**Note méthodologique : preuve de concept**

Tables des matières

[Dataset retenu](#_k81ww76gjwr3)

[Concepts de deBERTa](#_3h34yuuuawzi)

[Fonctionnement de la base BERT](#_8cxn7w8bge91)

[Exemple](#_9wduils7k2fy)

[Fonctionnement de l’attention de deBERTa](#_5m77wxs9ggfx)

[Modélisation](#_7d8agjffbvmn)

[Méthodologie](#_fja0dvfjnyay)

[Synthèse des résultats](#_ptxgh8jrsxxz)

[Bert T-SNE et K-means](#_ecfck098h773)

[deBERTa T-SNE et K-means](#_saj656wuvoxu)

[Random Forest](#_adrk0g7c6l1n)

[BERT](#_myfxdcmjrky)

[DeBERTa](#_j296lk8wmlok)

[Conclusion](#_biyu0eip81kb)

[Analyse de la feature importance globale et locale](#_6qr91mqzqdnk)

[Limites et améliorations possibles](#_n779q1ai5i58)

## **Dataset retenu**

Le dataset utilisé est celui d’un projet précédemment utilisé. Il avait pour objectif de permettre une classification automatique de divers biens de consommation pour une marketplace en ligne. A la fois en utilisant des éléments textuels mais également en utilisant des données visuelles au travers des différentes images produites.  
Dans le cadre de cette note méthodologique, seule la partie traitement textuelle sera utilisée.

Pour effectuer la classification lors de ce projet nous avions utilisé diverses méthodes de traitement de texte.  
Des méthodes de types Bag of Word[[1]](#footnote-0):

* **TF-IDF** : **Term Frequency :** Nombre d’apparitions du mot dans le corpus.**Inverse Document Frequency :** Réduction des mots communs dans le corpus.
* **Bag of Words** : Modèle focalisé sur la fréquence d’apparitions des mots dans le corpus.Chaque texte devient un vecteur où chaque position correspond à un mot.

Des méthodes embedding[[2]](#footnote-1):

* **Word2vec:** Modèle de représentation vectorielles des mots en tenant compte de leurs contextes.
* **USE:** Modèle de représentation vectorielle de phrases entières et pas uniquement de mots.
* **BERT:** Modèle entraîné sur un grand nombre de données et au fonctionnement bidirectionnel.

## **Concepts de deBERTa**

Lors du projet précédent,après une rapide analyse de faisabilité nous avions retenu BERT comme modèle le plus pertinent pour préparer nos divers éléments textuels à la classifications. Nous nous étions basées sur un ARI obtenu via Kmeans et une représentation visuelle d’un T-SNE.

Durant notre veille, un algorithme récent à attiré notre attention, deBERTa, venant apporter un certains nombres d’améliorations à BERT. Nous avons donc décidé de mettre celui-ci à l’épreuve sur le même dataset que nous avions utilisé afin de voir si, dans notre cas d'usage, ce nouvel algorithme apportait des améliorations aux précédents résultats.

### Fonctionnement de la base BERT

Entraînement sur un principe de mot masqué. Ainsi le modèle utilise les mots placé avant et après le blanc pour definir le terme masqué. De la même façon qu’un humain utilise le contexte pour déterminer le sens de certains mots.

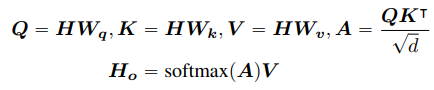
Le modèle utilise trois notions principalement :

**Word embedding :** transformation de chaque mot sous forme de vecteur.

**Positional Encoding :** ajout des informations concernant la position du mot au vecteur de mot.

**Attention[[3]](#footnote-2) :** Le score d’attention est combiné au vecteur du mot pour représenter la relation.

Voici la formule sur laquelle s’appuie BERT comme présenté dans la [documentation de deBERta](https://arxiv.org/pdf/2006.03654) :



**W :** les matrices de projection permettant de passer de H a Q,K et V.

**Q** : Représente le mot que le modèle traite.

**K** : Représente les autres mots de la phrase qui vont être comparés à Q à la recherche des informations les plus pertinentes.

**V** : Contient les diverses informations dans la phrase. Après application de l’attention V sera modifié selon le contexte et les mots fortement liés à Q pour créer une nouvelle représentation.

**A** : représente l’attention, calculée par le produit scalaire entre Q et V.

**Softmax :** permet de transformer les valeurs en probabilité (max =1).Multiplié par V on obtient les informations de sortie Ho.

