Grégory LANG
Openclassrooms
Parcours « Ingénieur Machine Learning »
19 janvier 2020

Projet 7 IML :
Détection de fake news avec le modèle CAMEMBERT

Bidirectional Encoder Representations from Transformers

Résumé :

Ce rapport présente mes travaux sur le projet 7 Openclassrooms du parcours Ingénieur Machine Learning.

Dans le but de créer un détecteur de « Fake News » françaises, j'ai entrainé en transfert learning un modèle récent camemBERT de type BERT qui est spécialement pré-entraîné en Français (Bidirectional Encoder Representations from Transformers).

Ce modèle peut-être utilisé en tant que « classifier » après l'avoir entrainé avec des news labélisées à partir de sources d'informations françaises de confiance et parodiques .

J'ai utilisé comme baseline un classifier « Régression logistique ».

Le dataset a été entièrement créé à partir de différents outils de scraping utilisés et pour certains développés spécifiquement.

Le github du code du projet est disponible ici :
https://github.com/jeugregg/FakeNewsDetectionFr

SOMMAIRE

1. La problématique et son Interprétation

1.1. Contexte et interprétation

Pour ce projet, il est demandé de développer une preuve de concept et d'effectuer une veille thématique.

Le domaine choisi est le **NLP** (**Natural Language Processing,** en français "Traitement automatique du langage naturel") et plus particulièrement les classifiers de texte.

Il est demandé de mettre en pratique un nouvel algorithme de façon autonome.

Mon choix s'est porté sur la réutilisation d'un nouveau code mais en l'utilisant avec un dataset original.

La finalité du modèle est de **détecter si le contenu d'un article est « fake »** ou « fiable ».

Ce modèle récent est comparé avec une baseline assez classique.

1.2. Pistes envisagées

Le dernier modèle prometteur dans le domaine du NLP est le modèle BERT de Google.

Une déclinaison française a été développée récemment fin 2019 : camemBERT.

Le modèle est entrainé en transfert learning pour une tache de classification.

La classification est binaire : « fake? » : **1** = « **fake** » ou 0 = « fiable ».

Le dataset français est à créer entièrement à partir de sources fiables et aussi à partir de sources parodiques.

2. Modèle camemBERT

2.1. Architecture

BERT est un modèle de réseau de neurones utilisant le principe des « Transformers Encoder-Decoder with Attention ». Son architecture lui permet d'être bi-directionnel.

L'architecture des modèles camemBERT, roBERTa et BERT est la même :

- 12 « layers »
- 768 dimensions cachées
- 12 attention « heads »
- 110 millions de paramètres.

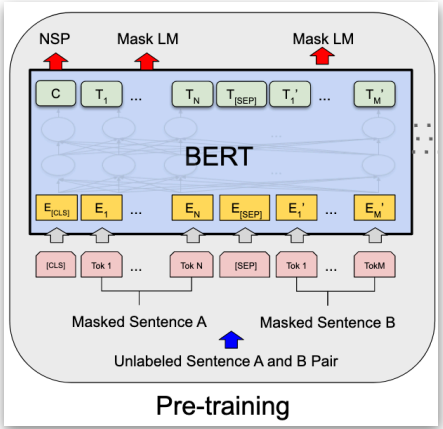
L'avantage de l'architecture de BERT est la possibilité du traitement en parallèle contrairement aux modèles de réseau de neurones récurrent existants :

- les RNN « Encoder-Decoder with Attention ».

2.2. Pré-entrainement

Les méthodes de pré-entrainement utilisées sont :

- le "masked language model" (MLM) dont l'objectif est de prédire un mot masqué dans une phrase.
- la prédiction de la phrase suivante



Par rapport à BERT, roBERTa utilise différents types d'entrainements et hyper-paramètres :

- il est entrainé plus longtemps avec plus de données 16Go -> 160Go de texte
- et il n'y a pas d'entrainement de la prochaine phrase

- le masquage des mots à prédire est dynamique à chaque phrase

- le vocabulaire est plus riche 50000 mots au lieu de 30000.

