Note méthodologique : preuve de concept modèle RoBERTa

Projet 8 - Parcours Data Scientist

Antoine Arragon - Juin 2024

1. Dataset retenu
    a. Présentation Générale
    b. Objectifs
2. Concepts de l'algorithme RoBERTa
    a. L'architecture Transformer
    b. BERT
    c. RoBERTa
3. Modélisation
    a. Implémentation du modèle
    b. Métriques utilisées
4. Synthèse des résultats
5. Interprétabilité
6. Limites et améliorations

1. Dataset retenu :

1. **Présentation générale :**

Le dataset est composé de 1050 produits de consommation pour lesquels nous disposons du nom du produit, d'une description effectuée par le vendeur du produit et de la catégorie de ce produit.

On distingue 7 catégories :

- Home Furnishing ;
- Baby Care ;
- Watches ;
- Home Decor & Festive Needs ;
- Kitchen & Dining ;
- Beauty & Personal Care ;
- Computers.

Les produits sont équitablement répartis et nous en avons donc 150 par catégorie.

2. **Objectifs :**

Il s'agit ici d'un problème de classification multiple : le but est de prédire la catégorie des produits à partir de leur description. En réalité nous avons créé une nouvelle colonne qui comprend à la fois le nom du produit et sa description, et c'est à partir de ce texte que la prédiction de la catégorie devra être effectuée.

Il s'agit donc de données non structurées sur lesquelles nous devons appliquer certains traitements afin de rendre possible leur utilisation par des modèles de machine learning.

Les modèles utilisés disposent d'outils permettant un traitement spécifique et adapté comme nous le verrons par la suite.

2. Concepts de l'algorithme RoBERTa :

Le modèle RoBERTa - A Robustly Optimized BERT Pretraining Approach – a été introduit et présenté le 26/07/2019. Il consiste en une amélioration du modèle BERT (qui lui avait été introduit en octobre 2018), à la fois par un entraînement sur un ensemble de données encore plus conséquent, et par une modification de la structure du modèle et de certains hyperparamètres.

Ces deux modèles, BERT et RoBERTa, utilisent l'architecture Transformer (2017) et les mécanismes d'attention qui lui sont inhérents.

Pour présenter l'algorithme RoBERTa nous allons donc, dans un premier temps explorer l'architecture des transformers.

1. L'architecture Transformer :

Présentée en juin 2017 dans un article intitulé « Attention is all you need », l'architecture Transformer a été une avancée significative dans le domaine du Natural Language Processing (NLP). Comme indiqué dans la présentation de l'article, les modèles dominants auparavant étaient essentiellement basés sur des structures assez complexes de réseaux de neurones récurrents ou convolutionnels. Même si des méthodes telles que la « Long-Short-Term-Memory » (LSTM) ou « Gated Recurrent Unit » ont permis de surmonter les limitations des RNN en termes de « mémoire contextuelle », le Transformer, en ne se basant que sur les mécanismes d'attention, a obtenu de meilleurs résultats tout en réduisant la complexité de ce type de modèles. On a notamment noté des améliorations significatives en termes de dépendances de long terme dans la représentation contextuelle des mots analysés.
L'architecture globale d'un Transformer se présente ainsi :

- Un encodeur composé de 6 couches identiques, chacune d'elle divisée en 2 sous-couches constituées d'un mécanisme d'attention multi-tête puis d'un réseau entièrement connecté ;

- Un décodeur constitué également de 6 couches identiques, chacune d'elle divisée en 3 sous-couches. La constitution du décodeur est très semblable à celle de l'encodeur mais on y ajoute une sous-couche d'attention multi-tête où la position des mots est indiquée et où la prédiction du mot actuel ne peut se faire qu'avec les informations concernant les mots qui le précèdent.

L'article initial nous présente ce schéma de description :

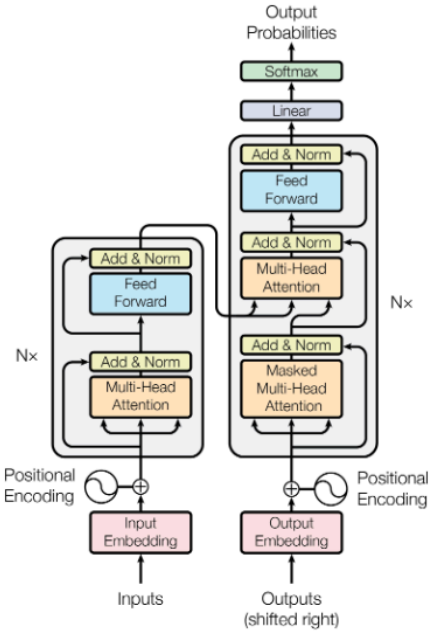


Figure 1: The Transformer - model architecture.

