Projet n°8 – formation data-scientist

Note méthodologique : preuve de concept de l'implémentation de l'algorithme CamemBERT

Jean EMIDIO - Janvier 2026

Sommaire

1) Choix du sujet

La mission proposée dans ce projet consiste à faire un état de l'art sur une technique de modélisation de données de type texte ou image, l'analyser, la tester et la comparer à une approche plus classique.

Le choix de mon sujet d'étude porte sur les méthodes récentes de NLP (Natural Language Processing) pour le traitement de données textes en langue française, avec comme cas d'emploi l'objectif de trouver le meilleur algorithme permettant de classifier des émotions à partir de textes français. Il s'agit ici d'un problème de classification multiple avec des données non structurées sur lesquelles nous devrons appliquer certains traitements afin de rendre possible leur utilisation par des modèles de Machine Learning.

2) Dataset retenu

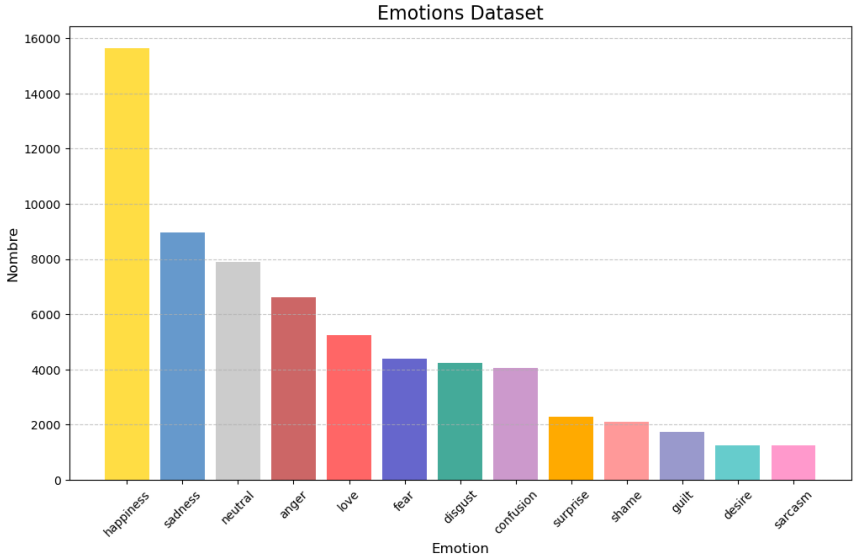
Présentation

C'est un dataset trouvé sur huggingface de 2 colonnes : Sentence et Label, le tout en anglais

- **Taille** : ~ 65 656 phrases en anglais + émotions associées
- **Classes** : 13 types d'émotions (colonne Label)
- **Type de texte** : Texte court, typique des usages réels (réseaux sociaux, sms …)

	Sentence	Label
0	Unfortunately later died from eating tainted m...	happiness
1	Last time I saw was loooong ago. Basically bef...	neutral
2	You mean by number of military personnel? Beca...	neutral
3	Need to go middle of the road no NAME is going...	sadness
4	feel melty miserable enough imagine must	sadness

Contenu du dataset

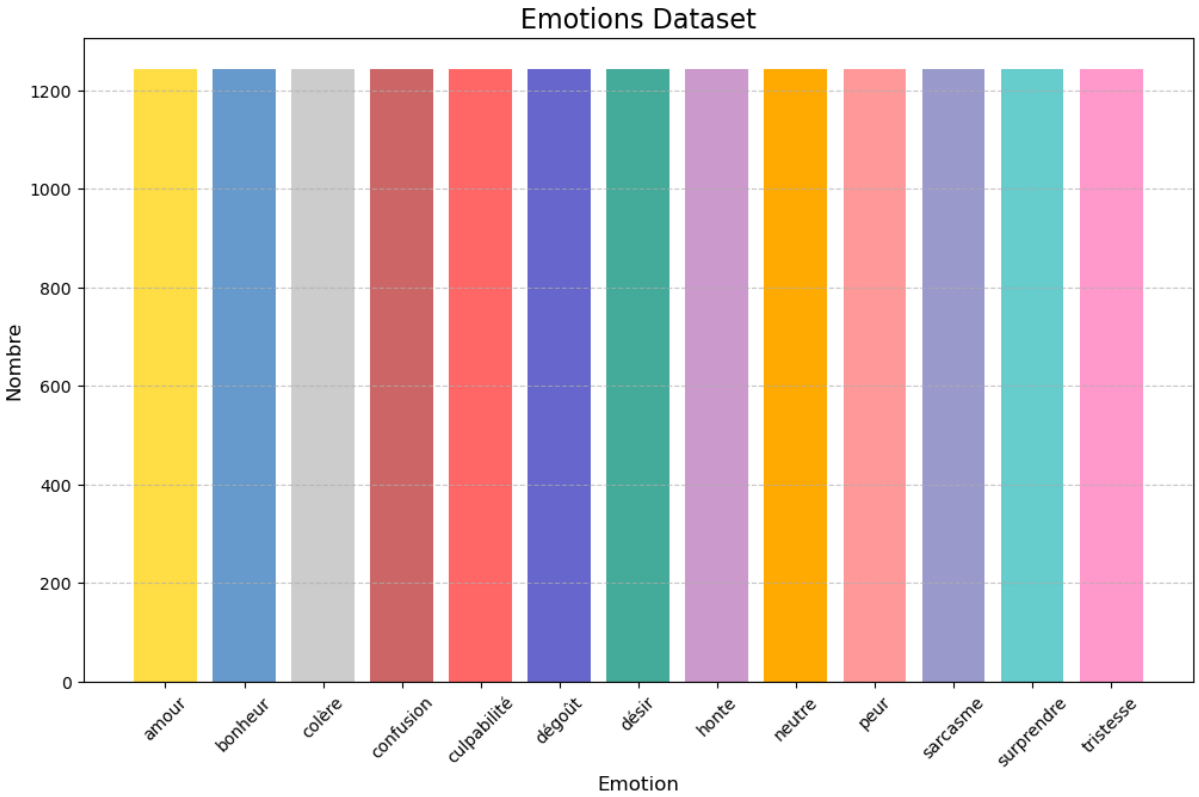


Transformations du dataset

- Traduire les phrases en français avec la librairie GoogleTranslator de deep_translator
- Equilibrer les classes pour un apprentissage stable et reproductible
- Encoder la colonne Label avec la librairie LabelEncoder de scikit-learn
- Ajouter une colonne Emoticone (pas nécessaire mais plus sympa pour une application future)