Le modele camemBERT diffère principalement de roBERTa par le fait qu'il soit entrainé avec un tokenizer different.

Enfin, camemBERT est entrainé avec le nouveau sous-corpus français d'OSCAR.

2.3. Transfert-learning pour la classification

Pour ce projet, il serait matériellement très difficile de pré-entrainer le modèle.

Le principe va être d'utiliser le modèle déjà pré-entrainé et de faire du transfert learning pour une tache de classification binaire : « fake » ou « pas fake ».

Pour cela on ajoute en sortie du modèle camemBERT un classifier avec les couches suivantes :

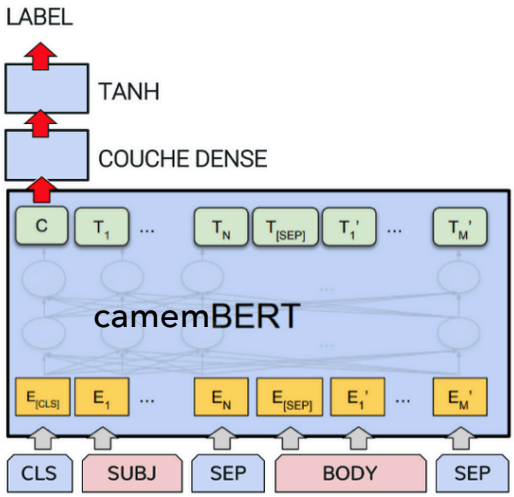
- en entrée : **n** états cachés du premier token (sortie du modele de base camemBERT)
- Couche intermédiaire « dropout »
- Couche **dense** avec **n** entrées / **n** sorties
  - fonction d'activation : tanh
- *Couche intermédiaire « dropout »*
- 1 couche finale **linéaire** : **n** entrées / **1** sortie

La fonction de perte utilisée est MSELoss : Mean Square Error (erreur quadratique).

Cette transformation est disponible dans le github suivant: https://github.com/ThilinaRajapakse/simpletransformers#text-classification

L'utilisation est simplifié car il suffit de fournir en entrée du modèle le dataFrame d'entrainement et celui de test.

Ils contiennent une colonne le texte et une colonne de classe, sans pré-traitement.



Pour l'entrainement, comme la vitesse de convergence du modèle est aléatoire et comme à chaque nouvelle « epoch », le répertoire de sortie du modèle pèse 900Mo de plus, il est nécessaire de limiter le nombre d'itérations (max = 33).

L'entrainement du modèle est répété 10 fois (en repartant de zéro) afin d'obtenir un meilleur résultat. (environ 45 min de calcul).

Probablement, que son entrainement sur plus d'itération donnerait un résultat équivalent, ou bien, meilleur.

L'utilisation d'un GPU est indispensable. J'ai utilisé le cloud computing avec Google Colab (accélération du calcul de l'ordre de x 10).

cf. fichier code GitHub : 03_Train_evaluate_camemBERT.ipynb

3. Modèle baseline

Le modèle servant de baseline utilisé est le modèle « **Regression logistique** » :

- LogisticRegression de « scikit-Learn ».

Pour pouvoir l'utiliser, il faut faire un pré-traitement du texte pour le nettoyer.

Voici les étapes de la **préparation du texte** pour la Baseline :

- Agrégation du « Title » et du « Body »

- Mise en minuscule

- Suppression caractères « espace» « tabulation » non désirés

- Suppression tag HTML

- Suppression des nombres

- Suppression de la ponctuation
  remplacement par espace

- Suppression des mots les plus utilisés
  création d'une liste de Stop Words = 100 mots les plus utilisés + `nltk.corpus.stopwords.words('french'))`

- Création de tokens
`nltk.word_tokenize(text, language='french')`

- Extraction de la racine
  `FrenchStemmer()`

Ensuite, il faut calculer la fréquence de chaque mots par rapport au corpus complet considéré.