Principe général du mécanisme d'attention : mettre en valeur les éléments les plus pertinents de l'input en fonction de la tâche que l'on souhaite effectuer et permettre une mémoire à long terme.

1. self attention :

- Si l'on prend l'exemple d'un problème de NLP, l'objectif sera dans un premier de temps de convertir une séquence textuelle d'entrée en une représentation vectorielle (embedding) ;

- On calcule ensuite trois vecteurs : un vecteur de requête, un vecteur de clé et enfin un vecteur de valeur. Cela est fait en multipliant chaque embedding par des poids, q-k-v, initialisés aléatoirement puis mis à jour et optimisé durant l'entraînement du modèle ;

- Pour chaque mot, on calcule ensuite un score, correspondant au produit scalaire du vecteur de requête du mot en question, avec l'ensemble des vecteurs clés de tous les mots ;

- On normalise ensuite ce score via une fonction softmax → score compris entre 0 et 1 et la somme de l'ensemble vaut 1 ;

- On multiplie chaque vecteur de valeur par ce score normalisé ;

- Et enfin on additionne ces vecteur de valeur pondérés pour produire l'output de cette couche de self attention.

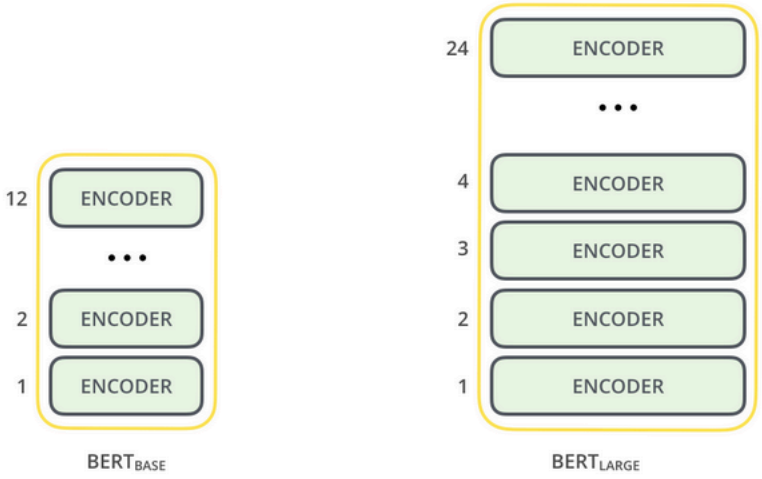
2. Multi-head attention :

L'architecture Transformer utilise également un concept d'attention multi-tête qui permet d'augmenter la robustesse du mécanisme de self-attention. En effet, ce mécanisme permet d'effectuer en parallèle plusieurs représentations vectorielles (8 dans l'architecture initiale) du sens et des similarités de chaque mot afin d'obtenir une précision sémantique et contextuelle plus approfondie.