Résultat

	Sentence_en	Label_en	Sentence_fr	Label_fr	label_encoded	Emoticone
8455	man i remember playing street hockey at desert breeze growing up good times	happiness	mec je me souviens avoir joué au hockey de rue à desert breeze en grandissant de bons moments	bonheur	1	😄
59982	i think a few guys on the team have said just listens to country music lol	happiness	je pense que quelques gars de léquipe ont dit quils écoutaient simplement de la musique country mdr	bonheur	1	😄
13790	but it s accurate the other link states he was living in one of the poorest provinces in china dude was just an idiot	anger	mais cest exact lautre lien indique quil vivait dans lune des provinces les plus pauvres de chine ce mec était juste un idiot	colère	2	😡
63850	im ecstatic the app delivers tickets so fast	happiness	je suis ravi que lapplication livre les billets si rapidement	bonheur	1	😄
26865	even though you can me im slow clapping to this	neutral	même si vous le pouvez je mets du temps à applaudir	neutre	8	😐
3358	you have no right to be there waiting to be nudged	anger	vous navez pas le droit dêtre là à attendre quon vous pousse	colère	2	😡
51622	harden heart toughen skin order truly protect myelf feel utterly devastate	sadness	durcir le cœur durcir lordre de la peau me protéger vraiment me sentir complètement dévasté	tristesse	12	😢
36716	i get the joke and still think this sucks	happiness	je comprends la blague et je pense toujours que cest nul	bonheur	1	😄
53538	matchroom squad dismiss name as not existing anymore the sky media influence is scary	fear	léquipe matchroom rejette name comme nexistant plus linfluence des médias célestes est effrayante	peur	9	😨
4619	literally awesome that someone else is doing the same thing funny thing is i go for two a year also heck yeah fellow dodger friend	love	cest vraiment génial que quelquun dautre fasse la même chose ce qui est drôle cest que jy vais aussi deux fois par an bon sang ouais mon ami dodger	amour	0	😍



3) Algorithmes étudiés

Etude bibliographique :

Après une recherche bibliographique sur ArXiv j'ai retenu pour cette étude comparative les 4 modèles NLP suivants :

- **BERT :** (Bidirectional Encoder Representations from Transformers) a été présenté pour la première fois en octobre 2018 par des chercheurs de Google AI.

  Article : Pre-training of Deep Bidirectional Transformers for Language Understanding

  Lien arXiv : https://arxiv.org/abs/1810.04805

- **ALBERT :** (A Lite BERT) a été introduit par Google Research (avec Toyota Technological Institute at Chicago) dans un article publié sur arXiv en septembre 2019 et présenté comme conférence à ICLR 2020.

  Article : ALBERT : A Lite BERT for Self-supervised Learning of Language Representations

  Lien arXiv : https://arxiv.org/abs/1909.11942

- **RoBERTa :** RoBERTa (Robustly Optimized BERT Pretraining Approach) est une amélioration de BERT publiée par Facebook AI en 2019 (quelques mois après BERT).

  Article : RoBERTa : A Robustly Optimized BERT Pretraining Approach

  Lien arXiv : https://arxiv.org/abs/1907.11692

- **CamemBERT :** Modèle dérivé de RoBERTa pour le français, a été lancé en novembre 2019 par Facebook AI Research en partenariat avec l'Inria.

  Article : CamemBERT : a Tasty French Language Model

  Lien arXiv : https://aclanthology.org/2020.acl-main.645.pdf

Le modèle BERT

- **Résumé :**

  BERT est le premier modèle Transformer bidirectionnel pré-entraîné ayant profondément transformé les NLP. Il apprend des représentations contextuelles profondes en lisant les phrases dans les deux sens simultanément.

- **Principe de fonctionnement :**

  BERT repose sur :

  ✓ Une architecture de type Transformer
  ✓ Un entraînement auto-supervisé via deux tâches :
  - Masked Language Modeling (MLM) → prédit des mots masqués dans une phrase
  - Next Sentence Prediction (NSP) → détermine si 2 phrases se suivent logiquement

  Cela permet à BERT de capturer le contexte global d'un mot.

- **Conditions d'utilisation :**
  - ✓ Textes plutôt courts à moyens (max 512 tokens)
  - ✓ Bon encodage des labels pour la classification
  - ✓ Ressources matérielles modérées à élevées (GPU recommandé)

Le modèle ALBERT

- **Résumé :**

  ALBERT est une version allégée et optimisée de BERT, conçue pour réduire drastiquement la mémoire tout en conservant de bonnes performances.

- **Principe de fonctionnement :**

  ALBERT introduit deux optimisations clés :

  - ✓ Factorisation des embeddings : Sépare la taille du vocabulaire et la dimension cachée
  - ✓ Partage des poids entre les couches : Les mêmes paramètres sont utilisés à chaque couche Transformer
  - ✓ Il remplace aussi NSP par une tâche plus efficace : Sentence Order Prediction (SOP).

- **Conditions d'utilisation :**
  - ✓ Contextes contraints en mémoire (GPU limité, cloud)
  - ✓ Projets nécessitant rapidité et légèreté
  - ✓ Bon pour prototypage et production

Le modèle RoBERTa

- **Résumé :**

  RoBERTa est une amélioration directe de BERT, optimisée au niveau de l'entraînement, sans modifier fondamentalement l'architecture.

- **Principe de fonctionnement :**

  RoBERTa améliore BERT en :

  - ✓ Supprimant la tâche NSP
  - ✓ Utilisant des datasets beaucoup plus grands
  - ✓ Augmentant le nombre d'epochs
  - ✓ Appliquant un dynamic masking
  - ✓ Utilisant des batchs plus larges

  Ces optimisations permettent un apprentissage plus robuste.

