécifiquement, ce corpus de données contient des paires de phrases qui sont soit des paraphrases, soit des phrases sans lien sémantique.

3.1. Source des données

QQP:

Source des données Le jeu de données Quora Question Pairs (QQP) est un jeu de données fourni par Quora en 2017 [2]. Il contient des paires de questions étiquetées comme étant sémantiquement similaires ou non.

Fiabilité Le jeu de données a été utilisé des centaines de fois depuis son apparition juste sur paperswithcode.com dans divers articles de recherches scientifiques. Il peut donc être considéré comme fiable.

Type de données Les données utiles à ce projet sont les phrases qui seront des chaînes de caractères (string) et l'étiquette qui indique un booléen si oui (1) ou non (0) une paire de phrases est une paraphrase de l'une et l'autre.

Quantité Un fichier `train.tsv`, un fichier `dev.tsv` et un fichier `test.tsv`. Le fichier `train.tsv` contient 363 845 paires de phrases étiquetées, le fichier `dev.tsv` contient 390 962 phrases étiquetées et le fichier `test.tsv` contient 1725 paires de phrases non étiquetées. Les données sont donc déjà séparées pour entraîner le modèle d'apprentissage et l'encodeur avec les données dans le fichier `train.tsv`, puis pour évaluer les performances avec le fichier `dev.tsv`. Le fichier `test.tsv` est là pour permettre d'expérimenter sur plus de phrases au besoin.

Attributs Les deux fichiers `train.tsv` et `test.tsv` sont séparés identiquement en 6 attributs. L'attribut `id` qui donne un identificateur unique à la paire de phrases, les attributs `qid1` et `qid2` donnent un identificateur unique à leur phrase respective, les attributs `question1` et

question2 sont les phrases de langage naturel à évaluer et l'attribut `is_duplicate` indique si la paire de phrases est une paraphrase ou non.

MRPC:

Source des données Le jeu de données Microsoft Research Paraphrase Corpus (MRPC) est un jeu de données fourni par Microsoft en 2005 [3]. Il contient des paires de phrases provenant d'articles de journaux. Chaque paire est étiquetée comme étant sémantiquement similaire ou non.

Fiabilité Le jeu de données est fourni par Microsoft, qui est une entreprise reconnue dans le domaine de l'apprentissage machine et du traitement de texte naturel, et le jeu de données a été utilisé des centaines de fois depuis son apparition juste sur paperswithcode.com dans divers articles de recherches scientifiques. Il peut donc être considéré comme fiable.

Type de données Les données qui seront utiles sont les phrases qui sont des chaînes de caractères (string) et l'étiquette qui indique un booléen si oui (1) ou non (0) une paire de phrases est une paraphrase de l'une et l'autre.

Quantité Un fichier `train.txt` et un fichier `test.txt`. Le fichier `train.txt` contient 4076 paires de phrases étiquetées et le fichier `test.txt` contient 1725 paires de phrases étiquetées. Les données sont donc déjà séparées pour entraîner le modèle d'apprentissage et l'encodeur avec les données dans le fichier `train.txt`, puis pour évaluer les performances avec le fichier `test.txt`.

Attributs Les deux fichiers `train.txt` et `test.txt` sont séparés identiquement en 5 attributs. L'attribut `Quality` qui indique si la paire de phrases est une paraphrase ou non, les attributs `#1 ID` et `#2 ID` donnent un identificateur unique à leur phrase respective, les attributs `#1 String` et `#2 String` sont les phrases de langage naturel à évaluer.

Dans les deux corpus de données, le traitement des données manquantes n'est pas nécessaire. Effectivement, étant des « benchmarks », ces jeux de données sont déjà propres afin d'être facilement utilisé pour comparer les résultats des algorithmes utilisés à ceux d'autres expérimentateurs.

En pratique, puisque le jeu de données QQP contient plus de données, il est plus intéressant de l'utiliser pour entraîner les modèles d'apprentissage automatique. Cependant, après plusieurs tentatives, des contraintes matérielles forcent l'utilisation d'un sous-ensemble du corpus total pour l'entraînement. Les algorithmes utilisés rencontrent également différents problèmes et arrêtent tout simplement de fonctionner par manque de mémoire. Ainsi, la décision a été prise d'entraîner les modèles sur le MRPC et de les tester sur le MRPC et sur le QQP. Le tester sur QQP apparaît tout de même intéressant, car la distribution de ses paires de phrases est différente. De plus, il devrait être tout de même pertinent d'entraîner les modèles sur un sous-ensemble de QQP, soit les 62 500 premières paires seulement, car MRPC contient beaucoup moins d'exemples d'entraînement. Ainsi, il sera possible d'observer si un modèle entraîné sur plus de données donne bel et bien de meilleurs résultats.

3.2. Statistiques descriptives

Afin de donner un meilleur aperçu des jeux de données, différentes statistiques descriptives de chacun des jeux de données peuvent être très utiles [4]. Cela permettra de comparer les similitudes et différences dans un même jeu de données entre sa partie train et sa partie test, et entre les jeux de données.

Proportion de phrases sémantiquement similaires L'équilibre des paraphrases dans les jeux de données est important à analyser. Un déséquilibre, avec une prévalence excessive de para-

phrases, peut favoriser la performance du modèle dans l'identification de ces dernières, mais entraver sa capacité à distinguer les non-paraphrases. Ceci peut induire un biais dans l'entraînement du modèle.

Fréquence de la longueur des phrases L'analyse de la longueur des phrases dans les jeux de données révèle leur distribution. Des disparités peuvent indiquer une influence sur la performance du modèle, car des phrases de longueur similaire sont généralement plus sémantiquement similaires. La comparaison entre les jeux de données est pertinente pour évaluer cette influence. De plus, l'encodeur transformant les phrases en vecteurs de dimension fixe pourrait être moins efficace pour les phrases de longueurs extrêmes. Ainsi, la distribution de cette fréquence sur les phrases similaires, non similaires et dans l'ensemble des données sera comparée.

Fréquence de la longueur des mots De façon analogue à la fréquence de la longueur des phrases, il est pertinent d'analyser la fréquence de la longueur des mots, car une grande disparité à l'intérieur du jeu de données d'entraînement va potentiellement venir rendre le choix de l'hyperparamètre de la taille des vecteurs générés par l'encodeur moins efficace. Aussi, une paire de phrases avec une distribution similaire de la longueur de leurs mots sera intuitivement plus probable à être sémantiquement similaire. Il est donc aussi intéressant de comparer la distribution de cette fréquence sur les phrases similaires, les phrases non similaires et globalement.

Fréquence de la longueur moyenne des mots par phrase De façon analogue à la fréquence de la longueur des phrases, les phrases ayant une structure syntaxique et un sens similaire tendent intuitivement à utiliser des mots d'une longueur moyenne comparable. Il apparaît intéressant de comparer alors la distribution de cette fréquence sur les phrases similaires, les phrases non similaires et globalement.

Fréquence des mots vides (*stop words*) L'analyse des mots vides est essentielle, car ils n'apportent pas de sens à la phrase. Filtrer ces mots lors du prétraitement des données est crucial pour éviter d'influencer la capacité des modèles à identifier des paraphrases. Si deux phrases non similaires partagent de nombreux mots vides, le modèle pourrait incorrectement les classifier comme des paraphrases en raison de cette similarité superficielle. En réalité, ces mots vides ne contribuent pas à la signification des phrases, ce qui fausserait la classification.

Top 10 des mots les plus fréquents Identifier les mots les plus fréquents est crucial pour détecter les sujets prédominants dans les jeux de données. Cela peut influencer l'efficacité d'un modèle entraîné sur des données ne traitant pas le même sujet. Avant de calculer leur fréquence, un prétraitement des données, excluant les mots vides et les symboles, est nécessaire. Ce processus est décrit en détail dans la section 4.

Top 10 par fréquence des bigrammes Dans la même idée que les mots les plus fréquents, les bigrammes permettent d'observer les thématiques récurrentes dans un jeu de données en identifiant les paires de mots apparaissant souvent. Parfois, il est plus intéressant d'analyser la signification deux mots à la fois, car tout seul, celui-ci pourrait prendre différentes portées.

Top 10 par fréquence des trigrammes Encore dans la même idée, mais sur les trigrammes, c'est-à-dire sur 3 mots qui se suivent. De nouvelles significations pourraient encore apparaître.

Fréquence des mots par type De manière similaire à l'analyse des mots les plus fréquents, examiner la fréquence des mots par type permet de mettre en évidence les sujets dominants dans un jeu de données. La bibliothèque `scipy` classe les types de mots en 18 catégories dis-

tinctes, détaillées en annexe 10.1. Certains types incluent principalement des noms propres, qui sont significatifs dans le traitement du langage naturel, car ils représentent des entités spécifiques. Bien que dépourvus de sens individuellement, les noms propres ont une importance contextuelle dans une phrase. Il est donc crucial de les analyser séparément et de prêter une attention particulière aux catégories qui les contiennent.