Pour cela on utilise successivement :

- CountVectorizer avec
  - un vocabulaire de 10000 mots,
  - une fréquence minimum de 10 mots
  - une fréquence maximum de 11% du corpus

```
X_counts : CountVectorizer(max_df=0.11, min_df=10, max_features=10000)
```

- puis TfidfTransformer

```
X_tfidf = tfidf_transformer.transform(X_train_counts)
```

Cela constitue l'entrée du modèle de Régression logistique.

cf. fichier code GitHub : 04_Train_evaluate_baseline.ipynb

4. Création du dataset des news

Le but est de récupérer des news depuis des sources françaises pour préparer les données d'entrainement et de test du modèle de détection de fake news.

Les données sont originales. Elle ont été récupérées depuis les sites d'informations avec « scrapy » et en partie avec l'outil newspaper_crawler.

4.1. Création de différents « spiders » Scrapy

Les news de certaines sources ont été récupérées en utilisant Srapy à travers leurs pages web.

Cf. code : 02_Scraping_French_News.ipynb

Pour une source fiable :

- 20 minutes

Pour les sites parodiques => « fake » :

- Le Gorafi
- buzzbeed.com
- NordPresse.be

4.2. Utilisation du GitHub gbolmier/newspaper-crawler

Cf. code : 01_Scraping_French_newspaper_crawler.ipynb

La récupération des article est fait grace un web scraper de flux RSS de journaux français. (Source : https://github.com/gbolmier/newspaper-crawler)

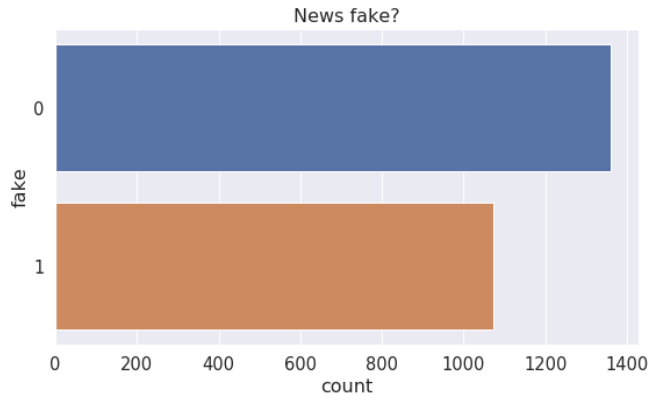
Les sources récupérées sont :

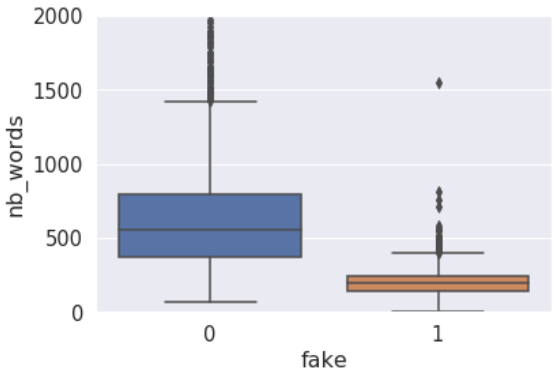
- Le Monde (avec correction proposée sur GitHub issue #1)
- Le Figaro
- Libération
- Futura Sciences