- **Conditions d'utilisation :**
  - ✓ Besoin de performance maximale

- ✓ Données en anglais
- ✓ Ressources GPU importantes

Le modèle CamemBERT

- **Résumé :**

  CamemBERT est un modèle spécialisé pour le français, basé sur RoBERTa et entraîné sur un très grand corpus francophone (OSCAR). CamemBERT peut identifier et caractériser dans un texte les noms propres, les verbes, les adverbes, les adjectifs, de distinguer toute la grammaire et la syntaxe française, avec un taux de réussite à 99%. Il s'agit d'une avancée car la plupart des modèles linguistiques sont créés à partir de données en anglais.

- **Principe de fonctionnement :**

  - ✓ Architecture identique à RoBERTa
  - ✓ Entraîné exclusivement sur du français
  - ✓ Pas de lower-casing obligatoire

  Il capture les spécificités linguistiques du français :

  - ✓ Accords
  - ✓ Morphologie
  - ✓ Syntaxe complexe.

- **Conditions d'utilisation :**

  - ✓ Données 100 % françaises
  - ✓ Classification, NER, embeddings
  - ✓ Nettoyage de texte recommandé

Tableau de synthèse

Modèle	Langue	Taille	Points forts	Limites principales	Cas d'usage recommandé
BERT	EN	Élevée	Polyvalent, stable	Lourd, dépassé	Baseline, recherche
ALBERT	EN	Faible	Léger, rapide	Moins flexible	Faible VRAM
RoBERTa	EN	Élevée	Très performant	Coûteux	Performance maximale
CamemBERT	FR	Élevée	Spécifique FR	Sensible aux données	NLP français

4) Principe de CamemBERT

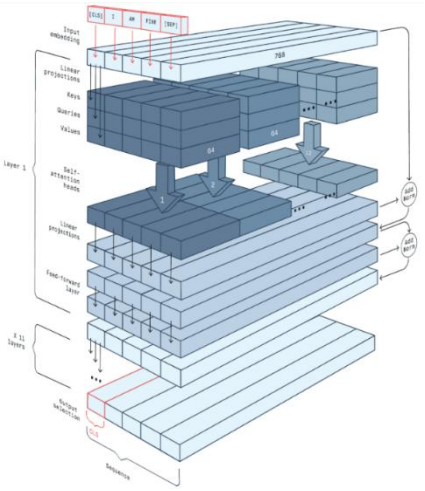
Introduction

Les modèles de langage pré-entraînés sont désormais omniprésents dans le traitement du langage naturel. Malgré leurs succès, la plupart des modèles disponibles ont été formés soit sur des corpus textuels en langue anglaise, soit sur la concaténation de données dans plusieurs langues. Cela rend l'utilisation de tels modèles limitée en France.

CamemBERT (Martin et al., 2019) est une version française des encodeurs bidirectionnels pour transformateurs (BERT). C'est un modèle linguistique de pointe basé sur l'architecture RoBERTa pré-entraînée sur le corpus multilingue OSCAR.

L'architecture de base est le Transformer : Architecture de deep learning qui permet de traiter des séquences dont les éléments sont fortement inter-dépendants, comme c'est le cas pour les mots d'une phrase par exemple.

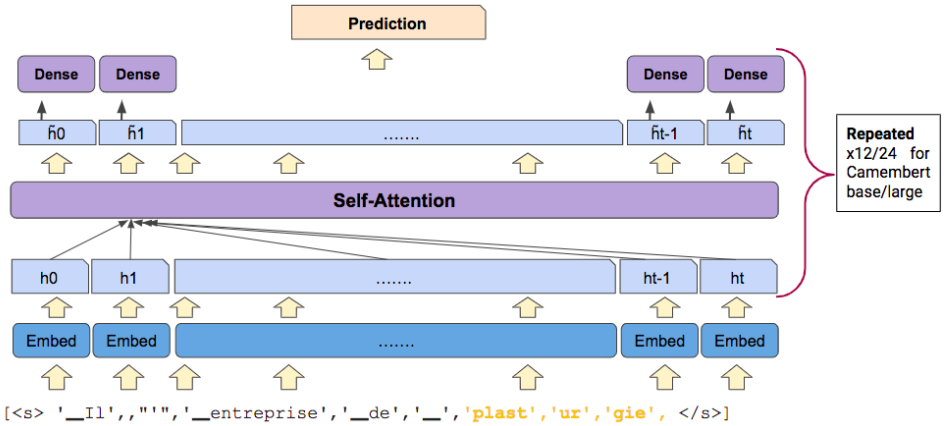
L'architecture d'un Transformer se présente ainsi :



- 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 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.

Un modèle CamemBERT est composé de :

- une couche d'embedding pour représenter chaque mot en vecteur
- 12 couches cachées composées principalement de deux types de transformations : des transformations dites self-attention et des transformations denses



5) Modélisation

Présentation

Le code développé ici met en œuvre un pipeline complet de classification d'émotions sur des textes français, et est basé sur le modèle CamemBERT.

Il inclue :

- La préparation des données
- La tokenisation
- La création de datasets PyTorch
- Le fine-tuning du modèle
- L'évaluation sur des jeux d'entraînement et de test
- L'analyse des performances via des métriques et des matrices de confusion

L'objectif est de prédire une classe émotionnelle associée à chaque texte du dataset étudié.

Préparation des données

Le dataset est divisé en :

- 80 % pour l'entraînement : data_train
- 20 % pour le test d'évaluation finale : data_test

Cette séparation garantit une évaluation non biaisée des performances.

La classe EmotionData hérite de torch.utils.data.Dataset et assure :

- La récupération du texte
- La tokenisation avec CamemBERT
- La création des tenseurs PyTorch

Chaque élément retourné correspond à une phrase tokenisée prête pour le modèle.