3.3. Résultats

3.3.a. QQP

Données pour l'entraînement (62500) 37.07% de paraphrases

Données pour le test 36.82% de paraphrases

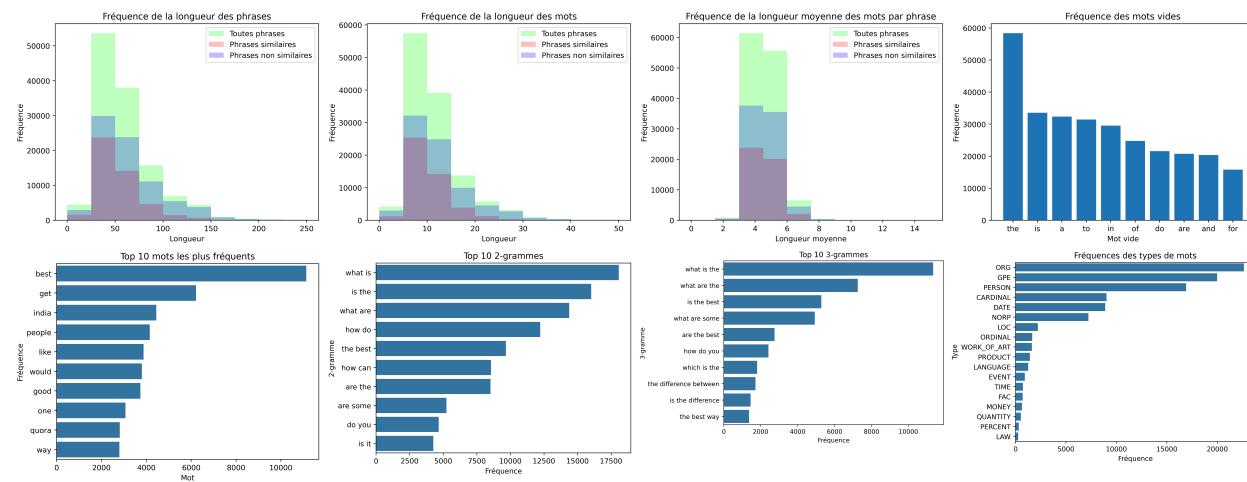


Fig. 1. – Analyse exploratoire de QQP - train.tsv (65 000)

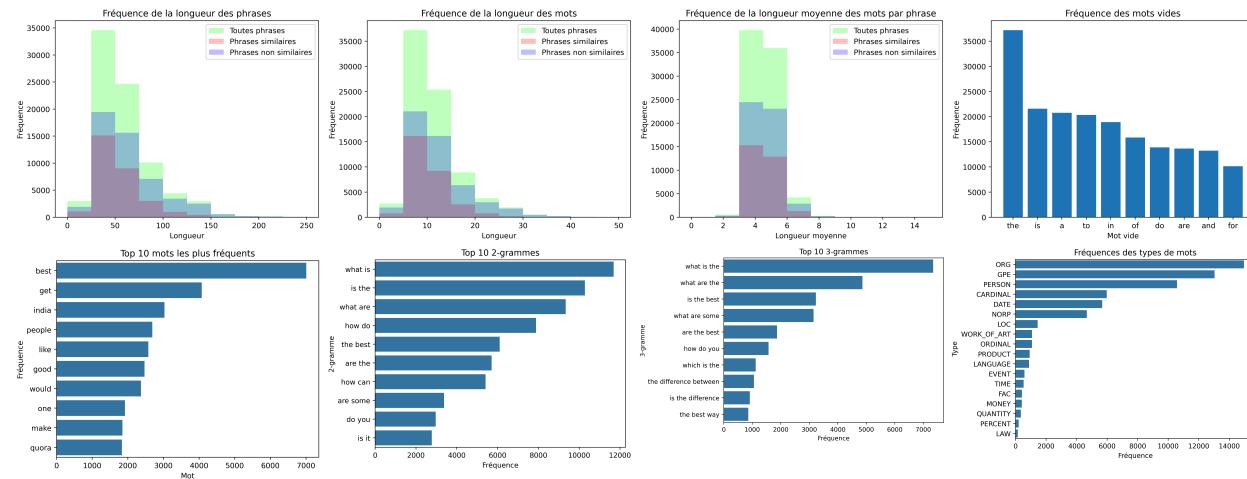


Fig. 2. – Analyse exploratoire de QQP - dev.tsv

3.3.b. MRPC

Données pour l'entraînement (62500) 67.54% de paraphrases

Données pour le test 66.49% de paraphrases

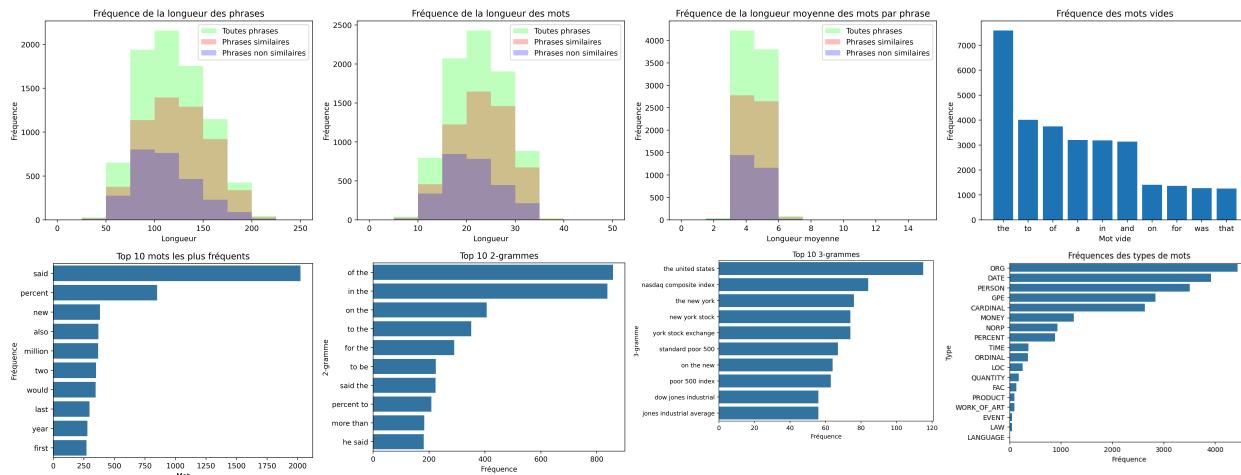


Fig. 3. – Analyse exploratoire de MRPC - train.txt

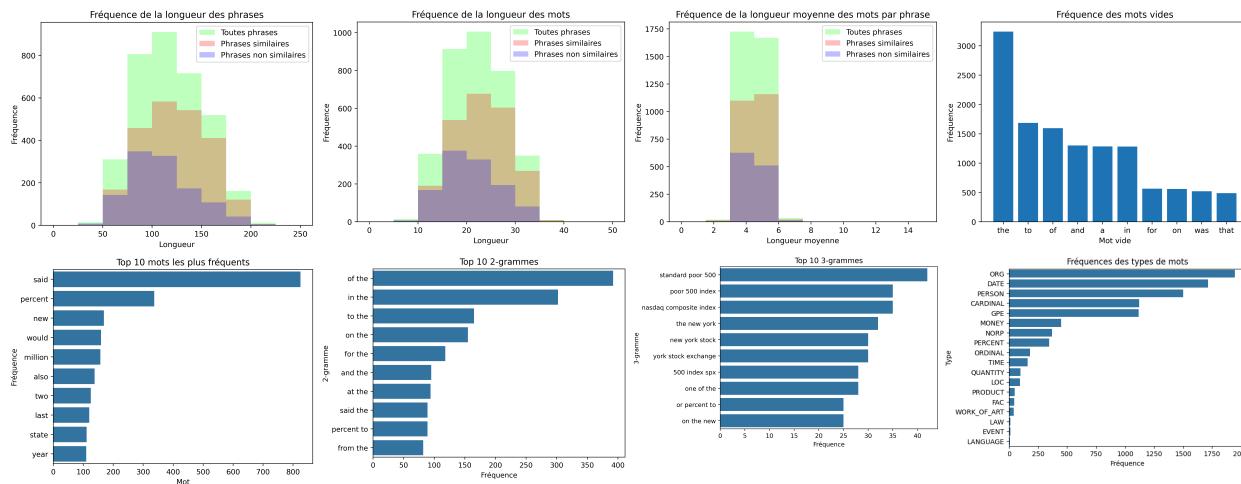


Fig. 4. – Analyse exploratoire de MRPC - test.txt

3.3.c. Analyse

A partir des analyses exploratoires, on observe une similarité dans la distribution des phrases des ensembles d'entraînement et de test des jeux de données QQP et MRPC du corpus GLUE.

Entre les données d'entraînement et de test du QQP, on note une proportion de paraphrases d'environ 37%. Les distributions des longueurs de phrases, des longueurs des mots, des mots vides et des types de mots sont similaires. Les n-grammes semblent majoritairement être des formulations de questions. De plus, les dix mots les plus fréquents sont quasiment identiques. Pour le MRPC, la proportion de paraphrases est d'environ 67%, et on observe également une similarité entre les distributions des longueurs, des mots vides, des types de mots et des mots les plus courants entre les ensembles d'entraînement et de test. Cependant, les n-grammes mentionnent New York et les « stocks ».