2. BERT :

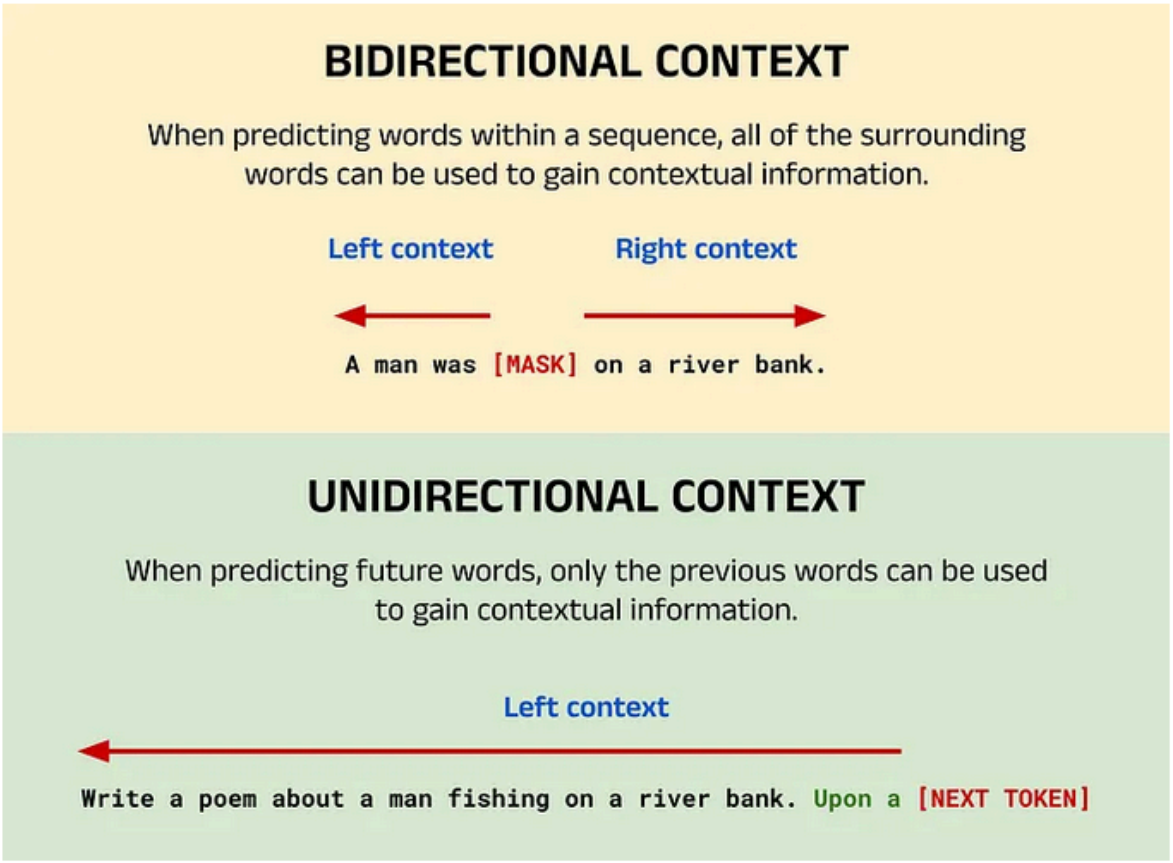
Le modèle BERT – Bidirectionnal Encoder Representations from Transformer – a été introduit par une équipe de recherche de Google AI en 2018.

Comme son nom l'indique, ce modèle se base sur l'architecture Transformer mais n'utilise que la partie encoder, se focalisant ainsi sur la compréhension du contexte entourant un mot ou une phrase.



(Source : https://jalammar.github.io/illustrated-bert/)

La bidirectionnalité a été un élément central de l'innovation et des gains de performance obtenus par BERT, en effet, cela permet au modèle de comprendre le sens d'un mot en « observant » simultanément les mots précédents et les mots suivants. De cette façon, la contextualisation et la qualité sémantique est plus élevée que dans les modèles précédents, unidirectionnels ou bidirectionnels mais de façon indépendante : de gauche à droite, puis de droite à gauche.



(Source :https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11)

Le modèle a été pré-entraîné sur un corpus massif de données, notamment l'intégralité du corpus Wikipédia anglais.

L'entraînement de BERT se fait de manière non supervisée et consiste en deux objectifs :

1. **la prédiction de mots masqués :**

L'article qui présente le modèle nous informe que 15% des tokens de chaque séquence d'entrée sont masqués lors de l'entraînement, l'objectif étant de prédire chaque token masqué.

2. **la prédiction de la phrase suivante :**

Ici il s'agit en fait pour le modèle de comprendre si deux phrases successives ont une relation logique ou non. A cette fin, durant l'entraînement, 50% des paires de phrases se succèdent « normalement » et on constitue 50% de paire de phrases de façon aléatoire.

BERT a obtenu des résultats significativement meilleurs que les modèles précédents sur des benchmarks tels que GLUE ou SquAD et ce pour divers type de tâches : classification de texte (analyse de sentiments par exemple), questions-réponses, prédiction de textes.

3. RoBERTa :

Comme dit plus haut, le modèle RoBERTa est en fait une optimisation du modèle BERT, l'architecture globale restant la même. Les modifications sont les suivantes :

- Augmentation du volume de données d'entraînement : le corpus d'entraînement de BERT représente 16GB de données, celui de RoBERTa environ 160 GB ;

- La façon d'encoder le texte est différente, les deux modèles implémentent du « Byte-Pair Encoding » mais BERT l'utilise au niveau des caractères pour une taille de vocabulaire de 30000 « subwords », tandis que RoBERTa utilise des octets comme base permettant la constitution d'un vocabulaire de 50000 « subwords ». Cela permet de ne pas avoir de tokens inconnus.