Tokenisation CamemBERT

Le tokenizer CamembertTokenizer.from_pretrained("camembert-base") effectue :

- Segmentation en sous-mots (SentencePiece)
- Ajout des tokens spéciaux
- Troncature et padding

Ce prétraitement rend les phrases compatibles avec le modèle Transformer.

Architecture du modèle

Le modèle pré-entraîné fournit des représentations contextuelles profondes des tokens.

La sortie du réseau de neurones est extraite puis passée dans :

- Une couche linéaire
- Une activation ReLU
- Un dropout (régularisation)
- Une couche finale de classification
- Un vecteur de scores pour les 13 classes d'émotion

Fonction de perte et optimisation

- La fonction de perte : CrossEntropyLoss, adaptée à la classification multi-classe
- L'optimiseur : Adam, utilisé pour ajuster tous les paramètres de CamemBERT

Entraînement (fine-tuning)

Étapes par batch :

- Transfert des données vers GPU / CPU
- Passage dans le modèle
- Calcul de la perte
- Rétropropagation (backpropagation)
- Mise à jour des poids

Indicateurs suivis :

- Loss cumulée
- Accuracy d'entraînement

Des logs intermédiaires permettent de suivre la convergence.

Évaluation du modèle

Deux fonctions d'évaluation sont définies :

- Évaluation sur train (fn_valid_train) :
  - ✓ Détecte un surapprentissage et compare train vs test
  - ✓ Calcule : loss, accuracy, prédictions, les classes d'émotion
- Évaluation sur test (fn_valid_test) :
  - ✓ Mesure la performance réelle
  - ✓ Calcule : accuracy, prédictions, métriques globales

Métriques de performance

Après chaque epoch on évalue :

- Accuracy
  Pprecision
- Recall

- F1-score
- Matrices de confusion

Ces métriques permettent :

- D'analyser les classes mal reconnues
- D'évaluer l'équilibre des prédictions

Visualisation des résultats

Deux matrices de confusion sont affichées :

- Sur le jeu d'entraînement
- Sur le jeu de test

Cela permet d'identifier :

- Les confusions entre émotions proches
- Les classes dominantes
- Les biais du modèle

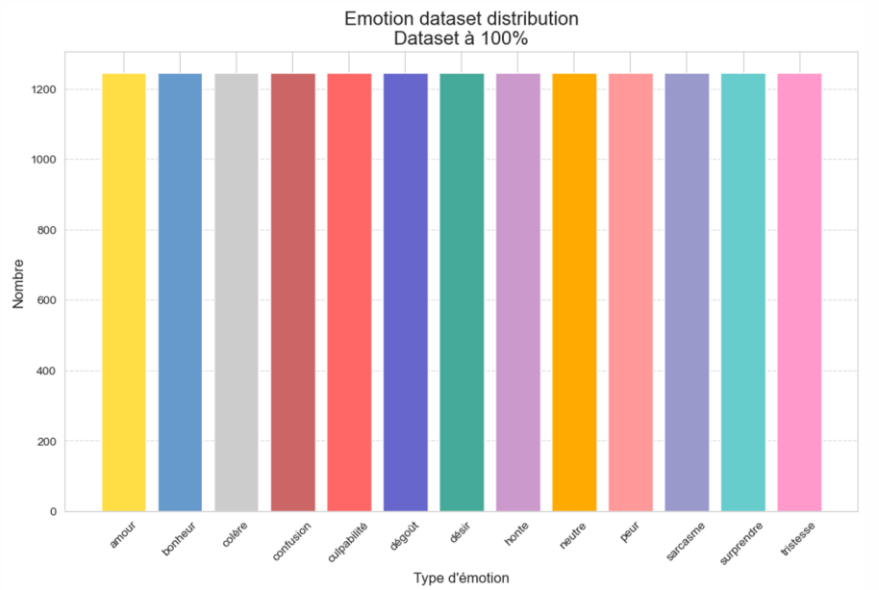
Intérêt global du pipeline

Ce code constitue un pipeline complet de classification NLP avec Transformer, intégrant :

- Prétraitement
- Tokenisation adaptée au français
- Fine-tuning du modèle pré-entraîné
- Evaluation rigoureuse
- Visualisation interprétable

6) Synthèse des résultats

Le dataset utilisé



Comparaison des métriques de performance

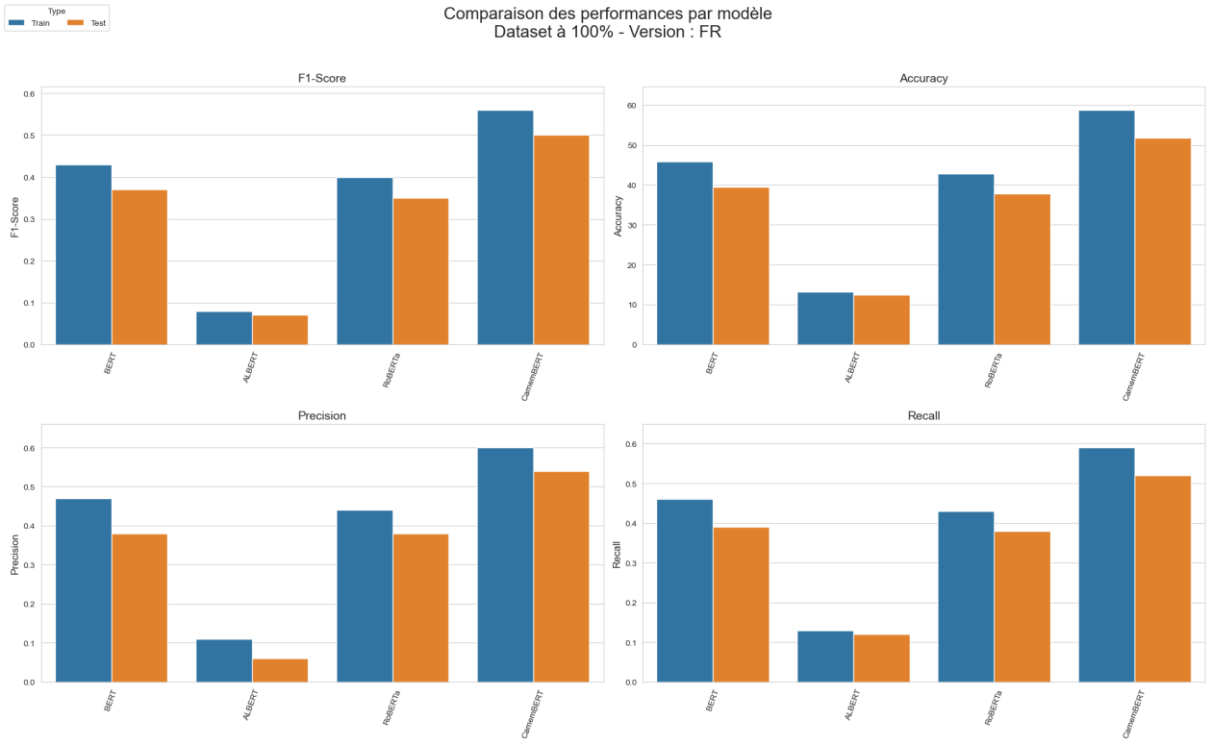
- **Tableau de comparaison des modèles testés :**

