Projet AirParadis

Détection des Bad Buzz

-

Article de blog

Présentation du travail réalisé et des résultats obtenus

1 - Contexte

Le but de cet article est de présenter le travail réalisé pour le projet AirParadis et d'en faire un retour d'expérience.

AirParadis est une compagnie aérienne qui souhaite créer un produit IA permettant d'anticiper les 'Bad Buzz' sur les réseaux sociaux.

2 - Objectif du projet

Le but du projet est de créer un prototype d'un produit IA permettant de prédire le sentiment associé à un Tweet.

Le projet sera réalisé à partir d'un jeu de données comportant des Tweets et le sentiment lié : 'Positif' (représenté par le chiffre 1) ou 'Négatif' (représenté par le chiffre 0).

3 – Approches testées et métrique retenue

Le projet a été réalisé en travaillant avec les services dédiés au Machine Learning de Microsoft Azure.

Trois différentes approches ont été implémentées :

- Approche 1 : 'API sur étagère'
- Approche 2 : 'Modèle sur mesure simple'
- Approche 3 : 'Modèle sur mesure avancé'

La métrique retenue pour l'évaluation des différentes approches est **l'Accuracy**. Le choix s'est porté sur cette métrique car :

- Les données sont équilibrées (autant de Tweets avec sentiment positif et négatif)
- On souhaite prédire les sentiments positifs et négatifs à égalité

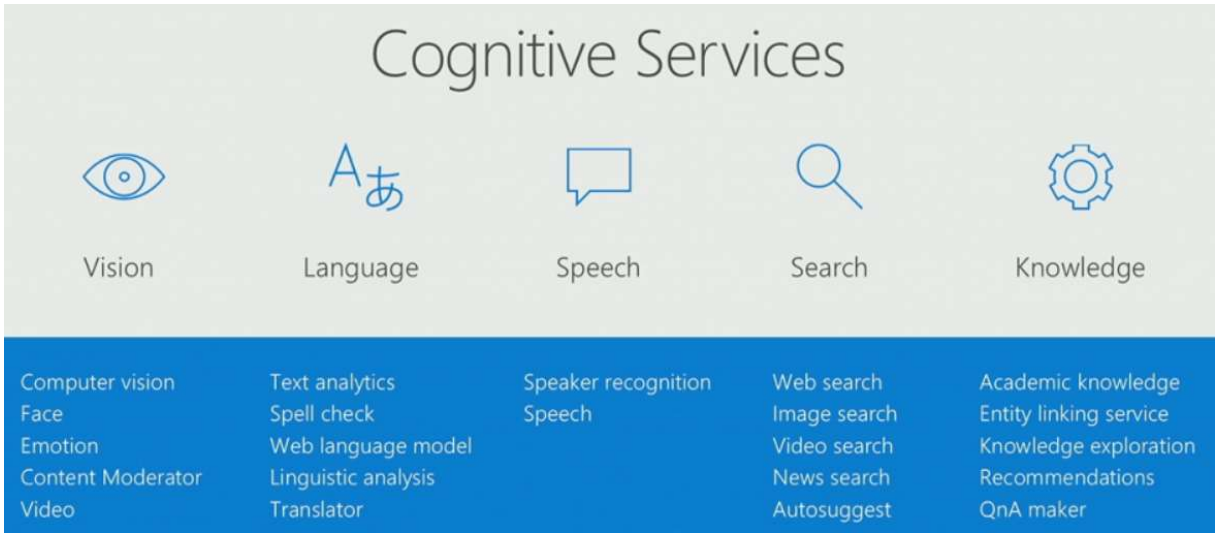
Les résultats des trois approches sont comparés en utilisant un même jeu de données de 2000 Tweets.

4 – Retour d'expérience : présentation de l'approche, de la mise en place et du résultat obtenu

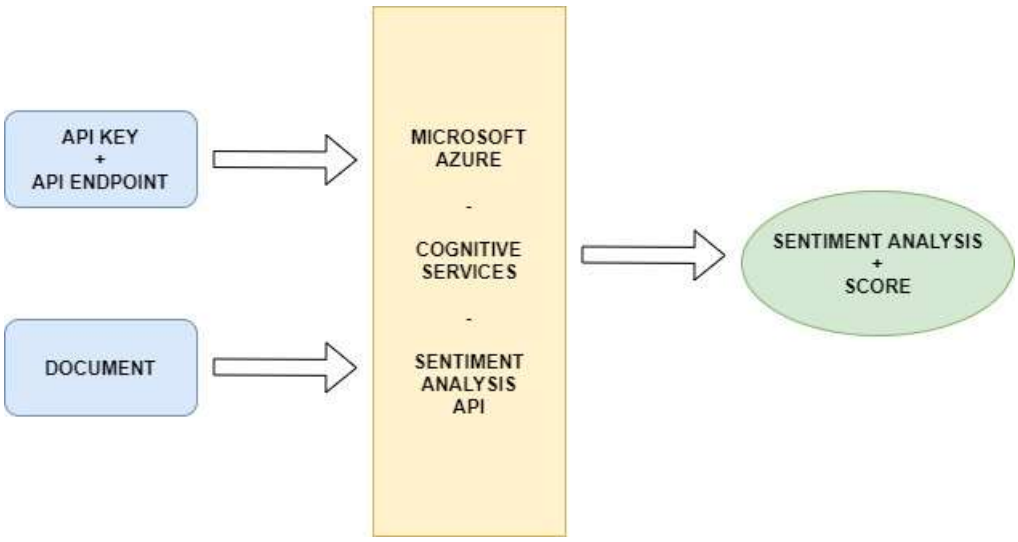
Approche 1 : 'API sur étagère'

Cette approche utilise le service 'Azure Cognitive Service' pour l'analyse de sentiment.

Azure Cognitive Service regroupe plusieurs modèles déjà entrainés et déployés dans les domaines suivants :



Ces services sont accessibles grâce à une API REST :



Après la création du service dans Azure, nous avons développé un script Python permettant d'interroger le service à distance. Les données sont échangées au format JSON.

Les étapes du script sont les suivantes :

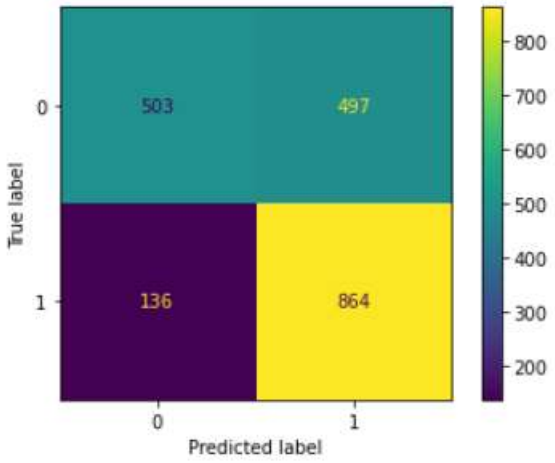
- Chargement des données
- Récupération de la clé d'authentification et du endpoint de connection à l'API Azure dans les variables d'environnement
- Authentification auprès de l'API
- Envoi de la requête et récupération du résultat

Le service renvoie une probabilité (score) pour chacun des sentiments suivants : 'Positif', 'Neutre' et 'Négatif'.

Notre jeu de données ne comportant pas de sentiment 'Neutre', nous avons adapté le résultat renvoyé de la façon suivante : le sentiment est considéré comme négatif si le score lié au sentiment négatif est supérieur à 0.5 et positif dans le cas contraire.

Nous obtenons une Accuracy de **0.550**.

Matrice de confusion :



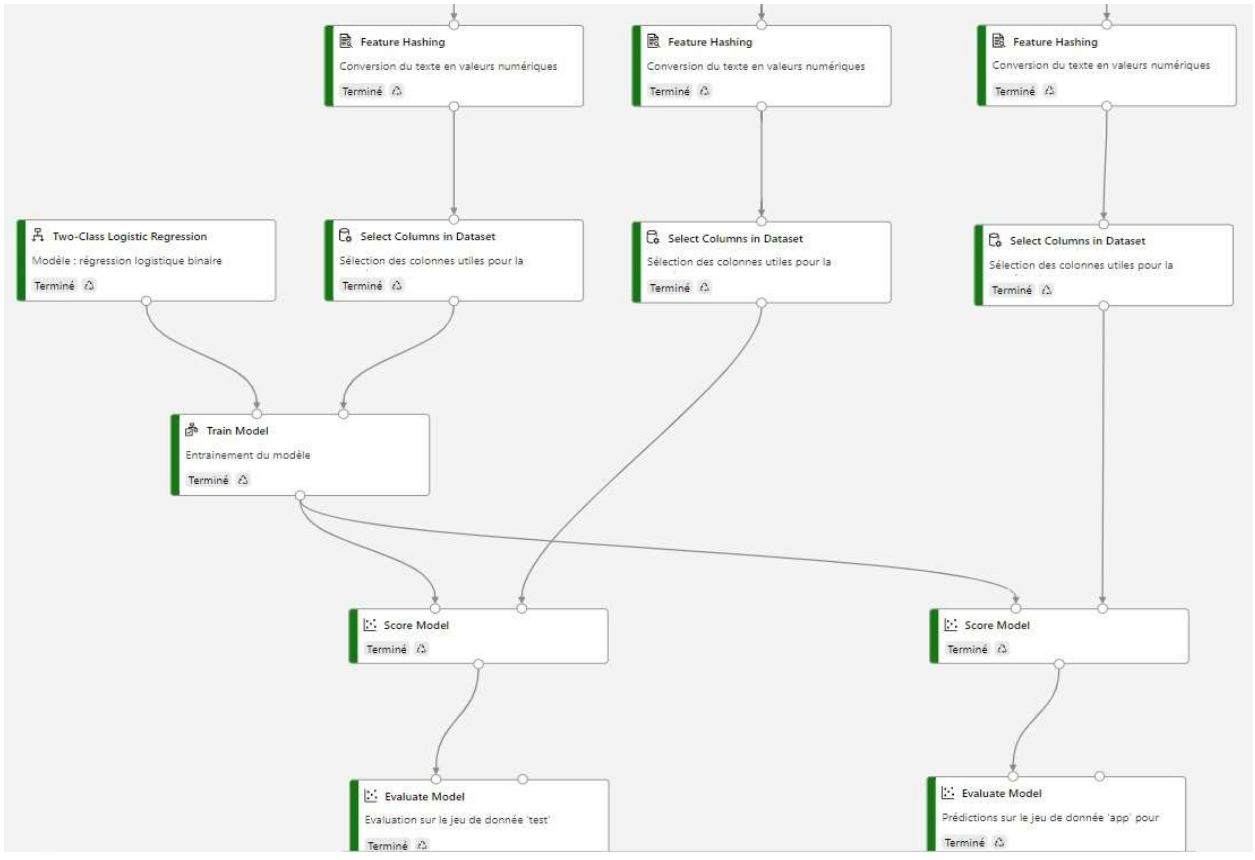
Approche 2 : 'Modèle sur mesure simple'

Cette approche utilise le service 'Designer' fourni par Azure Machine Learning.

Ce service permet de développer un modèle de Machine Learning classique de façon 'Drag&Drop' dans une interface graphique. Cette interface permet de choisir les différentes éléments composants le modèle (Dataset, Preprocessing, Datasplit, Train, Score…) sous forme de boîtes et de les enchainer.

Voici le modèle que nous avons réalisé :


p7_tweets_dataset
Select Columns in Dataset
Sélection des colonnes utiles pour l'analyse
Terminé
Partition and Sample
Sélection d'un échantillon de données
Terminé
Convert to Dataset
Remplacement de la valeur 4 par 1 pour le
Terminé
Split Data
Séparation du jeu de données
Terminé
Preprocess Text
Preprocessing des Tweets
Terminé
Preprocess Text
Preprocessing des Tweets
Terminé
airparadis_app_dataset
Edit Metadata
Renommage colonne
Terminé
Edit Metadata
Renommage colonne
Terminé
Select Columns in Dataset
Sélection des colonnes utiles pour l'analyse
Terminé
Preprocess Text
Preprocessing des Tweets
Terminé


Feature Hashing
Conversion du texte en valeurs numériques
Terminé
Feature Hashing
Conversion du texte en valeurs numériques
Terminé
Feature Hashing
Conversion du texte en valeurs numériques
Terminé
Two-Class Logistic Regression
Modèle : régression logistique binaire
Terminé
Select Columns in Dataset
Sélection des colonnes utiles pour la
Terminé
Select Columns in Dataset
Sélection des colonnes utiles pour la
Terminé
Select Columns in Dataset
Sélection des colonnes utiles pour la
Terminé
Train Model
Entrainement du modèle
Terminé
Score Model
Terminé
Score Model
Terminé
Evaluate Model
Evaluation sur le jeu de donnée 'test'
Terminé
Evaluate Model
Prédictions sur le jeu de donnée 'app' pour
Terminé

Ce modèle est une régression logistique binaire prenant en entrée des Tweets préprocessés et convertis en valeurs numériques.

Nous obtenons une Accuracy de **0.755**.

Matrice de confusion :

Predicted label	Actual_1	Actual_0
Predicted_1	751	241
Predicted_0	249	759

Approche 3 : 'Modèle sur mesure avancé'

Cette approche a consisté à développer un modèle sur mesure dans un Notebook Jupyter en Python puis le déployer en utilisant les services de Azure.

Nous avons tout d'abord **préparé les données** en procédant notamment à l'identification et l'analyse de la cible ainsi qu'à la séparation du jeu de données.

Ensuite nous avons réalisé le **preprocessing** des données textuelles. Nous avons utilisé deux techniques de prétraitement : un nettoyage des Tweets et une lemmatisation. Nous avons utilisé les librairies Prepocessor et SpaCy.

Puis nous avons développé un modèle de référence basé sur **un algorithme de Machine Learning classique** : une régression logistique et une représentation des mots de type TF-IDF. Nous avons utilisé la librairie SciKitLearn.

Avec ce modèle nous obtenons une Accuracy de **0.776**.

Nous sommes ensuite passés à **la modélisation Deep Learning** en utilisant différentes architectures de réseaux de neurones (RNN, LSTM, GRU, Uni et Bi-directionnels) avec plongement de mots appris par le réseau puis fourni au réseau (modèle GloVe classique préentrainé). Nous avons utilisé les librairies Keras et TensorFlow.

Nous avons également calculé les temps d'exécution afin de comparer les modèles.

Le modèle présentant les meilleurs résultats est l'architecture GRU Unidirectionnel utilisant le plongement de mots fourni au réseau. Le fait de fournir le plongement de mots permet d'améliorer la performance mais surtout réduire l'overfitting qui est présent quand le plongement de mots est appris par le réseau.

Nous avons optimisé ce modèle sous 3 axes.

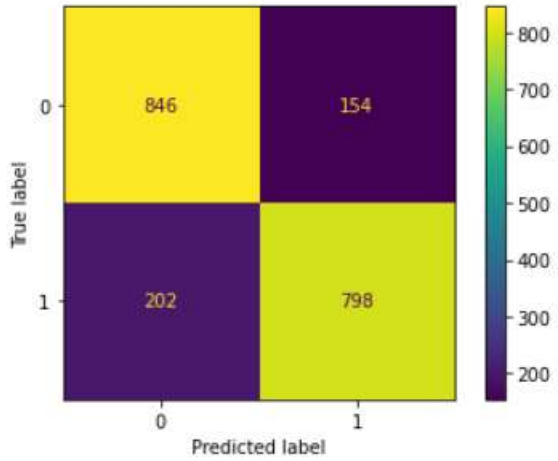
Nous avons tout d'abord **optimisé le plongement de mots utilisé**. Pour cela nous avons utilisé un modèle FastText entrainé sur les Tweets de notre jeu d'entrainement et un modèle GloVe préentrainé sur un très grand nombre de Tweets. C'est le modèle GloVe préentrainé sur un très grand nombre de Tweets qui donne les meilleurs résultats.

Puis nous avons optimisé les **hyper paramètres** Dropout et Learning Rate en utilisant KerasTuner et une recherche de type Random.

Et enfin nous avons cherché le nombre d'**epochs** qui donne le meilleur résultat.

Nous obtenons pour le meilleur modèle une Accuracy de **0.822**.

Matrice de confusion :



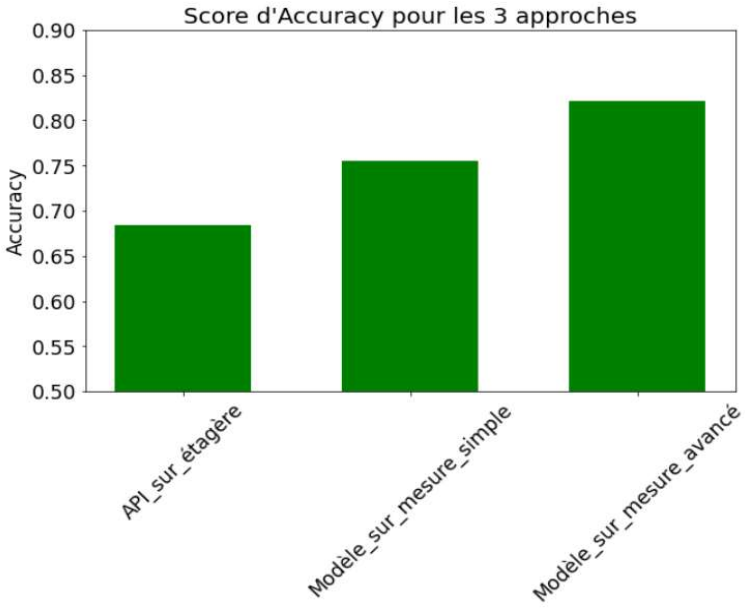
Pour finir nous avons déployé ce modèle en utilisant les services Azure.

Nous avons remarqué que le nettoyage des Tweets améliore le résultat, mais pas la lemmatisation.

L'analyse des erreurs nous a montré que la majorité d'entre elles viennent d'une ambiguïté dans le sentiment du Tweet ou d'une erreur de labélisation.

4 – Conclusion

Tableau récapitulatif des résultats obtenus :



Le modèle donnant la meilleure Accuracy est le modèle sur mesure avancé de Deep Learning (GRU), suivi par le modèle sur mesure simple puis le modèle API sur étagère.

On remarque qu'il y a un lien entre le niveau de complexité du modèle et de sa mise en place et le résultat obtenu : plus le modèle est 'avancé', meilleur est le résultat.