- Le masquage des mots se fait de façon dynamique et non pas statique comme dans BERT, cela permet d'améliorer encore la capacité de compréhension du contexte.

- Augmentation de la taille des « batch » et réduction dans le même temps du nombre d'étapes d'entraînement.

    - BERT : batch_size = 256 / steps = 1M

    - RoBERTa : batch_size = 8k / steps = 31k.

  Les auteurs justifient ce choix par la littérature précédente qui montrait que l'augmentation du batch_size conduisait à de meilleures performances et aussi par le fait que cela permet une meilleur parallélisation des calculs et une optimisation des ressources computationnelles.

- Suppression de l'objectif de prédiction de la phrase suivante. Les auteurs ont montré qu'ils obtenaient de meilleurs résultats (ou au moins des résultats équivalents) sur différentes tâches en enlevant cet objectif lors de l'entraînement du modèle. Par ailleurs, dans le modèle RoBERTa les séquences se suivent de documents en documents jusqu'à obtenir le nombre maximum de tokens, établis à 512.

Les auteurs montrent que ces modifications ont permis de surpasser BERT, mais aussi d'autres modèles comme le XLNet sur plusieurs problématiques de NLP.

3. Modélisation :

Bref rappel sur les données :

- textes : nom + description du produit ;
- variable cible : catégories des produits – 7 catégories

a) Implémentation du modèle :

Nous avons utilisé la librairie transformers, mise à disposition par HuggingFace, qui permet l'importation de différents modèles pré-entraînés ainsi que d'outils spécifiques adaptés à chaque problématique.

Les transformers et les modèles comme BERT et RoBERTa étant particulièrement longs à entraîner, nous avons utilisé du Transfer Learning et importé un modèle pré-entraîné, nous avons modifié la couche finale afin de l'adapter à notre classification multiple à 7 catégories.

1. Nous avons dans un premier temps encodé les catégories via un LabelEncoder puis séparé les données en jeux d'entraînement et de test ;

2. Ensuite nous avons effectué une tokenisation afin de créer les vecteurs d'embedding et d'avoir des données assimilables par le modèle. L'utilisation de troncature et de padding permet d'assurer des séquences d'entrée de longueurs similaires ;

3. Nous avons donc chargé le modèle « RobertaModel » pré-entraîné « roberta-base » ainsi que l'objet permettant la tokenisation appropriée : « RobertaTokenizer ».
   Nous avons ensuite spécifié le nombre de classes à prédire ;

4. Nous nous sommes inspirés d'un notebook d'exemple pour des problématiques de classification mis à disposition par Huggingface (https://colab.research.google.com/github/DhavalTaunk08/NLP_scripts/blob/master/sentiment_analysis_using_roberta.ipynb). On retrouve les différentes couches d'attention ici :

```
(encoder): RobertaEncoder(
  (layer): ModuleList(
    (0-11): 12 x RobertaLayer(
      (attention): RobertaAttention(
        (self): RobertaSelfAttention(
          (query): Linear(in_features=768, out_features=768, bias=True)
          (key): Linear(in_features=768, out_features=768, bias=True)
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
```

5. Pour des raisons techniques (temps de calcul et mémoire de l'ordinateur) le nombre d'epochs et le batchsize sont assez faibles, respectivement : 2 et 3, à la fois pour

l'entraînement et la validation. Le learning rate a été établi à 0.00001.

b) Les métriques utilisées :

- l'accuracy : fréquence des observations bien classées ;
- la précision : ratio du nombre de vrais positifs sur l'ensemble des prédictions positives ;
- le rappel : ratio du nombre de vrais positifs sur l'ensemble des éléments positifs ;
- le f1 score : moyenne harmonique de la précision et du rappel ;
- la matrice de confusion qui permet de voir en détail les prédictions effectuées et les catégories réelles ;

Nous avons retenu ces métriques à la fois sur le training set et sur le test set de manière à appréhender la capacité de généralisation du modèle.