#### Exemple

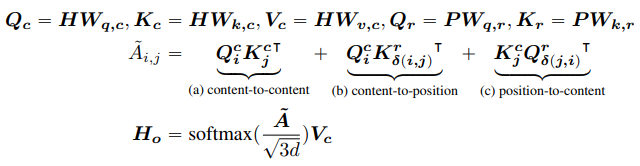
*Le chien aboie sur le voisin.*

**Q** = chien, on cherche donc les mots liés à celui ci

**K** = le, chien, aboie,sur,le,voisin, on compare chaque mot avec Q.

**V** = le,chien, aboie,sur,le,voisin, chaque mot transformé sans pondération par l’attention. Puis la matrice A sera appliquée aux valeurs de V. Par exemple si chien et aboie sont fortement liés alors le vecteur de chien intégrera la notion d’aboyer.

### Fonctionnement de l’attention de deBERTa



On retrouve Q, K et V comme dans BERT mais également Qr et Kr.

**Qr** = position query

**Kr** = position clés

**Content to content :** mesure la similarité entre les contenus des tokens via Q et K.

**Content to position :** mesure la relation spatiale entre Q et les autres tokens (K)

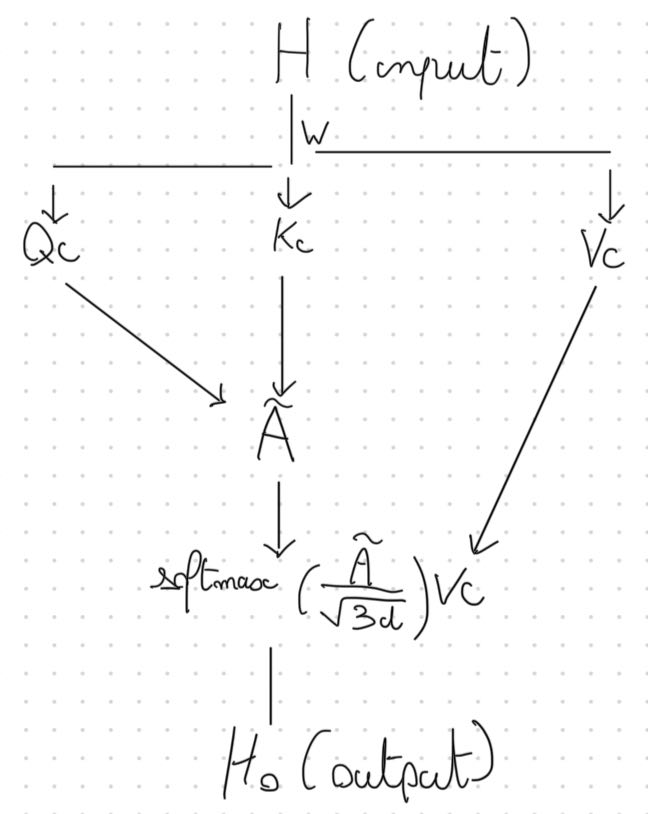
**Position to content :** influence l’importance des tokens K selon leur position à Q et leur contenu.

**A** : le calcul se fait maintenant avec trois éléments : le contenu basé sur le sens des tokens, le deuxième va influer l’attention selon la position des mots dans la phrase et la troisième va modifier l’importance des mots vis-à -vis de leur distance les un par rapport aux autres.

**SQRT(3d) :** est là pour normaliser le résultat en prenant en compte les trois dimensions de l’attention.

**Vc :** contient le contenu des mots sans informations de position.

La principale différence se fait sur le traitement de V. En effet dans BERT V contient déjà des informations quant à la position du mot. Dans deBERTa, Vc contient uniquement les informations du contenu des mots, les positions sont gérées par Qr et Kr. C’est uniquement aux moments du calcul de l’attention que les informations de position sont utilisées.



*Graphique du fonctionnement de deBERTa basé sur cet* [*article*](https://towardsdatascience.com/large-language-models-deberta-decoding-enhanced-bert-with-disentangled-attention-90016668db4b/)

## Modélisation

Pour évaluer les performances des deux modèles nous avons choisi en plus de la méthode précédente (T-SNE) d’ajouter une randomforest simple avec mesure d’une métrique pour comparer les performances obtenus avec les différents embeddings.

### Méthodologie

Nous avons dans un premier temps sélectionné les éléments textuels importants permettant de qualifier les produits (nom et description), les catégories sont égalements récupérées et clean simplement (retrait des crochets et guillemets) et isoler du reste du data frame pour éviter que les modèles de traitements de textes intègre ces informations dans les embeddings.

La distribution des catégories à également été vérifiée, les septs catégories sont représentées de manières égales dans le dataframe. Cet élément nous aidera à choisir notre métrique.

Les phrases avant traitements ont été tokenizer via le tokenizer lié au modèle.