	Type	Modele	Accuracy	Precision	Recall	F1-Score	Time
0	Train	BERT	45.81	0.47	0.46	0.43	898.86
1	Test	BERT	39.49	0.38	0.39	0.37	898.86
2	Train	ALBERT	13.21	0.11	0.13	0.08	698.78
3	Test	ALBERT	12.49	0.06	0.12	0.07	698.78
4	Train	RoBERTa	42.73	0.44	0.43	0.40	808.44
5	Test	RoBERTa	37.79	0.38	0.38	0.35	808.44
6	Train	CamemBERT	58.74	0.60	0.59	0.56	772.75
7	Test	CamemBERT	51.79	0.54	0.52	0.50	772.75

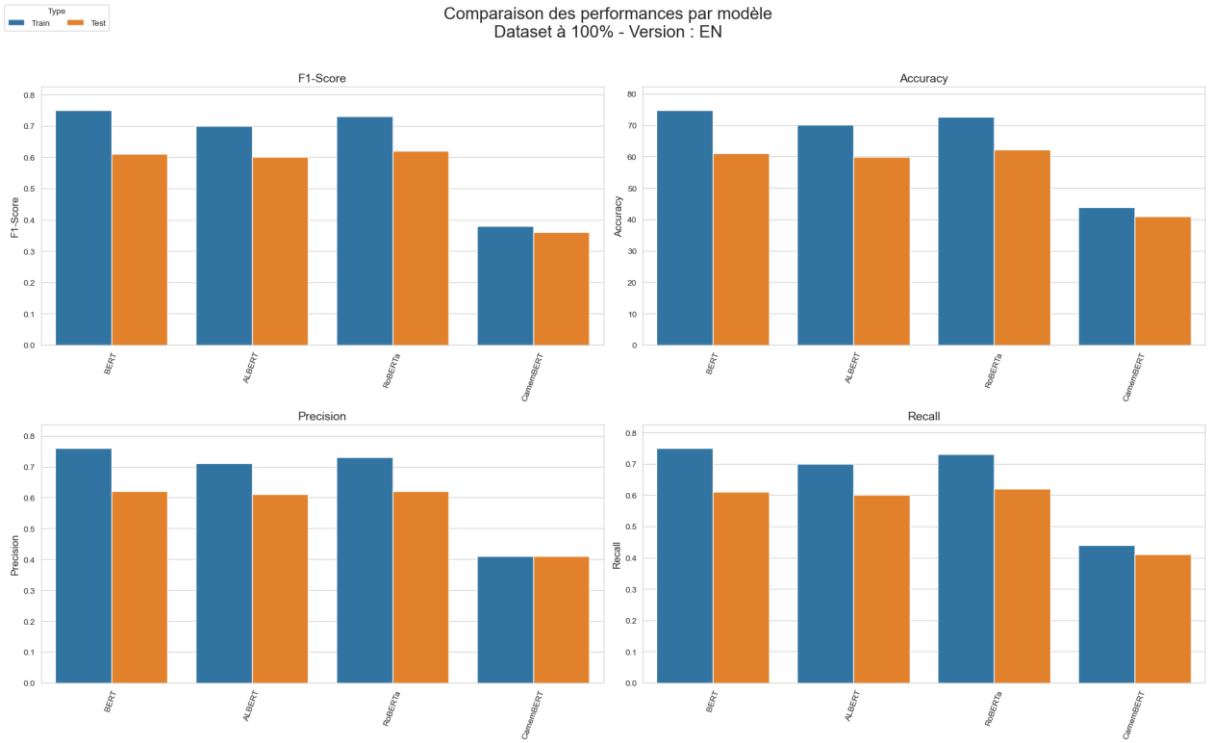
Tableau de comparaison des modèles pour un dataset français

	Type	Modele	Accuracy	Precision	Recall	F1-Score	Time
0	Train	BERT	74.76	0.76	0.75	0.75	947.84
1	Test	BERT	61.07	0.62	0.61	0.61	947.84
2	Train	ALBERT	70.06	0.71	0.70	0.70	1130.58
3	Test	ALBERT	59.96	0.61	0.60	0.60	1130.58
4	Train	RoBERTa	72.72	0.73	0.73	0.73	1679.86
5	Test	RoBERTa	62.21	0.62	0.62	0.62	1679.86
6	Train	CamemBERT	43.80	0.41	0.44	0.38	1852.21
7	Test	CamemBERT	41.03	0.41	0.41	0.36	1852.21