4. Synthèse des résultats :

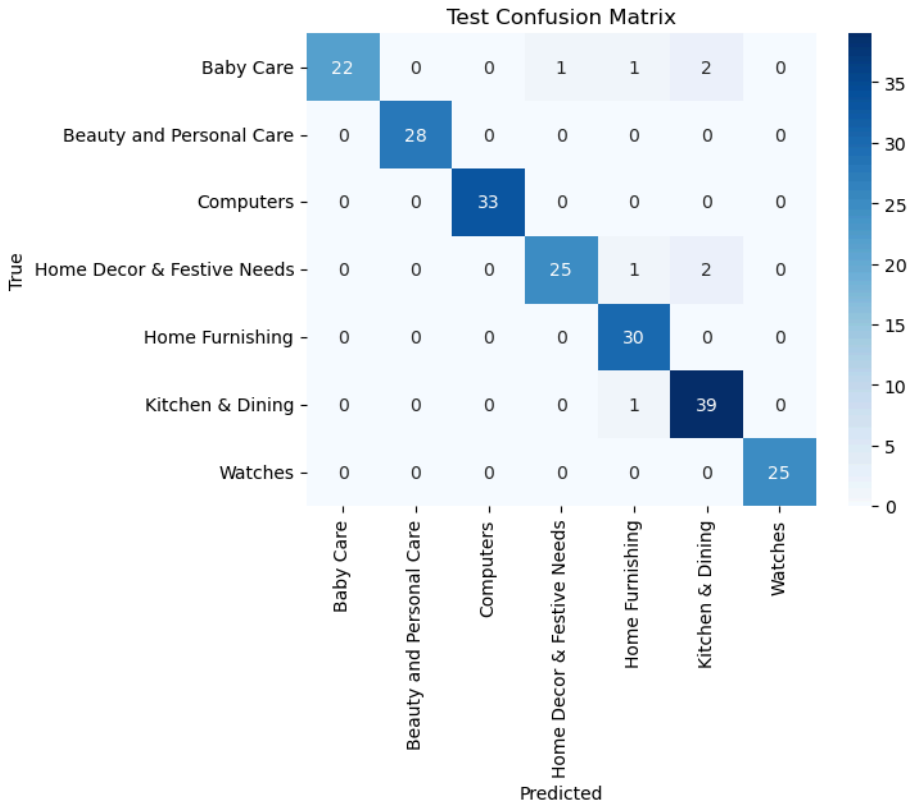
Nous avons voulu comparer les performances du modèle RoBERTa avec le modèle BERT initial, que nous avions utilisé, dans le cadre d'un projet précédent, pour de l'extraction de features de manière à se prononcer sur la faisabilité d'une classification pour le même jeu de données.

Nous avons implémenté ce dernier de la même manière, en utilisant les outils fournis par Huggingface : « BertModel » et « BertTokenizer » pour le modèle pré-entraîné « bert_base_uncased ».

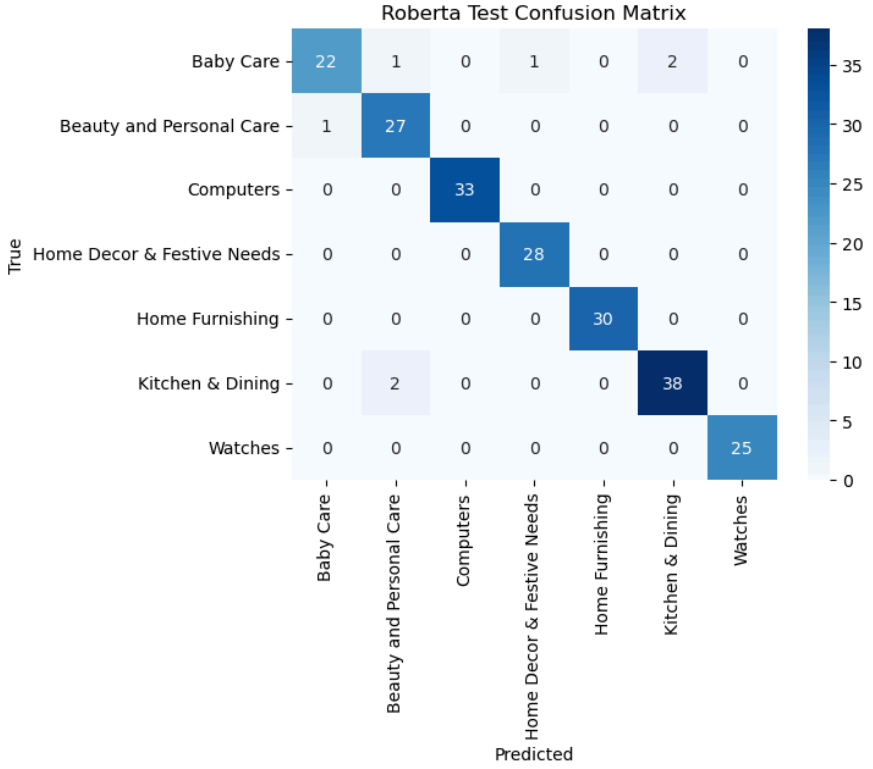
Nous avons obtenu des résultats sensiblement similaires entre les deux modèles pour ce problème de classification :