La maxlength[[4]](#footnote-3) et le batch\_size[[5]](#footnote-4) ont été définis pour une valeur de départ de 64 et 10.

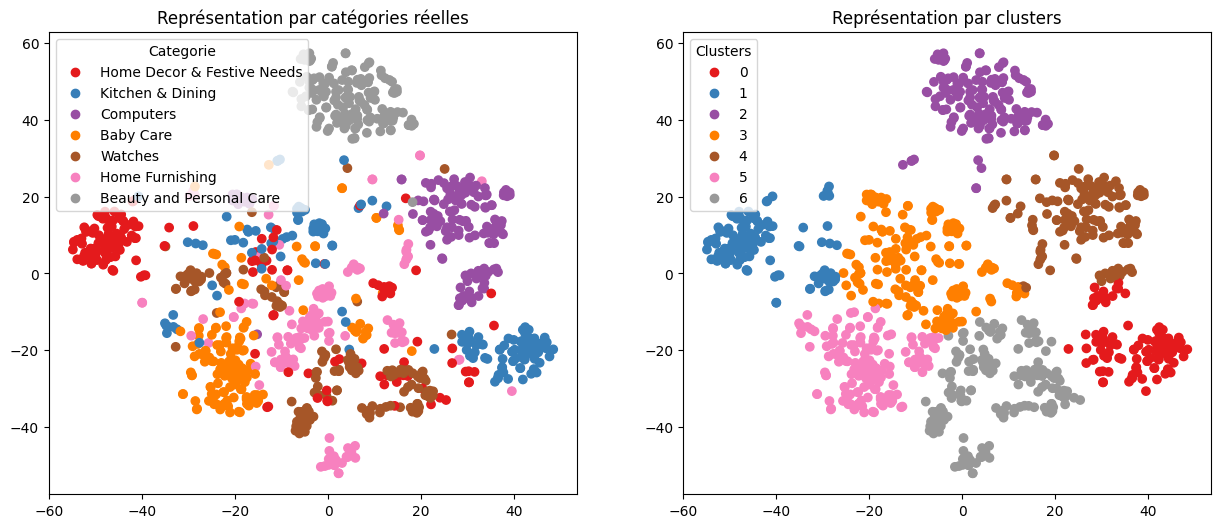
Une première analyse rapide et basique à été faites aux moyens d’un T-SNE et d’un calcul du score ARI[[6]](#footnote-5) pour un K-means[[7]](#footnote-6) simple à 7 voisins pour simuler une classification simple.

La max length à été défini en testant plusieurs valeurs différentes et en observant les résultats données par le T-SNE et l’ARI.

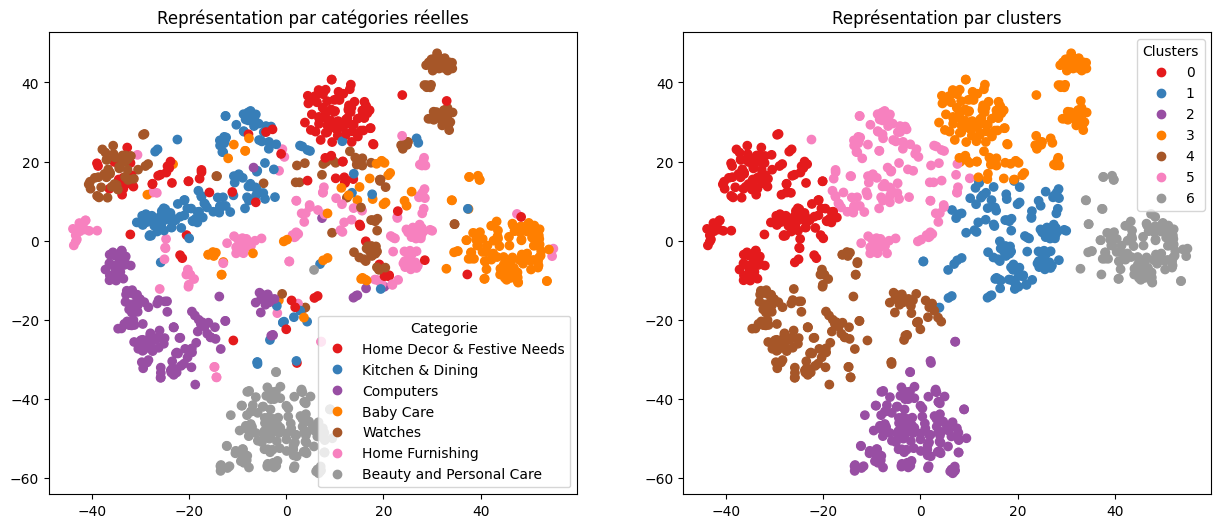
Pour pousser l’analyse plus loin, pour chaque modèles nous avons séparés les embedding obtenues en train et test dans le but d’utiliser une randomforest[[8]](#footnote-7) pour la classification. L’évaluation des modèles se fera via l’accuracy[[9]](#footnote-8).

## Synthèse des résultats

### Bert T-SNE et K-means

**ari : 0.4406

### deBERTa T-SNE et K-means



ari : 0.414

La première comparaison simple tend à nous montrer que la différence entre les deux modèles est faible avec un léger avantage pour l’ancien, BERT.

### Random Forest

Les résultats de la random forest avec les hyperparametres suivants : (n\_estimators=30, max\_depth=3, min\_samples\_split=5,min\_samples\_leaf=3, max\_features="log2") quant à eux place deBERTa devant BERT avec les résultats suivants pour les modèles :

#### BERT

Train accuracy: 0.85

Test accuracy: 0.77

#### DeBERTa

Train accuracy: 0.83

Test accuracy: 0.80

### Conclusion

Avec également un écart moins important en train et test pour le nouveau modèle.  
Ainsi, un modèle un peu plus évolué que le K-means tire un meilleur partie des embeddings de deBERTa que de BERT.

## Analyse de la feature importance globale et locale

L’embedding ne nous donnant pas directement accès aux différents features nous devrons changer notre manière d’analyser les résultats.

Les deux graphiques nous permettent également de visualiser s' il existe des différences notables aux niveaux globales ou locales entre les modèles.

Les scores de feature importance de la partie globale sont répartis sur l’ensemble des dimensions avec des pics à certains endroits. Nous pouvons en déduire qu’aucune plage de dimension de l’embedding ne porte d’informations fortement discriminantes pour nos deux modèles.

Pour le graphique local par contre on distingue des pics plus importants et plus nombreux suggérant que selon la catégorie certains embedding sont beaucoup plus important tandis que d’autres n’apportent pas de réelle information utile.

Nous pouvons également voir que les pics d’importances ne sont pas placés sur les mêmes dimensions mettant en évidence que les deux modèles n’ont pas encoder les données de façon identique.

Cette interprétation à des limites, en effet en nous basant uniquement sur les embeddings il ne nous est pas possible de savoir quels sont réellement les tokens et les notions ayant influencé les différentes décisions et si certains termes, indifféremment de la catégorie du texte ont tendances à être rattaché à certaines catégories bien précises.

## Limites et améliorations possibles

Au vu de l’objectif et des tâches pouvant être traité par deBERTa le dataset utilisé ici n’est pas le plus pertinent pour montrer l’ensemble des capacités du modèle. En effet, le modèle est censé exceller dans la compréhension de phrase. Hors ici nous sommes face à des descriptifs produits ou les vendeurs vont entrer des informations techniques ou de mise en avant de spécificité sans réellement de structure.

Le modèle étant plus complexe et plus lent que BERT il serait intéressant de mesurer son temps d'exécution sur un dataset plus large que celui utilisé afin de voir si le temps fonctionnement reste contenu, ou que l’écart de performance justifie son usage face à BERT.

Il existe plusieurs version depuis deBERTa-base, une comparaison entre certaines de celles ci pourrait être pertinente  
v3-base : pour voir si des améliorations ont été apportées, compréhension, performance etc.

large : est ce que le fait d’avoir un jeu de donnée plus important lors de l'entraînement à bénéficier au modèle et dans quel cas cela est le plus intéressant.

small : est ce que la performance n’est pas trop détériorée en comparaison du modèle de base et le gain de temps est il notable et en mesure de justifier l’usage de cette méthode pour certaines tâches.

1. Sac de mots [↑](#footnote-ref-0)
2. Vectorisation des mots : transformation des données textuelles en données numériques. [↑](#footnote-ref-1)
3. Prise en compte de la relation entre les mots. [↑](#footnote-ref-2)
4. Longueur maximum de token dans une phrase après tokenization.(Si la phrase est trop longue elle sera tronquée si elle est trop courte remplie avec des tokens de padding). [↑](#footnote-ref-3)
5. Le nombre maximum de phrases traité en simultanée. [↑](#footnote-ref-4)
6. Adjusted Rand Index : calcul la similarité entre des éléments du clustering à la classification de réel. [↑](#footnote-ref-5)
7. Algorithme permettant de regrouper des points en clusters basés sur la proximité des points en cherchant à limiter la variance au sein des clusters. [↑](#footnote-ref-6)
8. Algorithme basée sur des arbres de décisions qui vont fournir un résultat par agrégation des résultats. [↑](#footnote-ref-7)
9. La mesure du nombre de prédictions correctes par rapport au nombre de prédictions totale. [↑](#footnote-ref-8)