5. Exploration du dataset des news

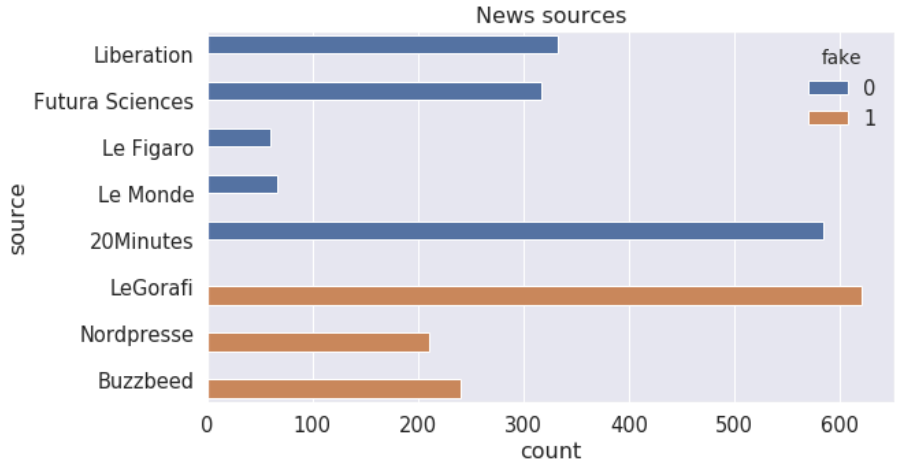
L'ensemble du dataset contient 2432 articles.

Il contient 45% de fake news.
Elle contiennent 60% de mots en moins par rapport au news fiables.





La répartition des sources montre que Le Gorafi et 20 Minutes sont le plus représentés.



Cela est dû au fait que la plupart des articles du Monde et du Figaro ou même de Liberation sont payants et donc peu accessibles.

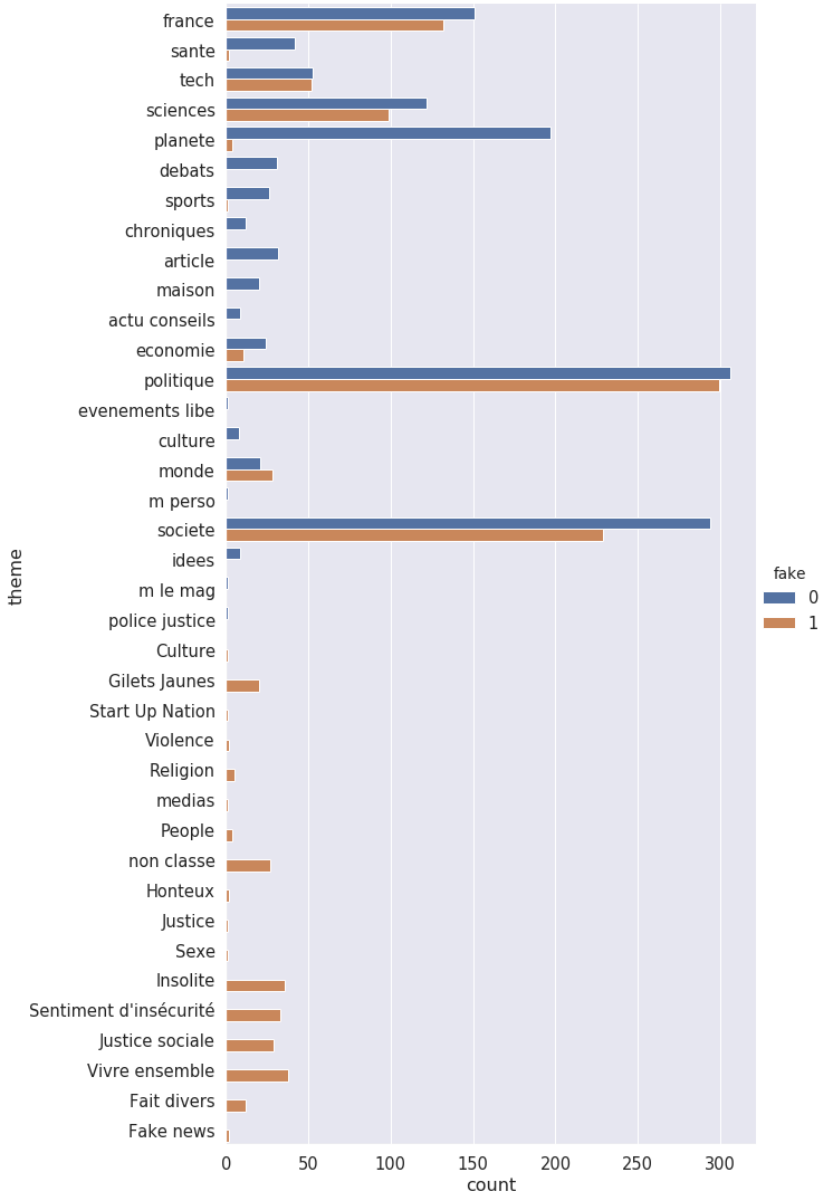
De plus, pour ces journaux, on utilise la méthode des flux RSS qui contiennent moins d'articles : uniquement les derniers en date.