	Model	Train Accuracy	Train Precision	Train Recall	Train F1-Score	Eval Accuracy	Eval Precision	Eval Recall	Eval F1-Score
0	BERT	0.9512	0.9526	0.9512	0.9509	0.9619	0.9642	0.9619	0.9615
1	Roberta	0.9821	0.9826	0.9821	0.9821	0.9667	0.9672	0.9667	0.9663

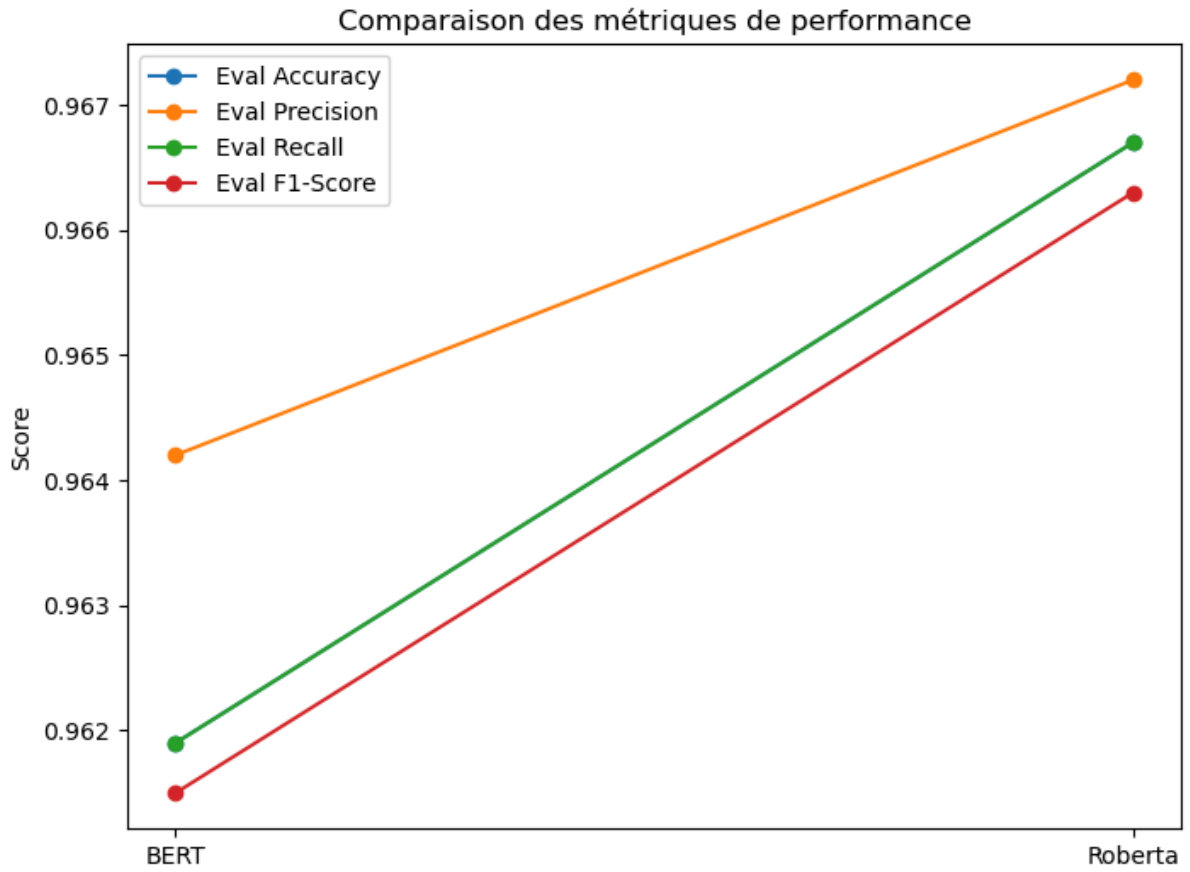
Pour le modèle BERT, pour la meilleure epoch et sur le jeu de validation, nous obtenons la matrice de confusion suivante :



Pour le modèle RoBERTa, pour la meilleure epoch et sur le jeu de validation, nous obtenons la matrice de confusion suivante :