Effectivement, il est clair que le QQP et le MRPC ont des proportions de paraphrases opposées : environ 37% pour le premier et environ 67% pour le second. Cette différence peut mener à l'hypothèse selon laquelle un modèle entraîné sur les données du QQP serait probablement plus efficace dans la détection des paires de phrases qui ne sont pas des paraphrases, tandis qu'un modèle

entraîné sur le MRPC pourrait être mieux adapté à l'identification des paraphrases. Cependant, cette hypothèse devra être testée et validée lors de l'analyse des résultats obtenus avec chaque modèle.

Entre QQP et MRPC, on observe des différences marquées dans les n-grammes et les mots les plus fréquents : le premier se compose uniquement de questions, tandis que le second présente des phrases portant sur la bourse. Les types de mots courants sont similaires (ORG, GPE, PERSON, CARDINAL, DATE, NRP), mais « MONEY » est beaucoup plus présent dans MRPC. Ces différences suggèrent que les modèles pourraient avoir des performances moins bonnes lorsqu'entraînés sur un corpus et testés sur l'autre, car ils seraient moins familiers avec le sujet abordé dans le jeu de données non utilisé pour l'entraînement.

La distribution des mots vides et des longueurs moyennes de mots par phrase est similaire entre QQP et MRPC, avec « the » comme mot vide le plus fréquent et une longueur moyenne d'environ 5.

Le QQP présente des valeurs extrêmes non représentatives des autres phrases dans son corpus, comme une phrase de 1151 caractères alors que les autres se situent autour de 50. Ces valeurs ont été exclues des graphiques pour se concentrer sur le comportement général des phrases, mais elles restent présentes dans le jeu de données afin de tester la robustesse des modèles face à des exemples sortant de la norme.

4. Prétraitement des données

Le prétraitement des données est composé de 4 étapes principales:

1. La *standardisation* des données

- Cette étape consiste à uniformiser le texte en le mettant en minuscules et en supprimant les espaces vides en début et en fin de chaîne à l'aide de la méthode `strip()`. Cela permet de normaliser le texte et de réduire les variations dues à la casse ou aux espaces supplémentaires. Effectivement, ces variations n'apportent pas d'information supplémentaire quant à la sémantique de la phrase.

2. La *tokenisation* des données

- La tokenisation consiste à diviser le texte en unités plus petites, généralement en mots ou en sous-chaînes, appelées « tokens ». La fonction `word_tokenize` de la bibliothèque NLTK (Natural Language Toolkit) est utilisée ici pour séparer le texte en mots individuels. Basée sur `TreebankWordTokenizer` et `PunktSentenceTokenizer` afin de spécifier la langue, cette technique permet de traiter chaque mot comme une entité distincte dans le traitement ultérieur.

3. La *lemmatisation* des données

- La lemmatisation est le processus de réduction des mots à leur forme de base, appelée « lemme ». La lemmatisation prend en considération la langue de la source. Par exemple, les mots « ran », « running » et « runs » seraient tous ramenés à leur lemme « run ». Cela aide à normaliser davantage le texte en réduisant les mots à leur forme canonique. La bibliothèque NLTK fournit une classe `WordNetLemmatizer` pour effectuer cette tâche. C'est une version plus intelligente du *stemming* qui, quant à lui, utilise seulement des heuristiques et est moins spécifique. Par exemple, les mots « running » et « runs » seraient ramenés à « run », mais « ran » serait ramené au lemme « ran ». L'idée est que le sens d'une phrase ne dépend pas vraiment de variations de ce genre et que d'éliminer ces variations peut aider le modèle à mieux généraliser.

4. La suppression des *stop words*

- Les « stop words » sont des mots courants qui ne portent généralement pas de sens significatif dans le contexte de l'analyse textuelle, comme « le », « la », « de », « et », etc. Ces mots devraient être supprimés du texte, car ils peuvent introduire du bruit dans les données et affecter négativement l'apprentissage et la généralité d'un modèle. Ici, la liste des mots vides est préalablement définie par NLTK. En supprimant ces mots, l'objectif est de « distiller » les phrases et de ne garder que les mots qui affectent la sémantique.
- Lors de cette recherche, la suppression des caractères non alphanumériques a été considérée. Cependant, ces caractères, tels que les symboles de ponctuation, peuvent grandement influencer la sémantique d'une phrase. Par exemple, considérons les deux phrases suivantes [5]:
 - « Et si on mangeait, les enfants? »
 - « Et si on mangeait les enfants? »

Ainsi, cette suppression n'a donc pas été effectuée.

5. Méthodologie d'analyse

Une variété de métriques d'analyse seront utilisées afin de permettre d'évaluer les performances des modèles générés et de faciliter la compréhension des scores obtenus. De plus, certaines techniques de visualisation telles que PCA, MDS et ISOMAP seront utilisées afin d'observer les relations sous-jacentes aux données utilisées, qui seront ici les phrases transformées en vecteurs. Ces dernières techniques sont choisies pour leur capacité à analyser des données grâce à des métriques de distance autant euclidiennes que non euclidiennes pour capturer les relations dans des données plus complexes.

5.1. Transformation des phrases en vecteurs (encodeurs)

Toutes les techniques présentées ont pour but, au final, de capturer la relation entre les mots d'un document (dans ce cas-ci une phrase) et son contexte (le reste de la phrase). L'objectif étant de déterminer, grâce à ces vecteurs, si la similarité entre une paire de vecteurs produite est suffisamment élevée pour considérer que les deux phrases encodées sont des paraphrases l'une de l'autre. Il est nécessaire de transformer les phrases en vecteur, car les modèles d'apprentissage ne peuvent prendre directement une chaîne de caractères comme entrée.

5.1.a. Doc2Vec

Doc2Vec est une méthode d'apprentissage non supervisée utilisée pour représenter des documents textuels sous forme de vecteurs. Cette méthode étend le concept de Word2Vec aux documents entiers. Elle consiste à associer à chaque document un vecteur dans un espace vectoriel. Doc2Vec utilise deux modèles principaux: DBOW (Distributed Bag of Words) et DM (Distributed Memory). Dans le cas de ce projet, le modèle DM est utilisé. Il tente de prédire le mot suivant à partir d'un contexte partiel et du document en cours [6].

5.1.b. CountVectorizer

CountVectorizer est une méthode de représentation de texte qui convertit un corpus de documents en une matrice où chaque ligne représente un document et chaque colonne représente un terme, et la valeur est le nombre d'occurrences de ce terme dans le document. Sommairement, CountVectorizer crée un vocabulaire de tous les mots uniques dans le corpus, puis transforme chaque

document en un vecteur de longueur égale à la taille du vocabulaire, avec les occurrences de chaque mot comme valeurs dans le vecteur [7].

5.1.c. TF-IDF

TF-IDF est une méthode de pondération couramment utilisée dans le traitement automatique du langage naturel et l'extraction d'informations pour évaluer l'importance d'un terme dans un document relativement à une collection de documents. TF-IDF est calculé en multipliant la fréquence d'apparition d'un terme (TF) dans un document par l'inverse de la fréquence d'apparition du terme dans l'ensemble des documents (IDF). Cela permet de pénaliser les termes qui apparaissent fréquemment dans tous les documents et de mettre en valeur les termes qui sont spécifiques à un document [8].

5.1.d. Universal Sentence Encoder

Le Universal Sentence Encoder est un modèle d'apprentissage profond spécialement conçu pour encoder les phrases en prenant en compte leur contexte global. Il est capable de capturer les nuances sémantiques et les similitudes entre les phrases. Étant déjà entraîné sur un grand corpus de données, le modèle est polyvalent et produit de bons résultats [9].

5.2. Métriques de distance

Sommairement, les métriques de distance qui suivent sont utilisées afin de mesurer la similarité entre des phrases pour éventuellement déterminer si elles sont des paraphrases l'une de l'autre ou non. Voici les principales définitions.

Définition. Voici l'équation de la similarité cosinus:

$$\cos(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \quad (1)$$

Définition. Et l'équation de la distance euclidienne:

$$l_2 = \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{\|x\|_2^2 + \|y\|_2^2 - 2(\langle x, y \rangle)} \quad (2)$$

Avec des vecteurs normalisés, on remarque que ces deux distances dépendent uniquement de $\langle x, y \rangle$ et sont donc proportionnelles.

Définition. Définissons maintenant la distance mahalanobis:

$$\text{mahal}(x, y) = \sqrt{(x - y)\Sigma^{-1}(x - y)^T} \quad (3)$$

Définition. Et définissons la distance de minkowski:

$$l_p = \sqrt[p]{\sum_i (x_i - y_i)^p} \quad (4)$$

5.3. MDS, ISOMAP, PCA pour visualisation des vecteurs

MDS est une méthode non supervisée qui permet de représenter les similarités ou distances entre des objets dans un espace de dimension réduite. En réduisant la dimensionnalité de l'espace des caractéristiques extraites des paires de phrases, MDS peut aider à visualiser la similarité sémantique entre elles, facilitant ainsi la découverte de structures sous-jacentes et la compréhension des relations entre ces paires.

PCA, également non supervisée, trouve les directions principales qui maximisent la variance des données. En appliquant PCA aux paires de phrases, on peut extraire les caractéristiques les plus importantes tout en réduisant la dimensionnalité des données, ce qui facilite leur analyse et leur interprétation.