En ce qui concerne les thèmes abordés dans les articles, 3/4 des news sont dans un thème équilibré fake/fiable :

- france
- technologies
- sciences
- politique
- société
- monde

Mais de nombreux thèmes reste peu représentés.

Ce sont souvent des sous-thèmes ou des thèmes très précis en majorité fake.



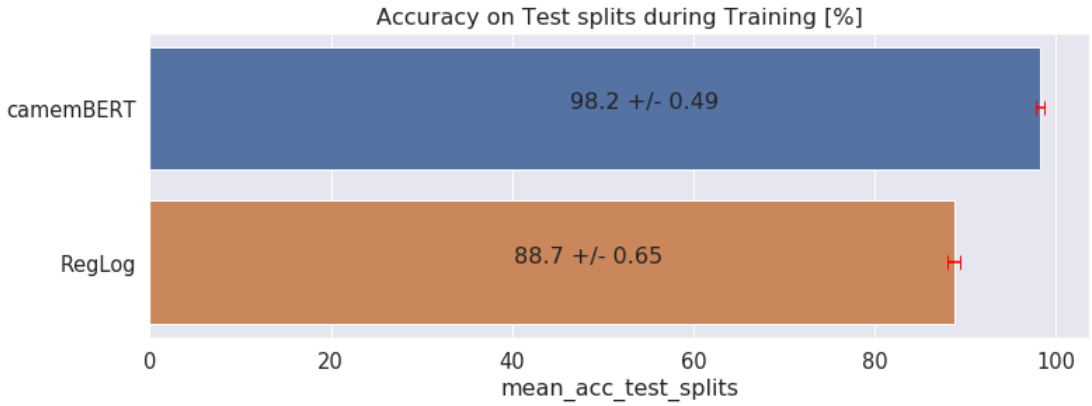
6. Résultats des modèles

6.1. Accuracy

Une validation croisée a été effectuée sur 5 splits avec StratifiedShuffleSplit, ce qui permet de choisir aléatoirement des news dans l'ensemble du jeu d'entrainement de façon à garder une répartition de news fake équivalente avec l'ensemble du jeu.

La taille des train splits est de 70% du jeu de « train » (test splits = 30%)

On trouve une très bonne accuracy de camemBERT sur les test splits > 98%

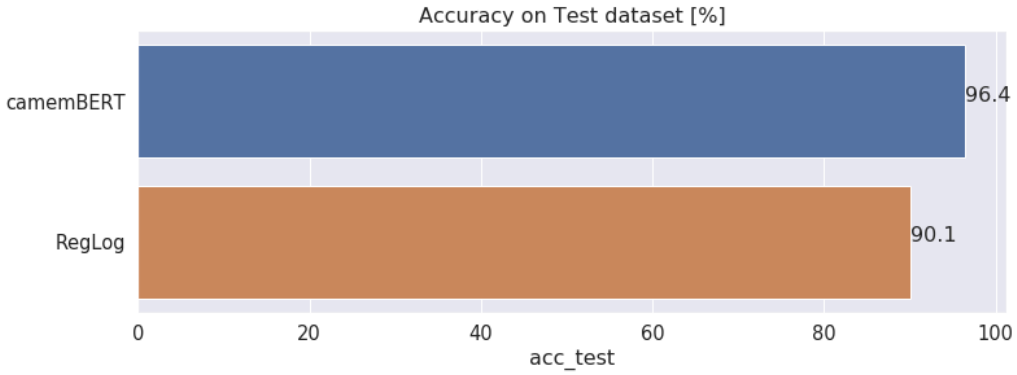


On remarque un faible intervalle de confiance.

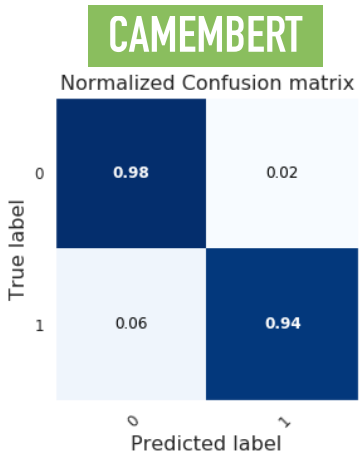
Ensuite, on observe une bonne accuracy de camemBERT sur jeu de « test » : > 96%

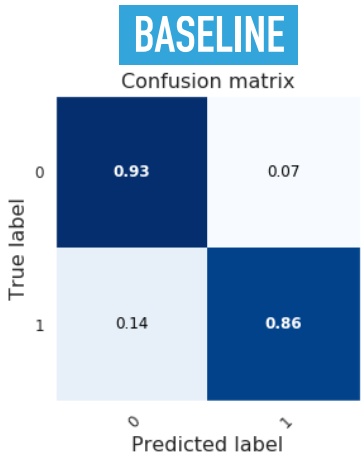
On s'attend donc à une bonne généralisation du modèle.

Au regard du niveau générale élevé de l'accuracy, peut être qu'il serait mis plus à l'épreuve avec d'autres sources non utilisés pour l'entrainement.



6.2. Matrices de confusion





Le modèle camemBERT est assez équilibré.

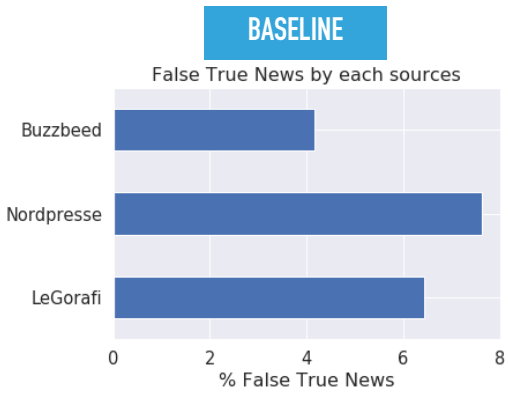
Il a néanmoins tendance à prédire légèrement moins bien les fake news que les « fiables » (vrais positif : 94%, vrais négatif : 98%)

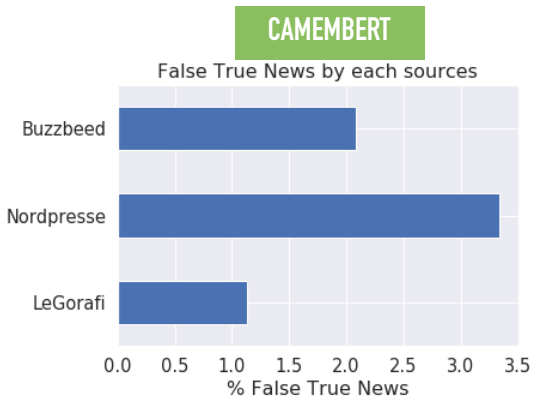
En comparaison avec la Baseline, camemBERT est meilleur pour détecter les fake news avec 94% de vrais positifs contre 86%.

Il a aussi un niveau plus faible d'erreur de prédiction « fiable » alors qu'elle sont « fake » avec 6% (Baseline 14%).

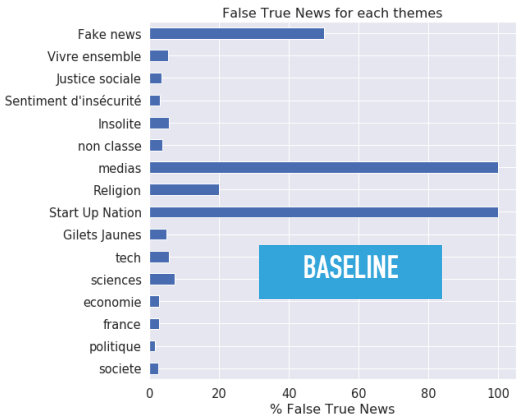
La BaseLine est moins équilibrée car elle est sensiblement moins efficace pour détecter les fake news (86% de vrais positifs contre 93% de vrais négatifs).

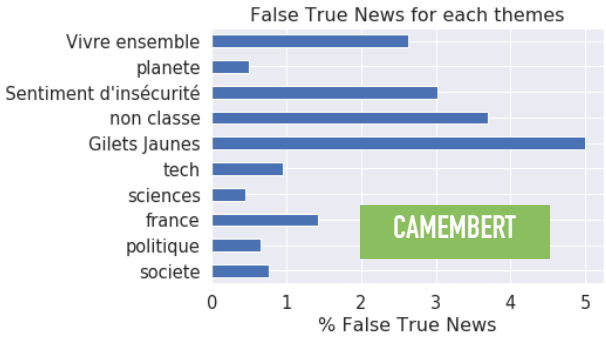
Pour les deux modèles, les erreurs « fake prédite fiable » sont plus nombreuses pour NordPresse.be (ce sont de courtes news par rapport aux autres sources).



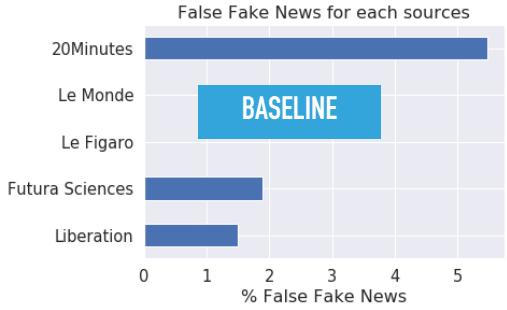


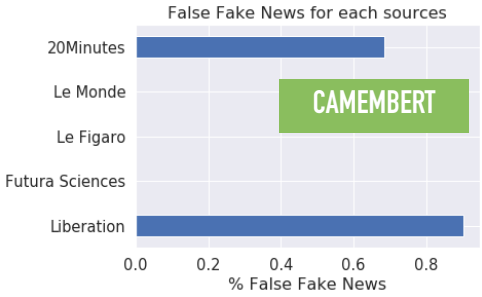
Le thème des fakes prédites fiables sont différentes en fonction du modèle :



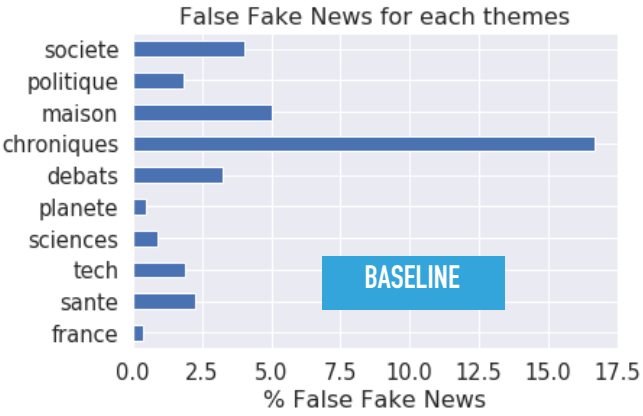


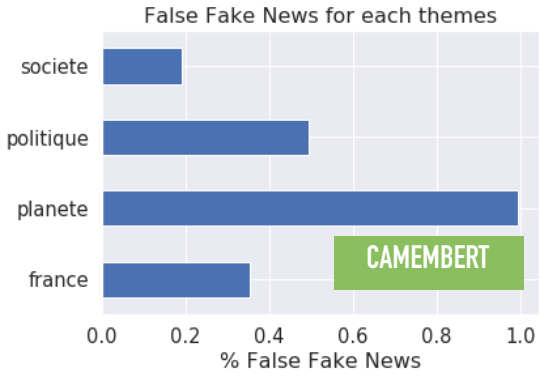
On n'observe pas de faux positif (fiable prédite fake) pour Le Monde et le Figaro :





Le thème « chroniques » est sensiblement le plus difficile pour la baseline :





Les « chroniques » , issues principalement du journal Libération, sont des contenus souvent plus longs et qui présentent plutôt un avis personnel.

7. Conclusions

Le modèle camemBERT se révèle être un très bon classifier.

Il montre une accuracy de +10% / baseline.

Mais, c'est un modèle beaucoup plus lourd à entrainer.

Il est indispensable d'avoir a disposition un GPU et un espace disque important (> 40 Go voir bcp plus en fonction du nombre d'epochs)

Le temps de calcul pour entrainer camemBERT est de 50 min environ sur Google Colab avec un environment GPU. Ce qui représente 330 epochs au total.

En comparaison, le modele baseline est très rapide (5min) et peu encombrant (50 Mo).
Mais il implique un pré-traitement qui double la quantité de données d'entrée.

8. Axes d'améliorations

Il serait interessant de tester le modèle avec d'autres sources inconnues du modèle.

Aussi, on peut imaginer améliorer l'entrainement sur plus d'« epocs ».
Actuellement le fait de répéter 10 fois l'entrainement sur 33 epochs ne garantie pas une meilleure convergence. Il serait préférable de lancer sur 100 epochs par exemple (nécessite 100 Go d'espace disque).

Les fake news sont relativement évidentes en générale car plutôt parodiques.
Que ce passerait-il si on invente une news qui peut sembler réelle ?

Il serait interessant d'avoir un niveau de confiance avec des classes intermédiaires : fiable - plutôt fiable - plutôt fake - fake. Cela nécessiterait d'avoir plus de source et une labélisation par l'humain.

Aussi, on pourrait avoir plus de news en améliorant le scraping des journaux par pages plutôt qu'avec les flux RSS qui a une limite dans le temps. Mais attention le Monde interdit cela.

Enfin, il est possible de créer un site web (API) de détection de la fiabilité d'un article à l'aide de ce modèle.

9. Sources bibliographiques

Code du modèle **BERT** : https://github.com/google-research/bert
Papier sur le pré-entrainement de **BERT** : https://arxiv.org/abs/1810.04805

Papier sur le modèle **camemBERT** : https://arxiv.org/abs/1911.03894
Code du modèle **camemBERT** : https://camembert-model.fr/
Corpus de camemBERT, corpus français d'**OSCAR** : https://traces1.inria.fr/oscar/

Article **transfert learning en classifier** : https://medium.com/@vslovik/fake-news-detection-empowered-with-bert-and-friends-20397f7e1675
Code pour utiliser camemBERT en classifier : https://github.com/ThilinaRajapakse/simpletransformers#text-classification

Article sur la **Baseline** : https://towardsdatascience.com/full-pipeline-project-python-ai-for-detecting-fake-news-with-nlp-bbb1eec4936d

Code pour la **récupération** de quelque sources de **news françaises** : https://github.com/gbolmier/newspaper-crawler