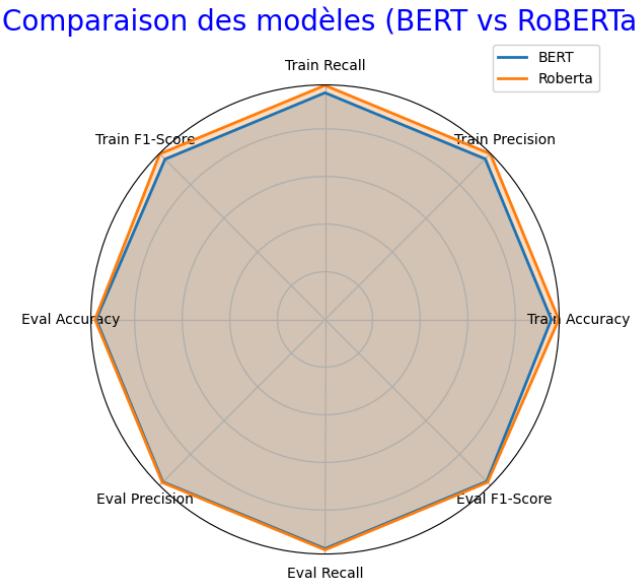


Enfin on voit ici graphiquement que sur le test set on a des résultats similaires légèrement meilleurs pour RoBERTa même s'ils sont sensiblement équivalents avec BERT :



(L'accuracy et le rappel se confondent).

On présente ici une autre manière de le voir, on voit par ailleurs que les résultats sur le jeu d'entraînement sont légèrement meilleurs, là aussi, pour RoBERTa :



5. Interprétabilité :

Pour ce type de modèle, il est plus difficile d'avoir une vue sur les importances locales et globales des features utilisées que pour des modèles plus « classiques » tels que XGBoost ou RandomForest.

Il existe cependant des moyens et notamment grâce à la librairie Captum. Des méthodes comme « Integrated Gradients » permettent d'identifier ce qu'on appelle les « features attribution » et qui correspondent à l'impact de chaque feature sur une prédiction. En l'occurrence il s'agirait d'avoir accès à l'importance de chaque mot dans le choix de la catégorie de produit.

Le site de Captum (https://captum.ai/tutorials/Bert_SQUAD_Interpret) fourni un exemple dans le cadre d'une tâche de Question-Réponse avec le modèle BERT :

Visualizations For Start Position

Legend: ■ Negative □ Neutral ■ Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
13	13 (0.72)	13	1.43	[CLS] what is important to us ? [SEP] it is important to us to include , em ##power and support humans of all kinds . [SEP]

Visualizations For End Position

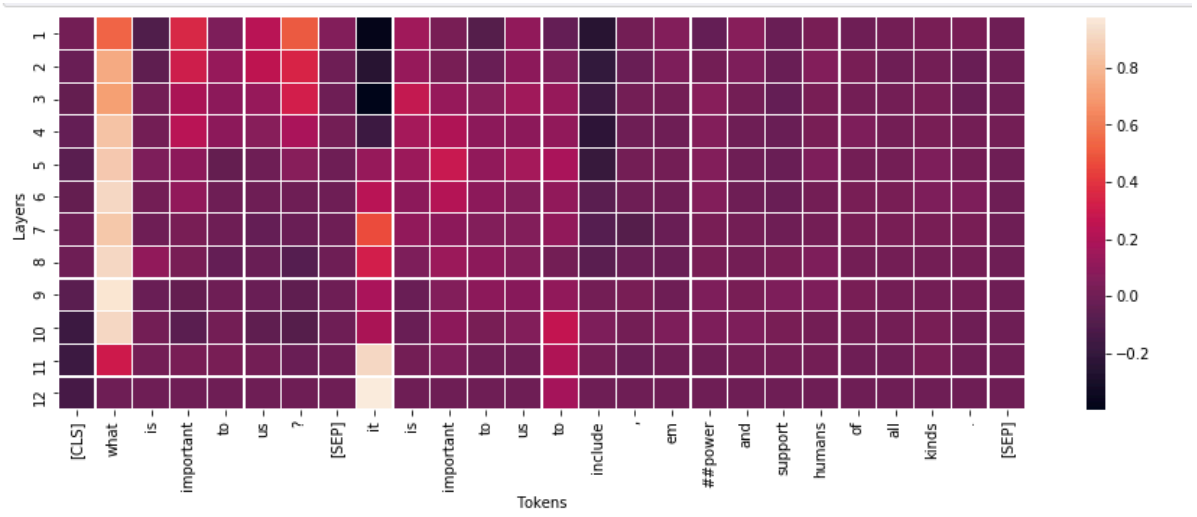
Legend: ■ Negative □ Neutral ■ Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
23	23 (0.73)	23	2.00	[CLS] what is important to us ? [SEP] it is important to us to include , em ##power and support humans of all kinds . [SEP]

Il s'agit donc de mettre en évidence les mots ayant été les plus importants dans la décision finale. Cela montre également que la « partie question » a été plus déterminante pour définir la position initiale alors que la « partie texte-réponse » a été plus importante pour définir la position finale.