Quant à ISOMAP, cette méthode de réduction de dimension vise à préserver les distances géodésiques entre les points dans un espace de dimension inférieure. Cela est un atout lorsque les données sont intrinsèquement non linéaires, comme c'est souvent le cas avec des données de texte ou de langage naturel. ISOMAP permet ainsi de capturer les relations de similarité sémantique complexes entre les paires de phrases en préservant leur structure intrinsèque dans un espace de dimension inférieure.

Ainsi, de par leur capacité à découvrir les relations sous-jacentes aux données, à réduire la dimension de ces dernières, et à maximiser l'information conservée lors du processus sans nécessiter de supervision externe, ces algorithmes sont parfaitement adaptés à la tâche.

5.4. Métriques d'évaluation

Les métriques utilisées se basent sur l'utilisation d'une matrice de confusion. Cette dernière est un outil d'évaluation de la performance d'un modèle de classification. Elle permet de visualiser les performances d'un algorithme de classification en comparant les valeurs réelles des données avec les prédictions faites par le modèle.

Les termes TP, TN, FP et FN sont des éléments clés dans une matrice de confusion et sont utilisés pour évaluer la performance d'un modèle de classification. Voir l'annexe Chapitre 9.3 pour les définitions.

Les métriques suivantes d'évaluation de la performance d'un classificateur sont choisies [10]: Précision, Rappel, NPV, Spécificité, Justesse, Score F_1 et MCC. Pour une plus ample définition de ces métriques, voir le Chapitre 9.3.

Un bon modèle pour ce projet voudrait maximiser toutes ces métriques.

5.5. Modèles d'apprentissage

Note. Cette section est majoritairement tirée de [11].

Ce projet se concentre sur l'utilisation de méthodes supervisées avec des données étiquetées. Pour atteindre des performances similaires, les techniques non supervisées nécessiteraient une plus grande quantité de données et des ressources informatiques plus importantes afin d'accommoder ces données supplémentaires.

5.5.a. Naïve Bayes

Le modèle Naïve Bayes est un algorithme d'apprentissage automatique probabiliste basé sur l'application du théorème de Bayes avec une hypothèse d'indépendance forte (et naïve) entre les don-

nées. Il estime les probabilités conditionnelles pour chaque classe et calcule ensuite la probabilité a posteriori pour les données d'entrée nouvelles, l'affectant à la classe la plus probable.

Ce modèle résistant au bruit semble particulièrement adapté à l'identification des paraphrases, car déterminer si deux phrases ont une signification similaire implique de trouver leur sens sémantique tout en ignorant les distractions.

5.5.b. Arbre de décision

Le modèle d'arbre de décision est un algorithme d'apprentissage automatique qui utilise une structure en arbre pour faire des prédictions ou classer des données en fonction de variables d'entrée. Il fonctionne en divisant les données en sous-ensembles plus petits à l'aide d'une série de décisions ou de branches, chacune représentant un test sur une ou plusieurs variables d'entrée. Ce processus se poursuit jusqu'à ce qu'un point d'arrêt optimal soit atteint et ensuite la décision finale est prise en fonction des valeurs attribuées aux feuilles.

Pour décider de l'optimalité d'une scission, on utilise généralement l'indice de Gini ou l'entropie. L'indice de Gini mesure la diversité des échantillons dans un noeud, et plus cet indice est faible, mieux c'est. L'entropie quantifie le degré d'imprévisibilité d'un ensemble de données, et un ensemble avec une entropie faible est considéré comme étant très prévisible. Ainsi, lorsque nous divisons les données en deux groupes distincts, nous cherchons à minimiser l'indice de Gini ou l'entropie pour chaque groupe, ce qui nous aide à choisir la scission optimale.

Ce modèle est adapté pour cette tâche, car il est robuste au bruit et permet d'efficacement sélectionner les caractéristiques apportant le plus d'informations. Ainsi, il est bien positionné pour identifier la sémantique d'une phrase.

5.5.c. Forêt aléatoire

Le modèle de forêt aléatoire représente un ensemble d'arbres de décision appris individuellement à partir des exemples d'entraînement. Ces arbres sont utilisés pour prédire la classe ou la valeur cible d'un exemple de test en procédant le vote majoritaire des prédictions de chaque arbre dans l'ensemble. L'avantage de ce modèle est qu'il résiste bien aux données bruyantes et au surapprentissage, puisque les différents arbres sont moins corrélés entre eux.

Ce modèle est aussi intéressant, car il résiste au bruit et au surapprentissage. Ainsi, il devrait être presque aussi efficace sur un jeu de données avec lequel il s'est entraîné que sur un autre.

5.5.d. SVM

Les modèles SVM (Support Vector Machine) sont des algorithmes d'apprentissage machine qui fonctionnent en cherchant un hyperplan optimal pour séparer les différentes classes de données. Le but est d'optimiser la marge entre ces deux groupes, ce qui permet un modèle plus robuste aux données aberrantes et une meilleure généralisation sur des données inconnues.

Les noyaux (kernels) sont utilisés dans les SVM pour transformer des données non linéairement séparables en un espace de dimension supérieure où elles peuvent être séparées par un hyperplan linéaire. Généralement, on utilise le noyau polynomial (polynomial) et le noyau RBF (Radial Basis Function). Le noyau linéaire est moins flexible que le noyau RBF, mais il est plus rapide à entraîner et nécessite généralement moins de données. Le noyau RBF est plus complexe et offre une meilleure performance sur des ensembles de données complexes, mais son temps d'entraînement est généralement plus long et nécessite plus de données.

Les SVMs offrent une flexibilité dans la gestion des relations de données, qu'elles soient linéaires ou non-linéaires, tout en étant résistants au bruit.

6. Résultats & Comparaison des modèles

6.1. Projections des données

Lorsque les vecteurs d'*embedding* sont calculés pour les phrases, chaque phrase obtient son propre embedding. Afin de procéder à l'apprentissage, les vecteurs de chaque paire de phrases doivent être combinés d'une certaine manière. Dans le cas des résultats présentés, la similarité cosinus est calculée entre chaque paire de vecteurs. Cela génère une liste d'entiers, un par paire de vecteurs, qui peut être utilisé afin de construire une matrice symétrique de distance. Sachant quelles paires de phrases sont des paraphrases, il est donc possible d'utiliser la matrice de covariance de ces distances cosinus ou cette matrice de distance afin de projeter les données en 2 ou 3 dimensions en utilisant PCA, MDS et Isomap. Ces projections, où les points bleus représentent des paraphrases et les points rouges sont des non-paraphrases, permettent d'identifier les relations sous-jacentes aux données entre les paraphrases et les non-paraphrases. Il semble logique que si une relation peut être déterminée à l'oeil nu, alors un algorithme d'apprentissage sera retrouver cette relation.

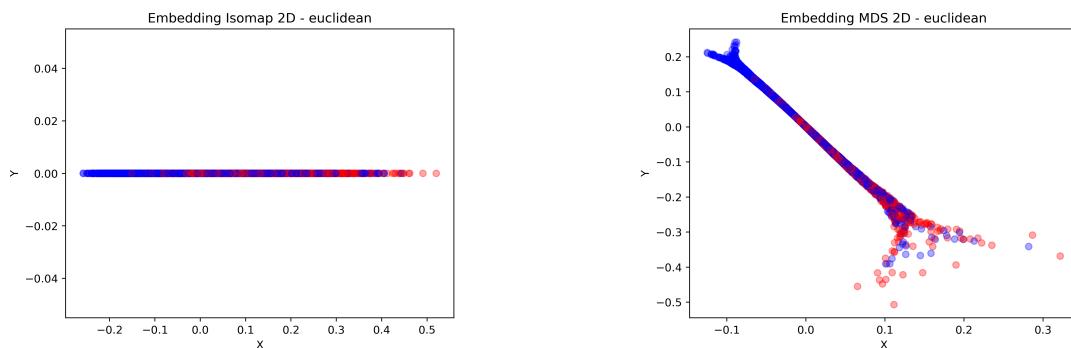


Fig. 5. – *Embeddings* du corpus MRPC

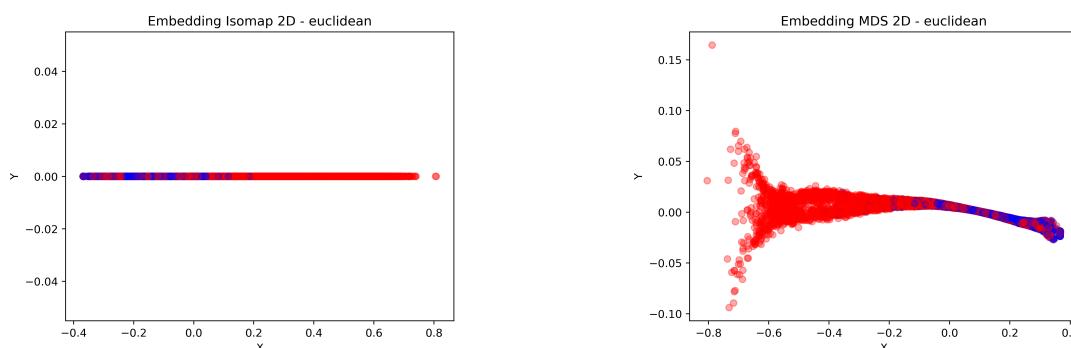


Fig. 6. – *Embeddings* du corpus QQP

Clairement dans les deux corpus de données, les points rouges et bleus des projections semblent s'ammasser d'un côté ou de l'autre des graphiques. Il est donc possible de déterminer si une phrase est une paraphrase d'une autre ou non. Cependant il est important de noter qu'il n'y a pas de coupure entre les points rouges et bleus. Ils finissent par se mélanger avant de se séparer. Encore

une fois, ce comportement peut être expliqué, car effectivement, la langue est un domaine ambigu. Une phrase peut souvent être interprétée de plusieurs manières et, ainsi, il est normal que les points rouges et bleus se mélangent. L'essentiel est qu'il est définitivement possible de séparer les paraphrases des non-paraphrases moyennant une certaine incertitude.

Pour d'autres projections 2D et 3D par PCA, MDS et Isomap, avec la distance Euclidienne, la distance de Mahalanobis et la distance de Minkowski, voir l'annexe au Chapitre 9.4.

6.2. Comparaison des modèles

Il a été suggéré que la source de données d'entraînement pourrait affecter l'efficacité d'un modèle lors du test avec un ensemble de données différent. Il faudra vérifier si certains modèles sont plus résistants à ce problème.

Des tests ont été effectués sur divers encodeurs et modèles, en ajustant les paramètres arbitrairement pour observer d'éventuelles différences de résultats. Un modèle aléatoire a également été inclus pour comparer les performances des modèles par rapport au hasard.

L'analyse sur le jeu de données QQP a posé des difficultés en raison de sa taille, limitant l'entraînement à seulement 62 500 lignes. Seuls Universal Sentence Encoder et Doc2Vec ont pu gérer cette quantité de données, les deux autres demandant trop de mémoire. Les quatre encodeurs ont été utilisés pour un entraînement sur MRPC et testés sur MRPC. Seul le Universal Sentence Encoder a été utilisé pour comparer un entraînement des modèles avec un corpus plus grand (QQP) et testé ensuite sur MRPC et QQP.

Voici donc les résultats :

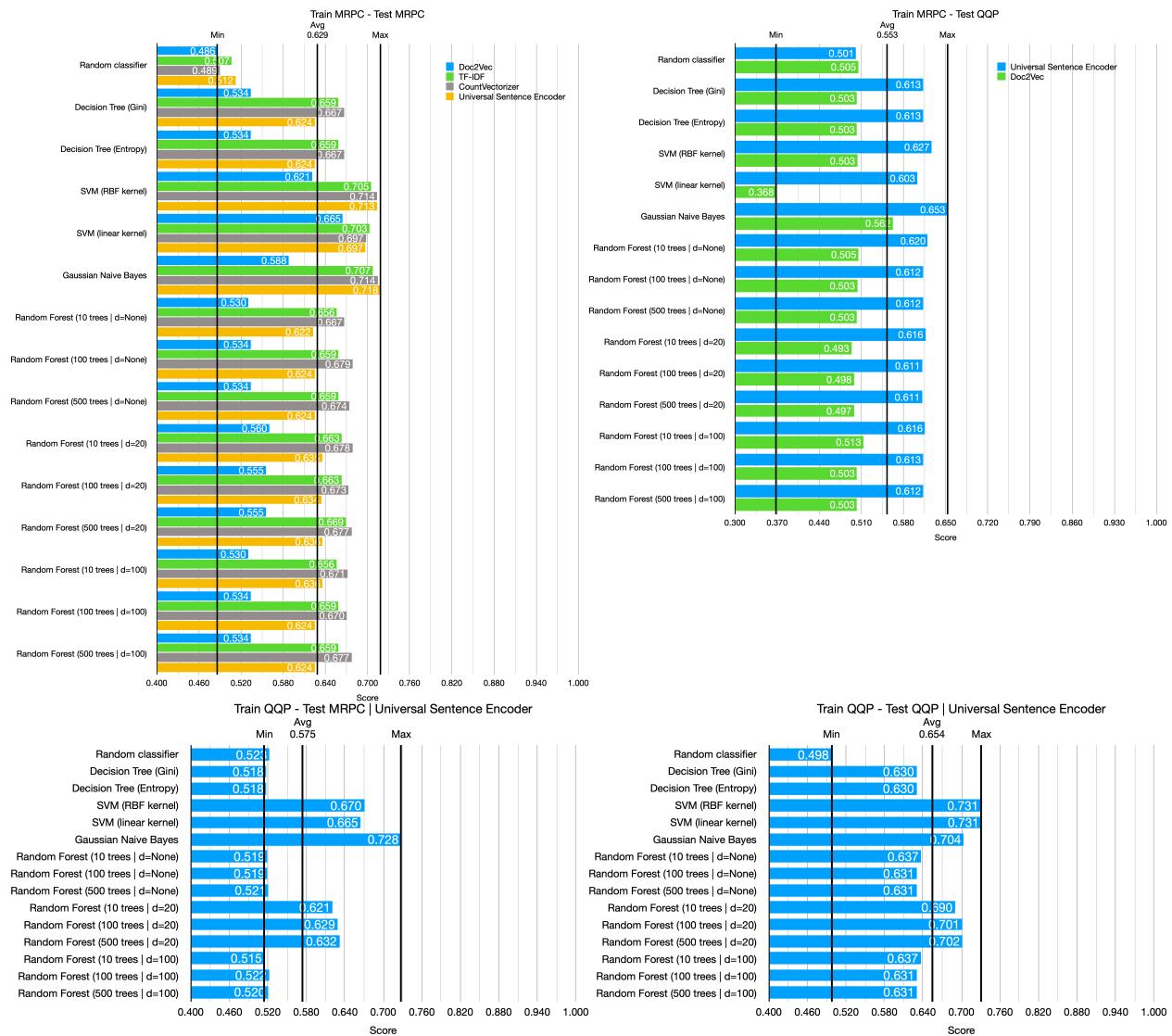


Fig. 7. – Résultats des différents modèles selon leur encodeur et leurs données d'entraînement et de test

Les résultats montrent rapidement que l'encodeur Doc2Vec a obtenu des performances inférieures par rapport aux autres encodeurs, qu'il s'agisse de tests sur le MRPC ou sur le QQP, avec un écart d'environ 10%. Le TF-IDF se situe toujours environ 1% devant le CountVectorizer, mais les deux obtiennent de bonnes performances sur chaque modèle. L'Universal Sentence Encoder est généralement légèrement derrière les encodeurs de TF-IDF et de CountVectorizer, avec un écart d'environ 4%.

Les SVMs avec RBF kernel et Gaussian Naive Bayes se distinguent comme les modèles offrant les meilleures performances, indépendamment de l'encodeur utilisé. Ces modèles permettent à l'Universal Sentence Encoder d'atteindre des niveaux de performance comparables à ceux du TF-IDF et du CountVectorizer.

Dans les SVMs, le kernel RBF présente une légère avance sur le kernel linear, suggérant que les données sont probablement assez bien linéairement séparables, mais que le kernel RBF, plus complexe, permet de mieux distinguer les paraphrases des non-paraphrases.

Leurs bons scores sont probablement dus à la capacité de ces algorithmes à gérer efficacement la sparsité des données et, en outre, au fait que l'hypothèse d'indépendance des données dans le Naive Bayes est probablement bien respectée dans les encodeurs TF-IDF et CountVectorizer. Effectivement, ces derniers sont des modèles basés sur « bags of words » où chaque mot est indépendant.

En général, les données encodées avec Universal Sentence Encoder et entraînées sur le QQP partiellement fournissent de meilleurs résultats pour la plupart des modèles testés sur MRPC et sur QQP.

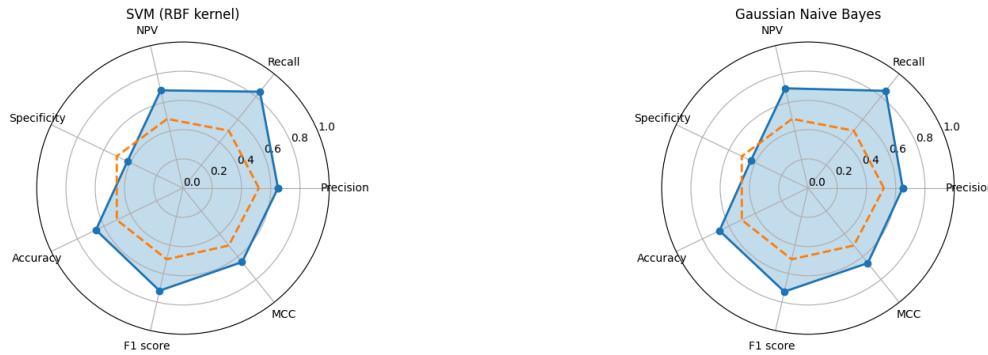


Fig. 8. – Résultats des deux meilleurs modèles selon leurs métriques d'évaluation moyennes (peu importe l'entraînement et le test sur MRPC et QQP, et peu importe l'encodeur).

Les résultats complets avec les métriques d'évaluation sont en annexes Chapitre 9.5, le pointillé orange représente le classificateur aléatoire. Le rappel est la métrique la plus élevée, ce qui signifie que les modèles sont particulièrement efficaces pour identifier toutes les paires de phrases qui sont des paraphrases parmi toutes les paraphrases du jeu de données. Cependant, ils sont moins bons que l'aléatoire en termes de spécificité, c'est-à-dire pour identifier toutes les paires de phrases qui ne sont pas des paraphrases parmi toutes les phrases non sémantiquement similaires du jeu de données. Les autres modèles présentent une meilleure équité dans toutes les métriques en moyenne.

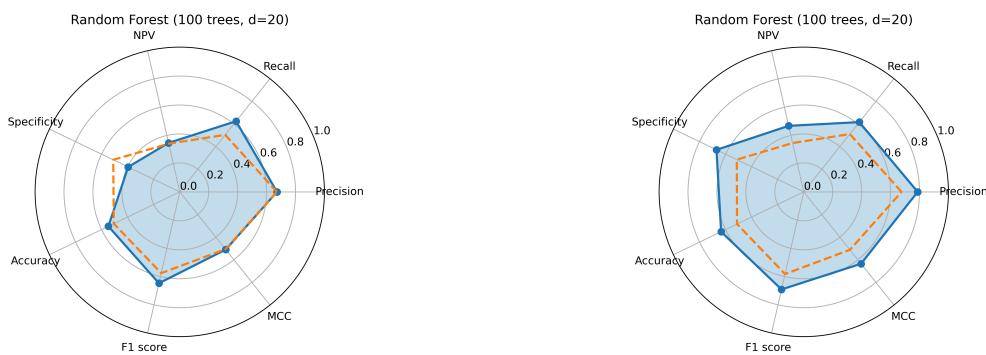


Fig. 9. – Résultats de deux modèles selon leurs métriques d'évaluation sur le dataset MRPC avec à gauche un entraînement sur MRPC et à droite un entraînement sur QQP.

De plus, il semble que les métriques soient biaisées selon le corpus d'entraînement. Par exemple, à la Fig. 9, on observe un faible niveau de spécificité pour le modèle entraîné sur MRPC (graphique de gauche), indiquant qu'il a du mal à correctement identifier les paires non-paraphrases. En revanche, le même modèle entraîné sur QQP présente une spécificité plus élevée (graphique de droite), ce qui

suggère une meilleure performance dans la classification des paires non-paraphrases. Des exemples de ces graphiques sont fournis en annexe. Cela coïncide avec la proportion de chaque jeu de donnée représentée par les paraphrases et non-paraphrases. Enfin, à dans la Fig. 8, on constate que les métriques se compensent lorsque les jeux de données sont combinés en une moyenne.

7. Conclusion & Discussion

En conclusion, à la lumière des résultats, il est recommandé d'utiliser l'encodeur Universal Sentence Encoder avec le noyau SVM RBF ou Gaussian Naive Bayes, car ils offrent une excellente performance dans la captation sémantique des phrases et sont simples à utiliser compte tenu des limitations matérielles.

Il serait intéressant, moyennant du matériel informatique plus puissant, d'entraîner les modèles utilisés sur l'entièreté des corpus QQP et MRPC. Cet entraînement sur une plus grande quantité de données permettrait potentiellement de meilleures performances de généralisation sur les modèles utilisés et une analyse plus complète des jeux de données.

Également, il serait intéressant de tenter de générer des paraphrases. Considérons un modèle bayésien: serait-il possible d'apprendre un modèle qui représente le *likelihood* (une classe sachant la phrase) et un qui apprend le *prior* (une phrase), afin de générer un *posterior* (une phrase sachant que l'on désire une classe « paraphrase »)? Ce processus est utilisé dans d'autres branches de l'apprentissage machine, telles que la génération d'images.

8. Contributions des membres de l'équipe

- **Etienne Collin**

- Objectifs
- MDS, ISOMAP et PCA: théorie et analyse des résultats
- Théorie modèles d'apprentissage
- Encodeurs
- Prétraitement des données
- Conclusion
- Programmation

- **Guillaume Genois**

- Introduction
- Analyse exploratoire des données
- Analyse résultats modèles d'apprentissage
- Métriques de distance
- Métriques d'évaluation
- Programmation.

Au final, nous avons travaillé en paire et la charge de travail a bien été répartie.

9. Annexes

9.1. Colaboratory

Voici deux liens vers les jupyter notebooks (Google Colab) contenant le code utilisé dans ce document:

- `main.ipynb`
 - <https://colab.research.google.com/drive/18691lH98qgGKTvEydDrbmXoN86SmRfp?usp=sharing>
- `pre_analysis.ipynb`
 - <https://colab.research.google.com/drive/1AT-wu4poh2w9wqdXBa8XZelxKmHEDNwC?usp=sharing>

9.2. Types de mots

PERSON Personne, incluant fictive.

NORP Nationalités, groupes religieux ou politiques.

FAC Bâtiments, aéroports, autoroutes, ponts, etc.

ORG Compagnies, agences, institutions, etc.

GPE Pays, villes, états.

LOC Locations non GPE, chaînes de montagnes, étendues d'eau.

PRODUCT Objets, véhicules, nourriture, etc.

EVENT Ouragans nommés, batailles historiques, événements sportifs, etc.

WORK_OF_ART Titres de livres, chansons, etc.

LAW Documents nommés devenus des lois.

LANGUAGE N'importe quel nom de langue.

DATE Date ou période relative ou absolue.

TIME Temps plus petits qu'une journée.

PERCENT Pourcentage, incluant « % »

MONEY Valeurs monétaires, incluant les unités.

QUANTITY Mesures, comme de poids et de distances.

ORDINAL « Premier », « Deuxième », etc.

CARDINAL Chiffres ou symboles n'appartenant à aucun autre type.

Source: [12]

9.3. Métriques d'évaluation

TP (True Positives) Les True Positives représentent les cas où le modèle a correctement prédit qu'une paire est une paraphrase. Plus généralement, il s'agit du nombre d'observations qui appartiennent à une classe positive dans les données réelles et qui ont été correctement classées comme positives par le modèle.

TN (True Negatives) Les True Negatives représentent les cas où le modèle a correctement prédit qu'une paire n'est pas une paraphrase. Il s'agit du nombre d'observations qui appartiennent à une classe négative dans les données réelles et qui ont été correctement classées comme négatives par le modèle.

FP (False Positives) Les False Positives représentent les cas où le modèle a incorrectement prédit qu'une paire est une paraphrase. Il s'agit du nombre d'observations qui appartiennent à une classe négative dans les données réelles, mais qui ont été incorrectement classées comme positives par le modèle.

FN (False Negatives) Les False Negatives représentent les cas où le modèle a incorrectement prédit qu'une paire n'est pas une paraphrase. Il s'agit du nombre d'observations qui appartiennent à une classe positive dans les données réelles, mais qui ont été incorrectement classées comme négatives par le modèle.

Précision (Precision) Mesure la capacité à bien identifier les paires de phrases qui sont des paraphrases parmi toutes les phrases que le modèle a identifiées comme telles.

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5)$$

Rappel (Recall) Mesure la capacité à bien identifier toutes les paires de phrases qui sont des paraphrases parmi toutes les paraphrases dans le jeu de données.

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6)$$

Valeur prédictive négative (Negative predictive value) Mesure la capacité à bien identifier seulement les paires de phrases qui ne sont pas des paraphrases parmi toutes les phrases que le modèle a identifiées comme telles. Cette mesure est analogue à la précision.

$$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}} \quad (7)$$

Spécificité (Specificity) Mesure la capacité à bien identifier toutes les paires de phrases qui ne sont pas des paraphrases parmi toutes les phrases non sémantiquement similaires dans le jeu de données. Cette mesure est analogue au rappel.

$$\text{Spec} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (8)$$

Justesse (Accuracy) Mesure la capacité globalement à bien identifier s'il s'agit oui ou non de paraphrases.

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (9)$$

Score F_1 (F_1 score) Mesure la moyenne harmonique de la précision et du rappel.

$$F_1 = 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} \quad (10)$$

Coefficient de corrélation de Matthews normalisé (Matthews correlation coefficient)

Mesure la capacité balancée entre identifier correctement qu'il s'agit de paraphrases et identifier correctement que non. Pour avoir une valeur du coefficient dans l'intervalle $[0,1]$, la version normalisée qui divise par 2 et ajoute 0.5 au coefficient original permet d'être dans l'intervalle $[-1,1]$.

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{2 \cdot \sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} + 0.5 \quad (11)$$

9.4. Embeddings

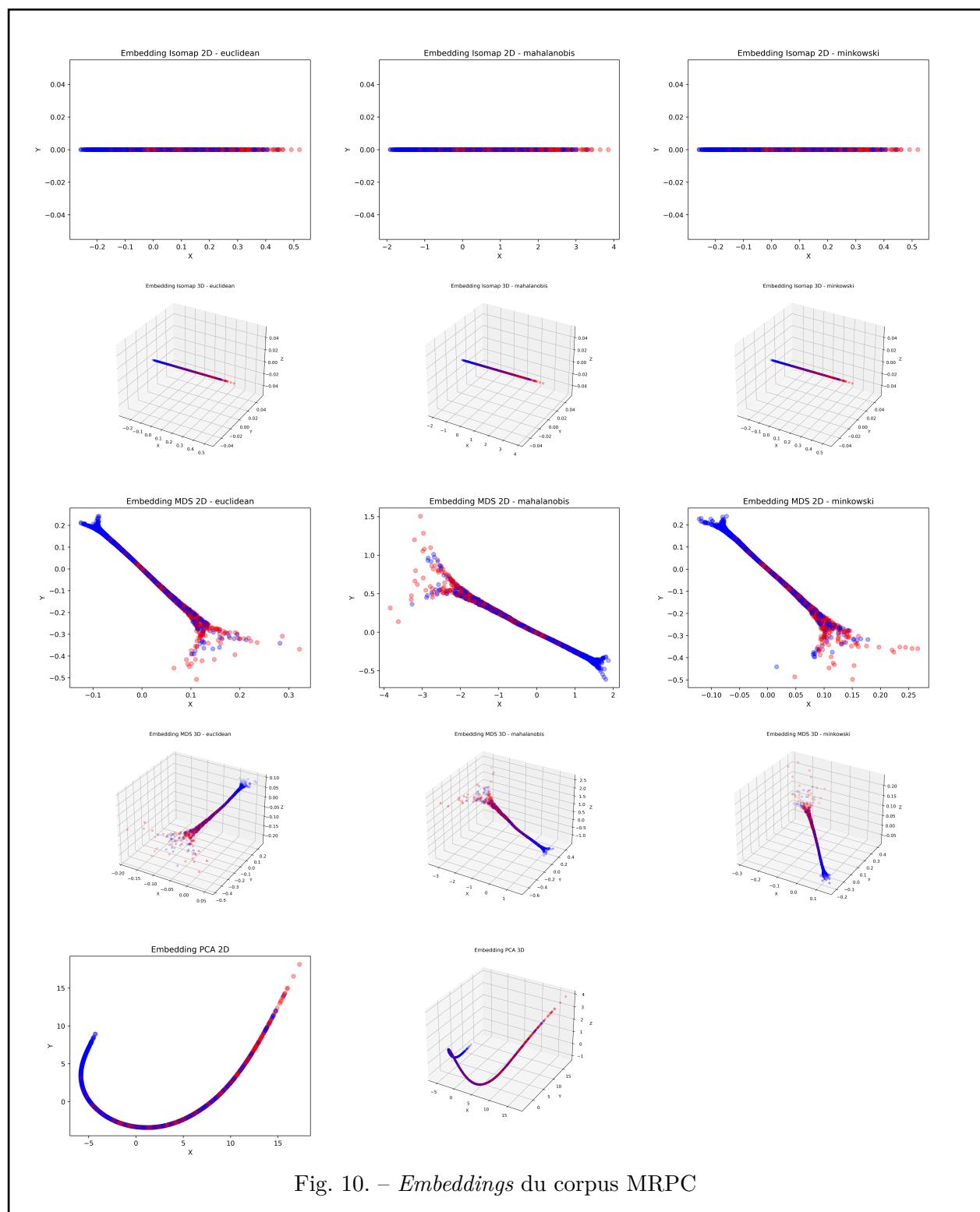
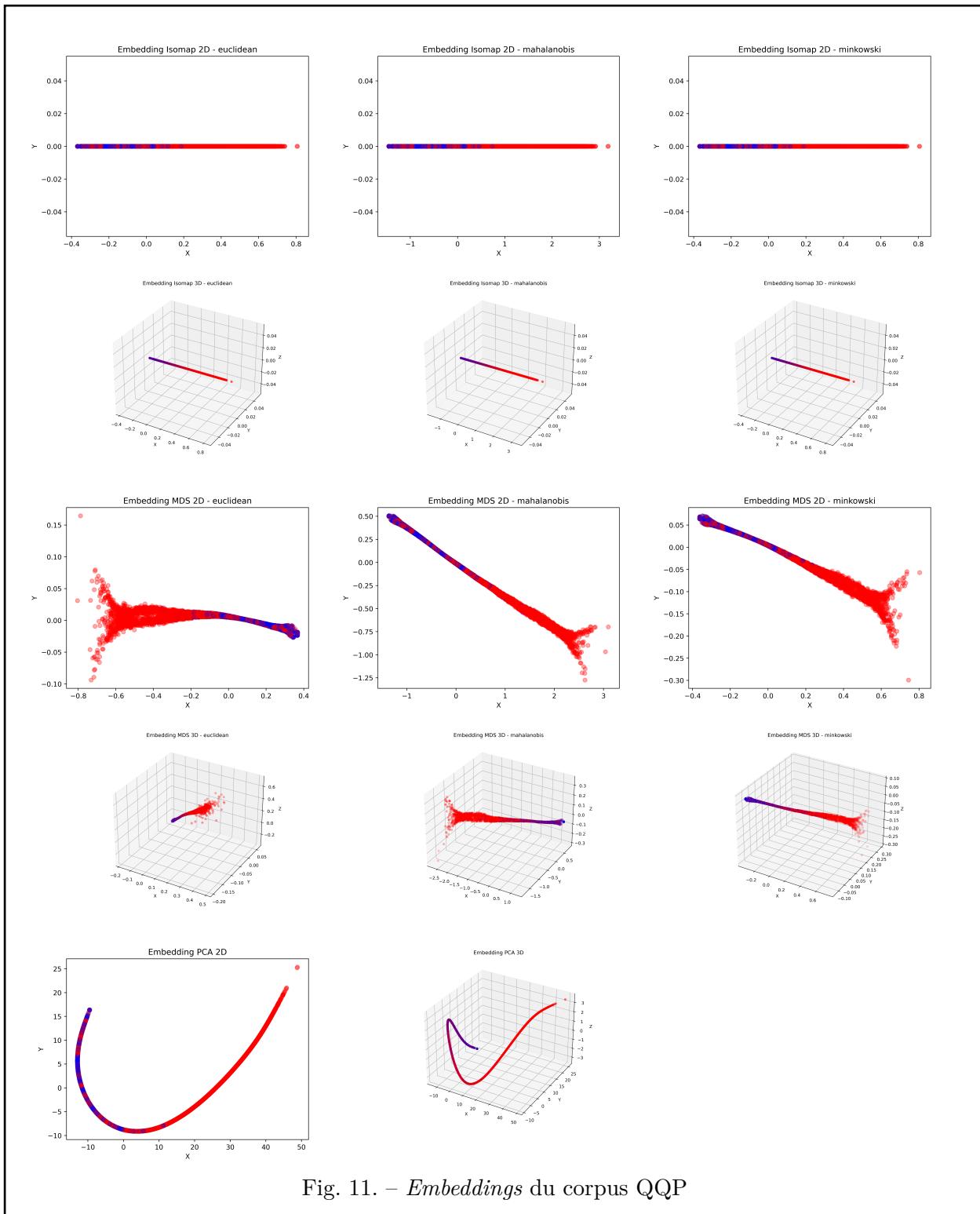


Fig. 10. – *Embeddings* du corpus MRPC

Fig. 11. – *Embeddings du corpus QQP*

9.5. Graphiques de tous les modèles selon leurs métriques d'évaluation

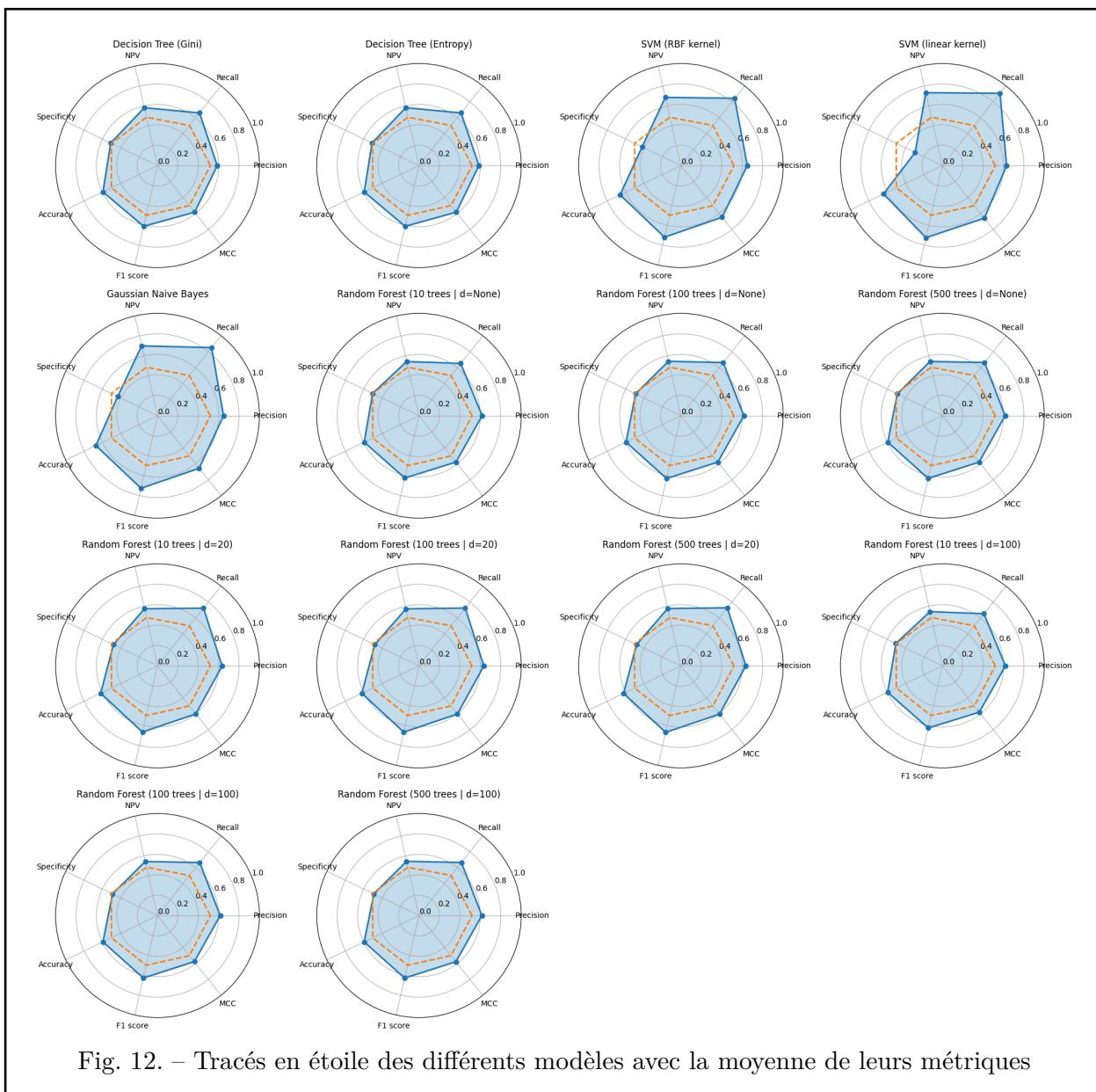


Fig. 12. – Tracés en étoile des différents modèles avec la moyenne de leurs métriques

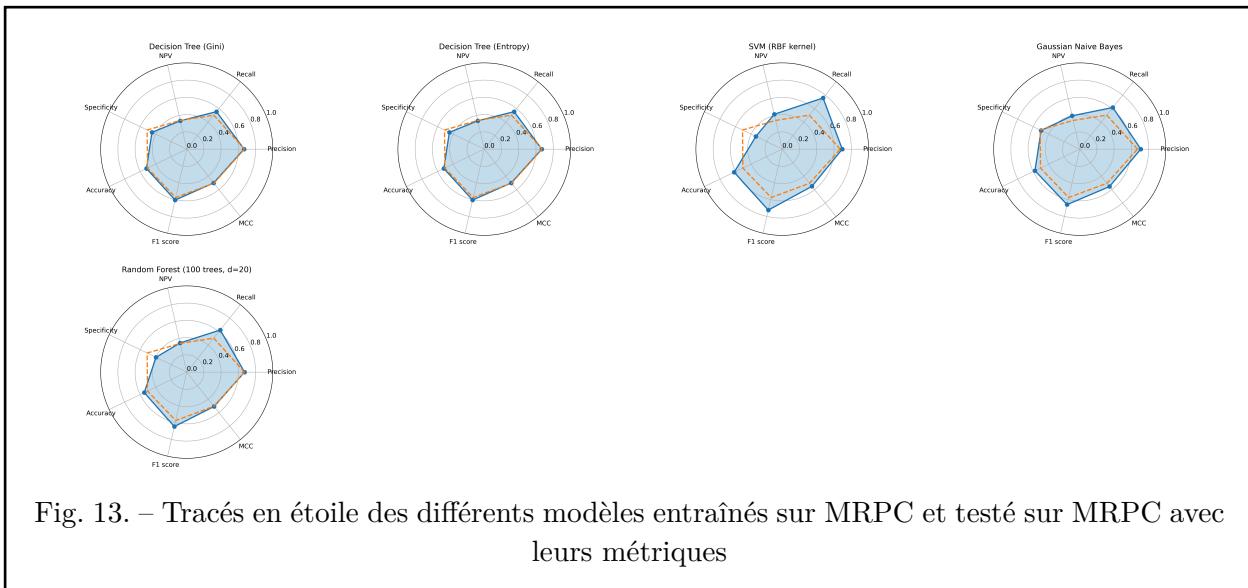


Fig. 13. – Tracés en étoile des différents modèles entraînés sur MRPC et testé sur MRPC avec leurs métriques

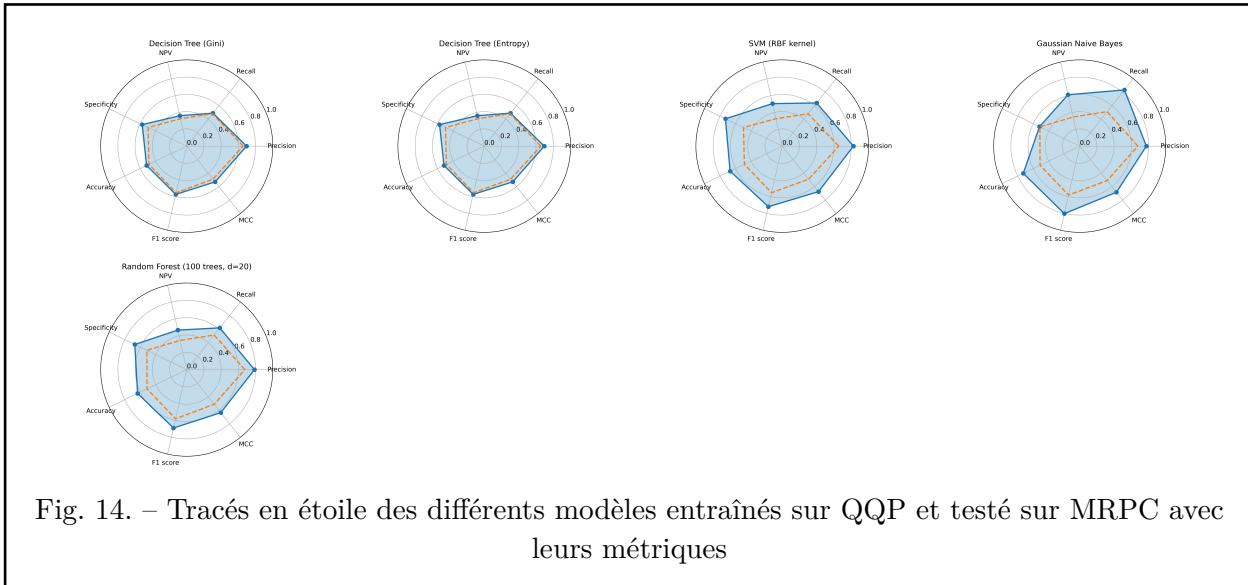


Fig. 14. – Tracés en étoile des différents modèles entraînés sur QQP et testé sur MRPC avec leurs métriques

Bibliographie

- [1] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, et S. R. Bowman, « GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: <http://arxiv.org/abs/1804.07461>
- [2] S. Iyer, N. Dandekar, et K. Csernai, « First Quora Dataset Release: Question Pairs ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>
- [3] W. B. Dolan et C. Brockett, « Automatically Constructing a Corpus of Sentential Paraphrases », dans *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. Consulté le: 28 avril 2024. [En ligne]. Disponible à: <https://aclanthology.org/I05-5002>
- [4] S. ES, « Exploratory Data Analysis for Natural Language Processing: A Complete Guide to Python Tools ». Consulté le: 29 avril 2024. [En ligne]. Disponible à: <https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools>
- [5] LaDepeche, « "On mange, les enfants" ou "on mange les enfants" les pièges du français à la loupe ». Consulté le: 29 avril 2024. [En ligne]. Disponible à: <https://www.ladepeche.fr/article/2018/02/25/2748992-mange-enfants-mange-enfants-pieges-francais-loupe.html>
- [6] « Gensim: Topic Modelling for Humans ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: <https://radimrehurek.com/gensim/models/doc2vec.html>
- [7] « Sklearn.Feature_extraction.Text.CountVectorizer ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
- [8] « Sklearn.Feature_extraction.Text.TfidfVectorizer ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- [9] « Universal Sentence Encoder | TensorFlow Hub ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder
- [10] E. Syriani, I. David, et G. Kumar, « Assessing the Ability of ChatGPT to Screen Articles for Systematic Reviews ». Consulté le: 27 avril 2024. [En ligne]. Disponible à: <http://arxiv.org/abs/2307.06464>
- [11] E. Collin, A. Dabilgou, et R. Gilles-Lesage, « TP2 - Apprentissage machine », dans *IFT3335 - Intelligence artificielle : introduction*, Montréal, QC, Canada: Université de Montréal, avr. 2024, p. 8-9.
- [12] « Linguistic Features · spaCy Usage Documentation ». Consulté le: 28 avril 2024. [En ligne]. Disponible à: <https://spacy.io/usage/linguistic-features#entity-types>