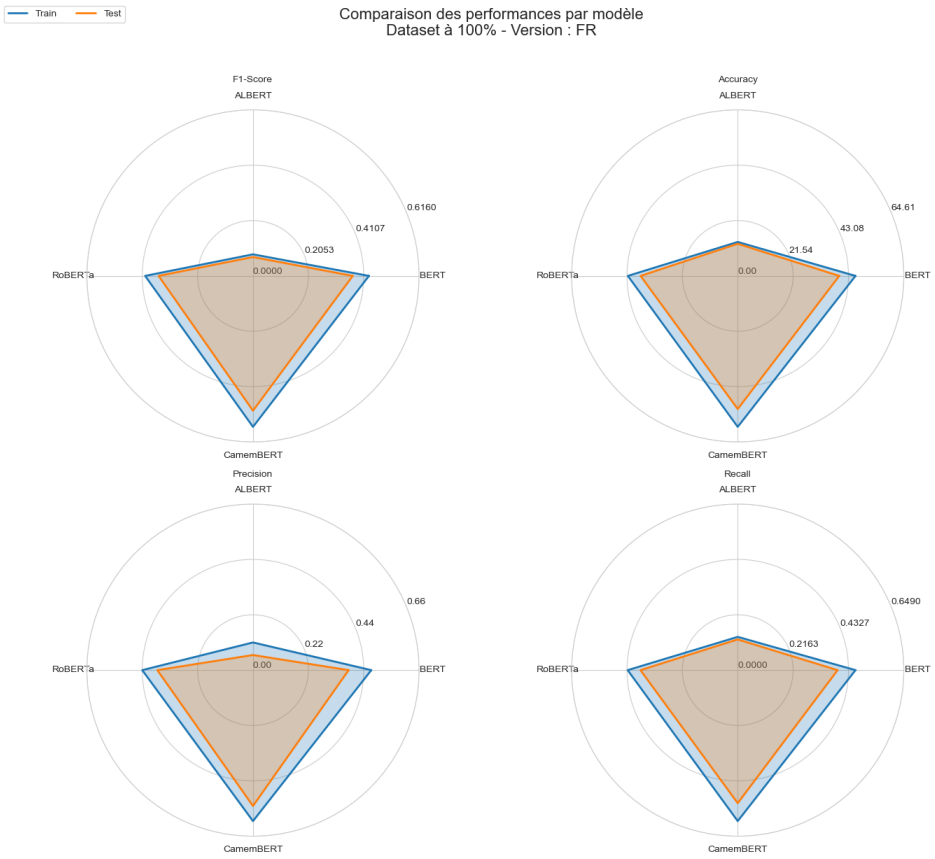
Tableau de comparaison des modèles pour un dataset anglais



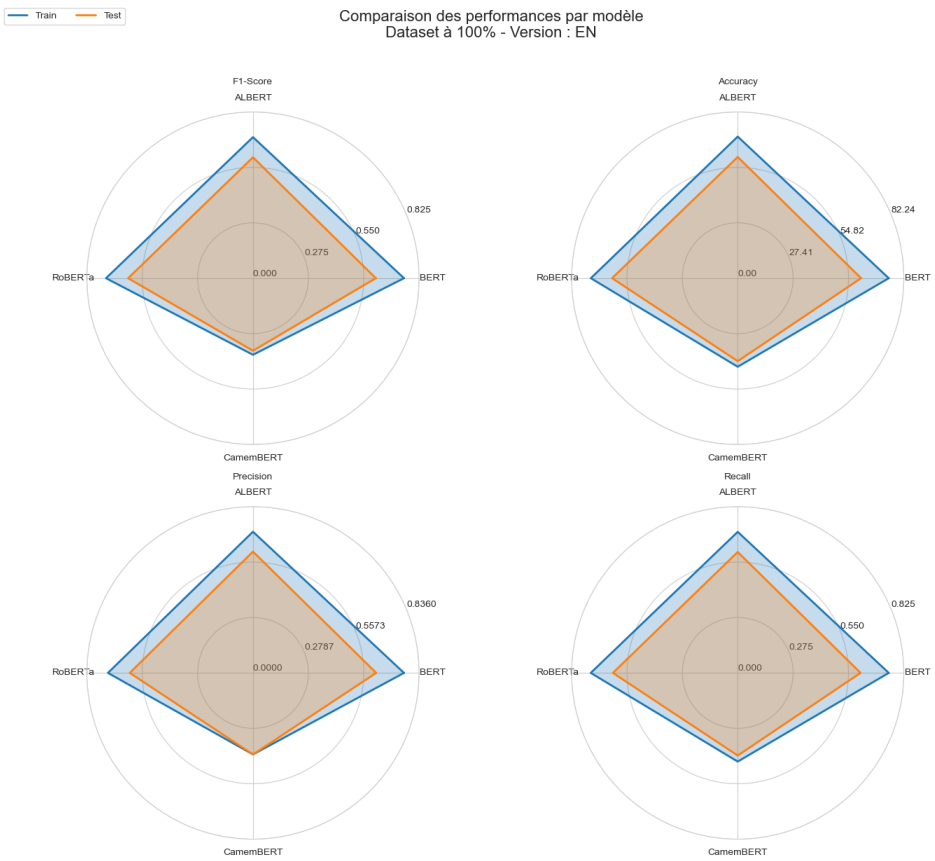
Comparaison des modèles pour un dataset français



Comparaison des modèles pour un dataset anglais

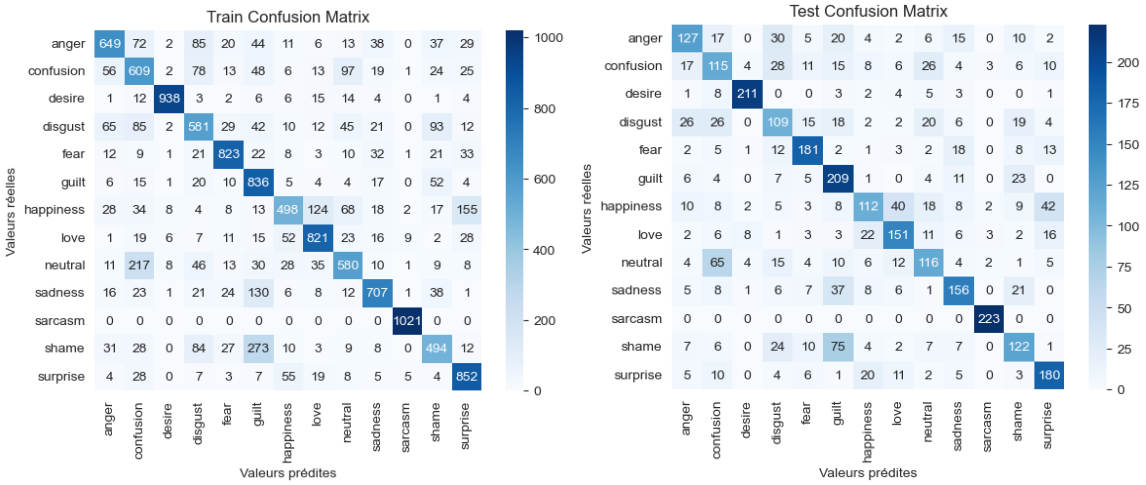


Comparaison des modèles pour un dataset français

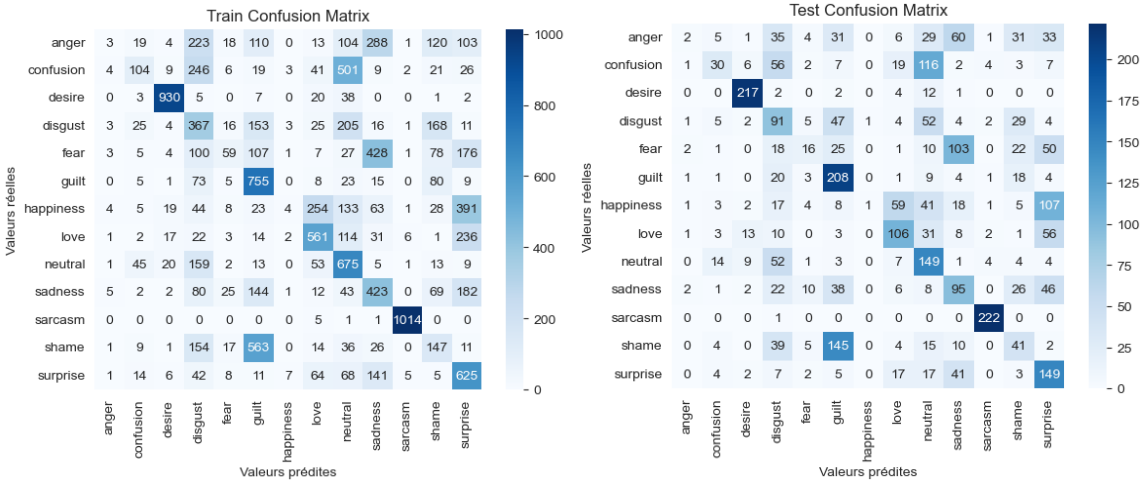


Comparaison des modèles pour un dataset anglais

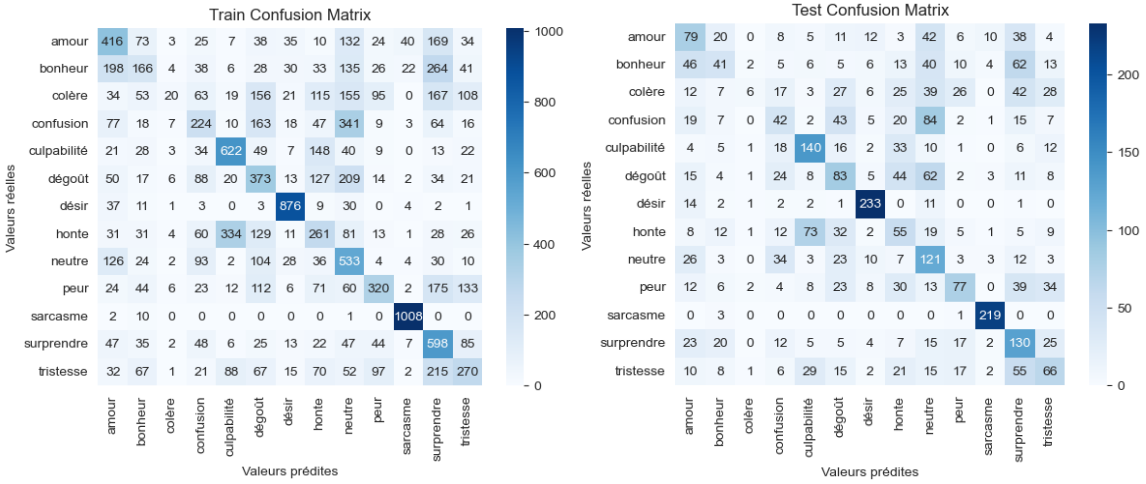
- **Matrices de confusion pour les modèles RoBERTa et CamemBERT :**



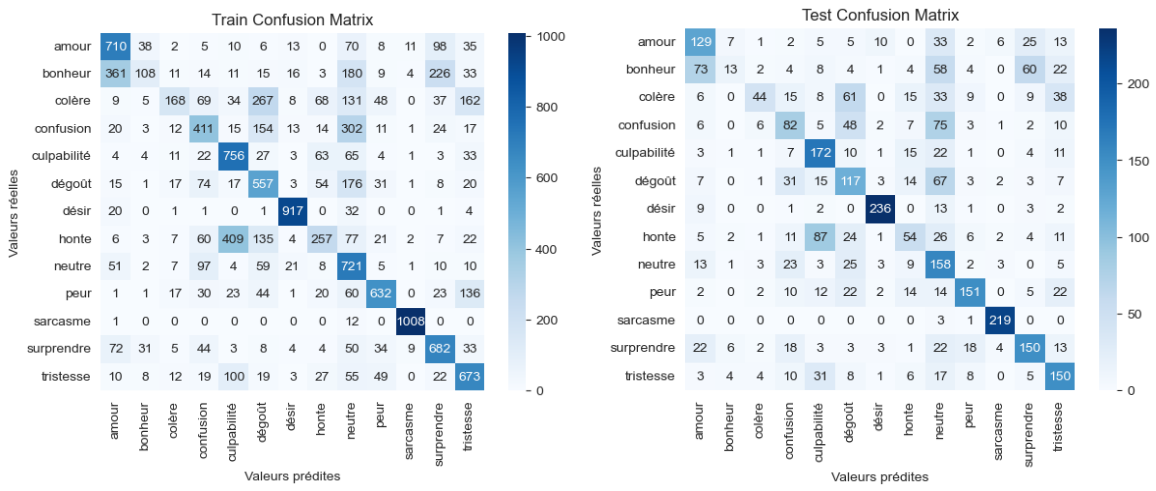
Matrices de confusion pour un dataset EN pour le modèle RoBERTa



Matrices de confusion pour un dataset EN pour le modèle CamemBERT



Matrices de confusion pour un dataset FR pour le modèle RoBERTa



Matrices de confusion pour un dataset FR pour le modèle CamemBERT

Les graphes présentés dans ce paragraphe montrent que lorsque le dataset est anglais les modèles BERT, ALBERT et RoBERTa sont les meilleurs. En revanche pour un dataset français, c'est bien le modèle CamemBERT qui se retrouve en tête comme prévu, même si localement certaines classes d'émotions sont mieux représentées par le modèle RoBERTa. Nous voyons donc ici l'intérêt d'un tel modèle lorsque nous devons traiter des textes français.

7) Importance globale et locale

Objectif

Le code proposé dans le notebook met en place une méthode d'explicabilité du modèle CamemBERT de classification de texte en utilisant Integrated Gradients (Captum) afin de :

- Fournir une importance locale. Quels mots ont le plus influencé la prédiction pour une phrase donnée. On détermine pour cela :
  - ✓ la classe prédite
  - ✓ la probabilité
  - ✓ l'importance de chaque token dans cette prédiction

- Fournir une importance globale par classe. Quels mots sont, en moyenne, les plus influents pour chaque classe d'émotion dans le jeu de test :
  - ✓ Extraction des tokens les plus importants par classe
  - ✓ Affichage des 3 mots les plus importants
  - ✓ Identification des mots-clés caractéristiques d'émotions dans le modèle

Le tout est basé sur les embeddings d'entrée du modèle.

Résultats

- **Importance locale :**

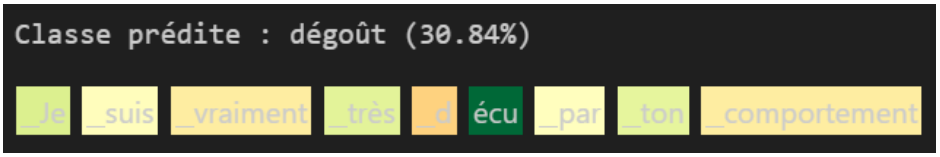
  Explication phrase par phrase, avec visualisation de l'importance de chaque mot via une heatmap textuelle. On affiche la phrase avec un fond coloré par token :

  ✓ vert → contribution positive

  ✓ rouge → contribution négative

  ✓ intensité → importance absolue

  ✓ Exemple de phrase : "Je suis vraiment très déçu par ton comportement"



  ✓ Exemple de phrase : " Je suis impatiente à l'idée de commencer mon nouveau job"

- **Importance globale :**

  Agrégation des attributions sur l'ensemble du jeu de test afin d'identifier les tokens les plus discriminants pour chaque classe d'émotion :

```
========================
Classe 2 — colère
========================
_me              0.1385
_stressé        -0.1376
_c               0.0672

========================
Classe 4 — culpabilité
========================
_comment         1.0666
_pu              0.9771
_aurais          0.9502

========================
Classe 11 — surprendre
========================
_curieux         0.7855
_très            0.7801
_attaché         0.6498
```

```
========================
Classe 6 — désir
========================
_veux            0.4627
_jai             0.3353
_je              0.2519

========================
Classe 0 — amour
========================
_chance          1.2725
_belle           0.7229
_rend            0.4214

========================
Classe 8 — neutre
========================
_agréable       -0.8532
_mépris         -0.7451
_sent           -0.7056
```

```
========================
Classe 5 — dégoût
========================
_dire            0.1337
_pense           0.1327
_quoi            0.1075

========================
Classe 1 — bonheur
========================
_images          0.6793
_entendre        0.6376
_rire            0.6275

========================
Classe 12 — tristesse
========================
_ressentir       1.3401
_fardeau         1.2556
_pression        0.8319
```

```
========================
Classe 7 — honte
========================
_non             0.4071
_mauvais         0.2007
_ressemblait     0.1691

========================
Classe 3 — confusion
========================
_colère         -0.3214
_ai              0.2973
_je              0.2468

========================
Classe 9 — peur
========================
_honte           0.6300
allait           0.3669
_chose           0.2366
```

8) Limites et améliorations

Limites du modèle CamemBERT :

- Contraintes de longueur de séquence : peu adapté aux documents longs, contrats juridiques, articles complets
- Temps de calcul élevé : Le modèle comporte environ 110 millions de paramètres, ce qui implique un temps d'inférence non négligeable, un fine-tuning coûteux en GPU, une consommation mémoire importante
- Sensibilité au bruit des données : Étant entraîné sur des données web (OSCAR), CamemBERT peut être sensible aux fautes d'orthographe, moins robuste face aux textes OCR, influencé par du langage informel
- Domaine d'entraînement généraliste : Bien que performant, CamemBERT est non spécialisé métier, moins précis sur une terminologie technique

Pistes d'amélioration :

- Adaptation à des textes longs
- Fine-tuning spécialisé à certains corpus métiers (juridique, médical, finance, ingénierie)
- Optimisation pour l'embedding de phrases (sentence-camembert-base, sentence-camembert-large)

Autres solutions :

- FlauBERT : Plus léger mais moins performant
- mBERT : Multilingue, mais moins précis pour le français
- DistilCamemBERT : Rapide, mais légère perte de précision

9) Conclusion

CamemBERT reste le modèle de référence pour le NLP en français. Cependant il présente des limites structurelles liées aux architectures type BERT :

- Contraintes de longueur
- Coût de calcul
- Manque de spécialisation métier
- Embeddings peu optimisés.