Une autre possibilité offerte par cette librairie est de visualiser l'importance de certaines features (certains mots ou tokens) à mesure qu'on « avance » dans les différentes couches du modèle. Certains mots peuvent gagner en importance ou au contraire voir leur impact diminuer.

Toujours avec le même exemple :



La librairie Captum permet également de visualiser des matrices d'attention pour une couche spécifique. Cela peut permettre de visualiser l'importance d'un mot pour d'autres mots dans leur représentation contextuelle.

Dans ce travail nous n'avons malheureusement pas réussi à implémenter ces différents éléments d'interprétabilité. C'est un travail qui demanderait plus de temps et de comprendre plus finement la complexité de ces outils.

6. Les limites et améliorations possibles :

Une des premières limites de ce travail est évidemment liée au point précédent, à savoir le manque d'interprétabilité des prédictions de notre modèle.

L'utilisation de ce type de modèle est assez coûteuse en ressources et en temps, cela nous a pénalisé à la fois pour affiner et essayer différents paramètres : nous n'avons pas pu augmenter le batch_size ou le nombre d'epochs par exemple, ni faire de multiples essais et itérations en changeant le taux d'apprentissage ou d'autres éléments. Ce travail mériterait plus de temps et un approfondissement de nos connaissances concernant les réseaux de neurones, l'architecture transformer et les modèles basés sur BERT.

Il pourrait être intéressant d'utiliser des modèles connus pour être plus légers tels que DistilBert, de creuser un peu les différentes possibilités offertes par HuggingFace dans les ajustements des modèles pré-entraînés mis à disposition.

Dans un notebook d'exemple pour de la classification de texte, HuggingFace indique qu'il est possible d'implémenter une recherche des meilleurs hyperparemètres via leur objet « trainer ». Cela peut notamment permettre de trouver le taux d'apprentissage, le nombre d'epochs et le batchsize optimal.

L'utilisation d'autres types de Large Language Model pourrait être envisagée, tels que Llama 2 ou les modèles proposés par Mistral ou OpenAI. Nous avons ici fait le choix de présenter un modèle se situant dans la continuité directe de celui que nous avions rapidement commencé à utiliser dans un projet précédent.

Bibliographie / webographie :

1. Articles initiaux :

- “Attention Is All You Need” - 2017 : https://arxiv.org/abs/1706.03762
- “RoBERTa: A Robustly Optimized BERT Pretraining Approach” - 2019 : https://arxiv.org/abs/1907.11692
- “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” - 2018 : https://arxiv.org/abs/1810.04805

2. Ressources HuggingFace :

- https://huggingface.co/docs/transformers/model_doc/roberta
- https://huggingface.co/tasks/text-classification
- https://colab.research.google.com/github/DhavalTaunk08/NLP_scripts/blob/master/sentiment_analysis_using_roberta.ipynb
- https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/text_classification.ipynb#scrollTo=Bliy8zgjlrJY
- https://huggingface.co/blog/bert-101
- https://huggingface.co/docs/transformers/v4.41.2/en/model_doc/bert#transformers.BertModel
- https://huggingface.co/learn/nlp-course/chapter0/1

3. Articles de blogs :

- https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/
- https://machinelearningmastery.com/natural-language-processing/
- https://machinelearningmastery.com/the-transformer-model/
- https://machinelearningmastery.com/what-is-attention/
- https://machinelearningmastery.com/the-attention-mechanism-from-scratch/
- https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/
- https://machinelearningmastery.com/a-brief-introduction-to-bert/
- https://towardsdatascience.com/contextual-transformer-embeddings-using-self-attention-explained-with-diagrams-and-python-code-d7a9f0f4d94e
- https://towardsdatascience.com/a-complete-guide-to-bert-with-code-9f87602e4a11
- https://jalammar.github.io/illustrated-bert/
- http://jalammar.github.io/illustrated-transformer/
- https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente