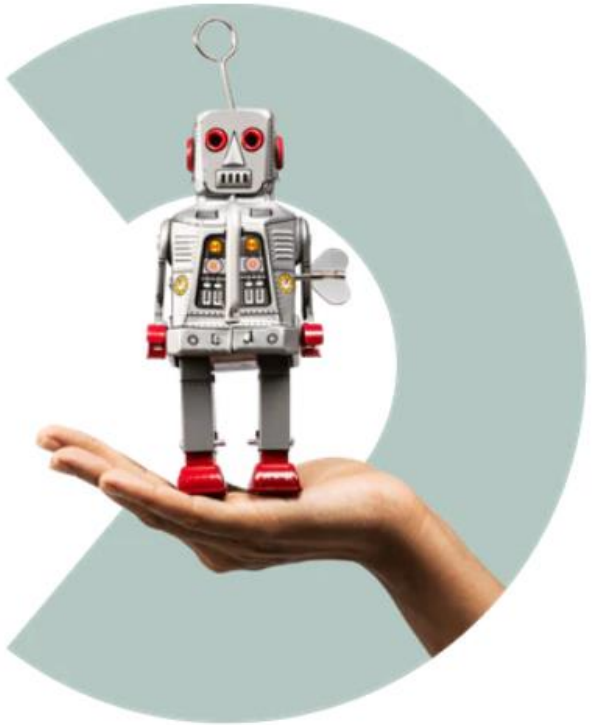
PROJET 7

DÉTECTEZ LES BAD BUZZ GRÂCE AU DEEP LEARNING:
PRÉDIRE LE SENTIMENT ASSOCIÉ À UN TWEET

#NLP #DEEP LEARNING #LSTM #TRANSFORMER BERT #ANALYSE DE SENTIMENT
#ENTRAINEMENT #DEPLOIEMENT
#AZURE MACHINE LEARNING #AZURE COGNITIVE SERVICES
#KERAS #HUGGING FACE



Ingénieur IA

Développez et intégrez des algorithmes de Deep Learning au sein d'un produit IA

OPENCLASSROOMS

OUDDANE NABIL

SOMMAIRE

Projet 7

Détectez les Bad Buzz grâce au Deep Learning

A INTRODUCTION

A. Contexte

- ENJEU global:
  - Air Paradis veut un prototype d'un produit IA permettant de **prédire le sentiment associé à un tweet**.

- ENJEU Compétences DU P7:
  - NLP / deep learning / Azure

- DONNEES SOURCES
  - Pas de données clients chez air paradis
  - Données Kaggle 'sentiment140 » 1600000 tweets annotés de 0 (négative) à 4 (positive)
    - https://s3-eu-west-1.amazonaws.com/static.oc-static.com/prod/courses/files/AI+Engineer/Project+7%C2%A0-+D%C3%A9tectez+les+Bad+Buzz+gr%C3%A2ce+au+Deep+Learning/sentiment140.zip

B. Objectifs

- SCRIPT:
  - **Préparer un modèle fonctionnel du modèle:**
    - Envoie de tweet
    - Récupération de sentiment
  - **3 approches à tester**
    - **<u>API sur étagère : API service cognitif Azure</u>** *pour l'analyse du sentiment (attention très couteux, donc seulement quelques tests sur quelques milliers de tweets)*
    - **<u>Modèle bespoke simple</u>** *(type régression logistique par exemple) avec le designer graphique de Azure Machine Learning*
    - **<u>Modèle bespoke avancé</u>** *(type deep learning) toujours avec Azure Machine Learning, à déployer et à montrer à air paradis*
      - *2 word embeddings différents à tester (word2vec, glove, fasttext*
      - *Modele keras de base*
      - *Modele keras avec couche LSTM*
      - *Modèle BERT*

C. Ressources complémentaires

- Deploiement API de prediction: https://s3.eu-west-1.amazonaws.com/course.oc-static.com/projects/Webinars/Data/API_Dec_2020/corrige%CC%81_deploy_ml_api.zip
- Librairie BERT HuggingFace:
  - Principe de BERT: https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente
  - Exemple de code: https://swatimeena989.medium.com/bert-text-classification-using-keras-903671e0207d
  - Lib : https://huggingface.co/transformers/
  - Github: https://github.com/google-research/bert
- Azure:
  - Text analytics: https://docs.microsoft.com/fr-fr/azure/cognitive-services/text-analytics/quickstarts/client-libraries-rest-api?tabs=version-3-1&pivots=programming-language-python
  - Azure Machine Learning: https://docs.microsoft.com/fr-fr/learn/paths/build-ai-solutions-with-azure-ml-service/
  - Jupyter dans Azure Machine Learning: https://docs.microsoft.com/fr-fr/azure/machine-learning/tutorial-1st-experiment-sdk-setup
  - Exemple de notebook dans Azure ML: https://s3-eu-west-1.amazonaws.com/course.oc-static.com/projects/Ing%C3%A9nieur_IA_P7/img-classification-part2-deploy.ipynb
  - Deployer un modèle dans Azure Machine Learning: https://docs.microsoft.com/fr-fr/azure/machine-learning/tutorial-deploy-models-with-aml
  - Exemple designer Azure ML: https://docs.microsoft.com/fr-fr/azure/machine-learning/samples-designer
- KERAS:
  - Analyse de sentiment : Modèle simple: https://docs.microsoft.com/fr-fr/learn/modules/analyze-review-sentiment-with-keras/2-build-and-train-a-neural-network
  - Analyse de sentiment : modèle LSTM: https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47

AZURE ML: Les différentes fonctionnalités de l'environnement

Connection workspace
Via azure ML python SDK
Et Fichier de config

DATASTORE:
-un conteneur BLOB ET FICHIERS

DATASET à inscrire:
Tabulaire et file

ENVIRONNEMENT VIRTUEL :
Gestion automatique
ou
Specification:
inscription de l'environnement
Configuration des conteneur d'environnement

CIBLE DE CALCUL
Creation / recherche / utilisation

Experience: Création et Exécution INLINE OU ENCAPSULEE:
2 éléments clés
1) Configuration de script:
définition du script à exécuter
définition de l'environnement d'exécution
2) Script python de l'expérience

PIPELINE
Possibilité de le publier pour faire du on demand via un endpoint

AUTOML
Automatiser la sélection de modèles

HYPERDRIVE
Reglage des hyperparamtres

Interpretabilité
azureml-interpret et azureml-contrib-interpret

SMARTNOISE
Confidentialité différentielle

FAIRLEARN
Attenuer la PARTIALITE

ENTRAINEMENT

Inscription / enregistrement des modèles
À partir d'un fichier local ou d'un objet run de l'entrainement
pour:
Versionning, inference

CONFIGURATION D'INFERENCE:
InferenceConfig
Script de scoring *avec 2 fonctions init et run permettant de charger le modèle enregistré et de demander la prédiction*
Création de l'environnement d'exécution

Possibilité de batch pipeline pour le non real time avec parallélisation et planification

CONFIGURATION DE DEPLOIEMENT
Cible de calcul
Definition de la config

DEPLOIEMENT

SUPERVISION des modèles en production avec **application insights**

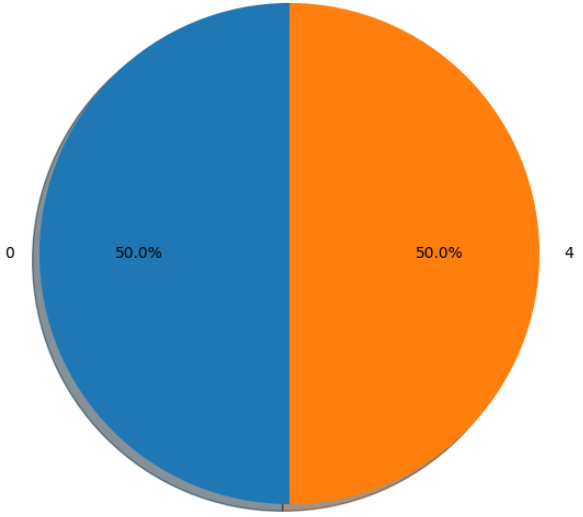
Surveiller la **DERIVE DES DONNEES** avec azureml.datadrift

B

ANALYSE DE SENTIMENT SUR MACHINE LOCALE AVEC GPU

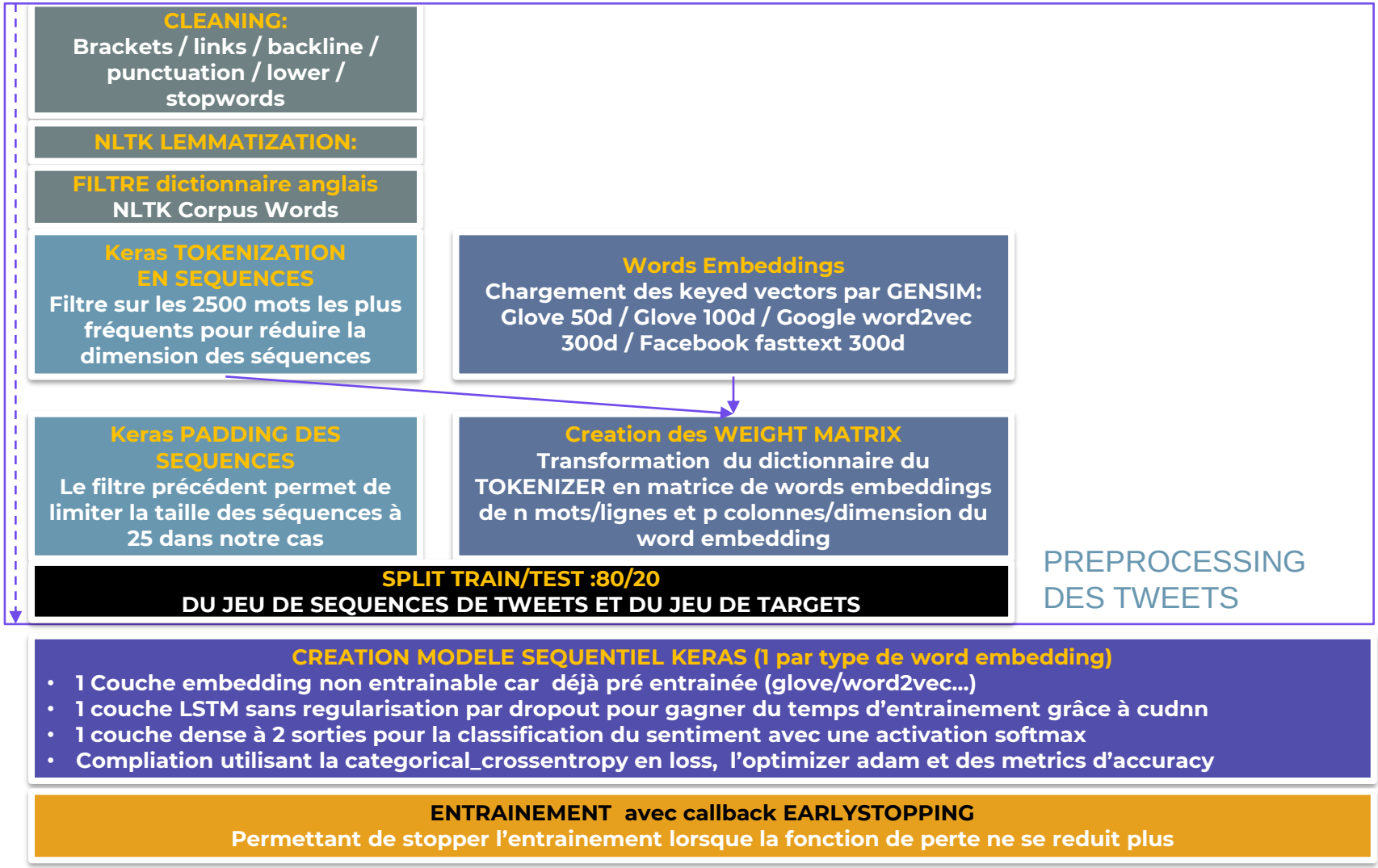
A- Analyse rapide du fichier sentiment 140

- Lecture en encoding:LATIN pour eviter les problèmes de décodage
- **1.6 millions de tweets**
- quelques doublons avec des targets différents
  - 3400 lignes en moins après drop duplicates
- **660000 users**:
  - le plus prolifique a 550 tweets
- **2 targets**/sentiments 0 et 4 **équilibrés** (50/50)



B-LSTM

Entrainement LSTM sur machine locale avec GPU



B-LSTM

Entrainement LSTM sur machine locale avec GPU

- Summary du modèle 1 layer LSTM avec word embeddings glove 50 d

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 25, 50)            1672500

 lstm (LSTM)                 (None, 50)                20200

 dense (Dense)               (None, 2)                 102

=================================================================
Total params: 1,692,802
Trainable params: 20,302
Non-trainable params: 1,672,500
```

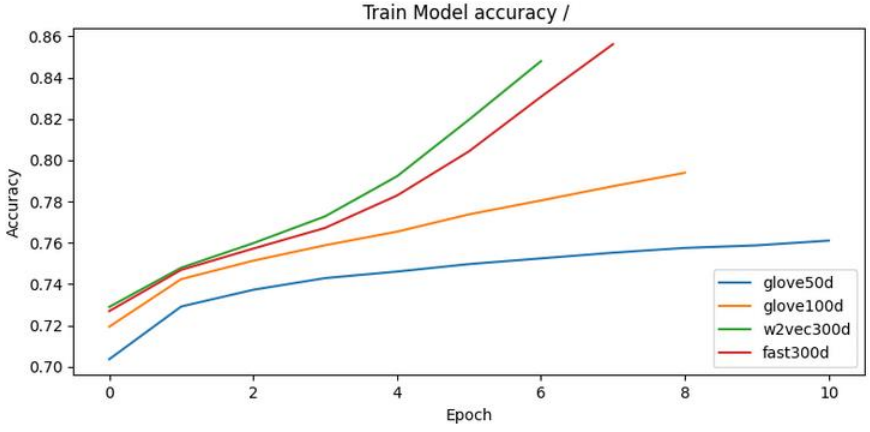
- Summary du modèle 1 layer LSTM avec word embeddings word2vec 300d

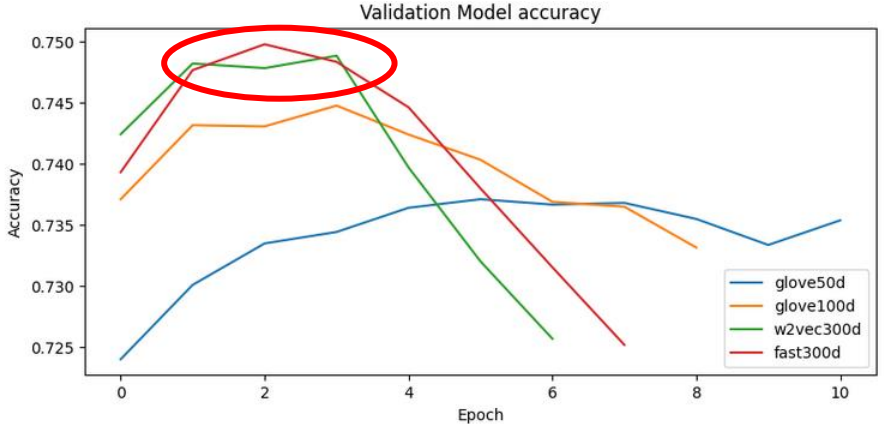
```
model3 None
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 25, 300)           10035000

 lstm_3 (LSTM)               (None, 300)               721200

 dense_3 (Dense)             (None, 2)                 602

=================================================================
Total params: 10,756,802
Trainable params: 721,802
Non-trainable params: 10,035,000
```

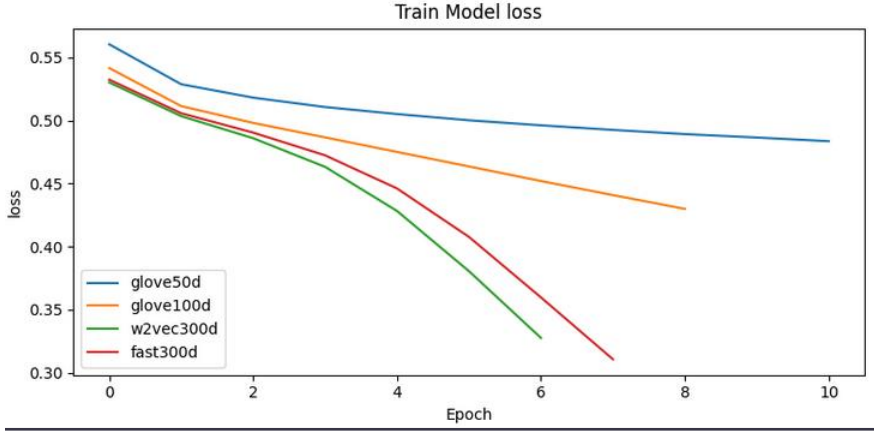
B-LSTM

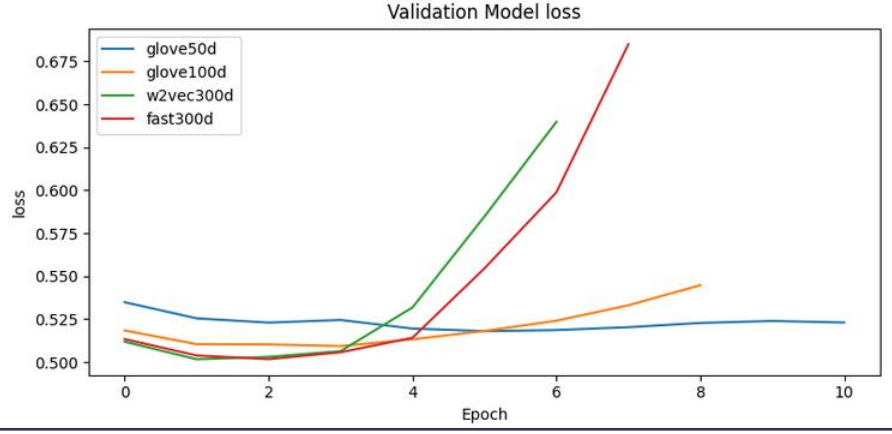
Entrainement LSTM sur machine locale avec GPU

- 500000 tweets / 4 modeles entrainés en fonction du word embeddings
- Early stopping sur validation loss min avec une patience de 5 epochs
  - **Entrainement ok autour de 2 à 5 epochs / accuracy croissante avec le nombre de dimension du we**
  - **Validation/Test Accuracy max autour de 0,75 avec des words embeddings de 300d**









C- Transformer BERT Introduction

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - Sorti des labos de Google fin 2018
  - Plus performant que ses prédécesseurs en terme de résultats et de rapidité d'apprentissage
  - Une fois pré-entraîné, de façon non supervisée il possède une "représentation" linguistique qui lui est propre.
    - Il est ensuite possible, sur la base de cette représentation initiale, de le customiser pour une tâche particulière.
  - Enfin il peut fonctionner de façon multi-modèle, en prenant en entrée des données de différents types comme des images ou/et du texte, moyennant quelques manipulations

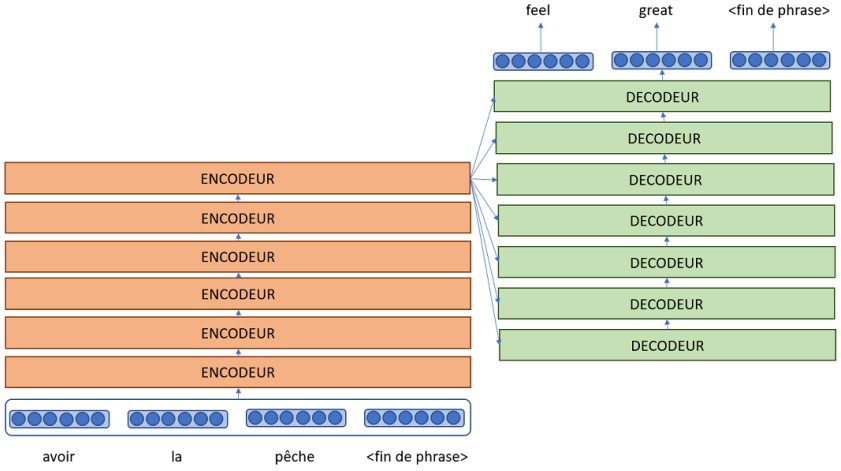
- **BERT peut :**
  - ***faire de la traduction***
  - ***comparer le sens de deux phrases pour voir si elles sont équivalentes.***
  - ***générer du texte.***
  - ***décrire et catégoriser une image***
  - ***déterminer si tel élément est un sujet, un verbe, un complément d'objet direct***
  - ***répondre à des questions***

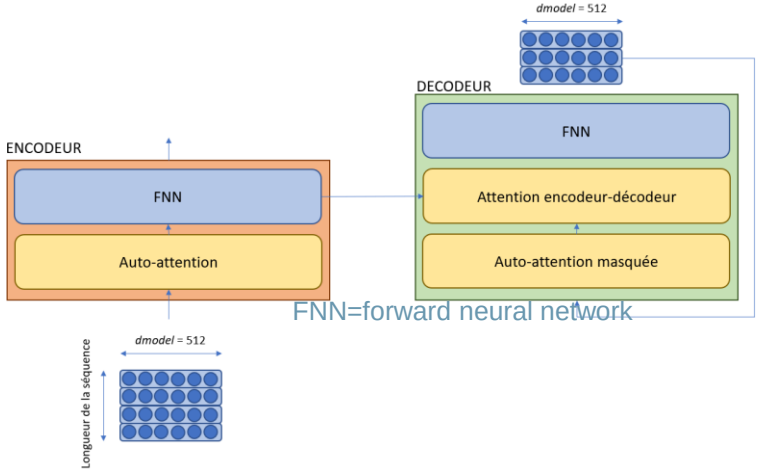
- **BERT n'utilise qu'une partie de l'architecture Transformer.**
  - Comme son nom l'indique (Bidirectional Encoder Representations from Transformers) Bert n'est composé que d'un empilement de blocs type "Encodeur" sans "Décodeur"

C- Transformer BERT

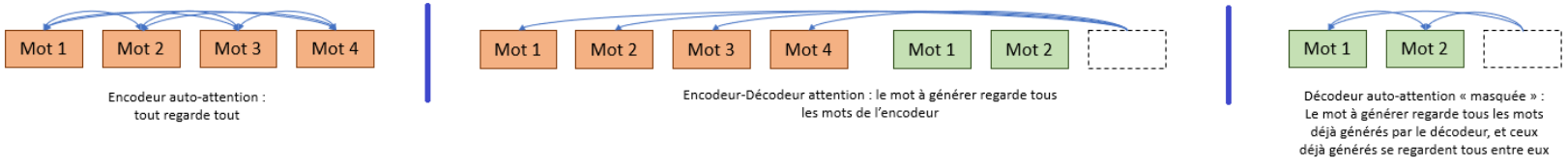
Architecture d'un Transformer:

un encodeur qui lit le texte et un decodeur qui fait une prediction avec des masques d'attention





FNN=forward neural network



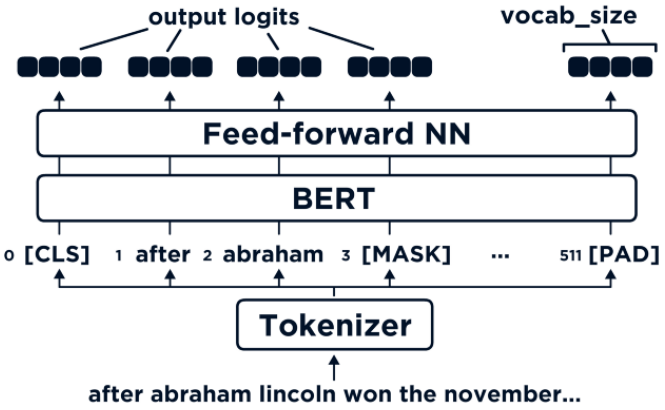
C- Transformer BERT

Principe

- **Son fonctionnement bidirectionnel lui permet d'avoir une bien meilleure compréhension que les modèles unidirectionnels**
  - Il va masquer aléatoirement des mots d'une séquence pour essayer de les prédire en regardant simultanément le contexte de gauche et de droite
  - il applique un **mécanisme d'attention** pour comprendre les relations entre les mots de la phrase, quelles que soient leurs positions respectives
  - Le but de BERT étant de générer un modèle de langage , il n'utilise que la partie encodeur
  - *Plutôt qu'un fonctionnement bidirectionnel, il vaudrait mieux dire un fonctionnement non directionnel, BERT se sert de tout le voisinage de contexte*
- **BERT utilise 2 stratégies d'apprentissage :**
  - ***Masked LM:***
    - ***15% de la séquence est plus ou moins masquée et le modèle tente de prédire ces mots masqués en affectant des probabilités a chaque mot du dictionnaire:***
      - ***80% de ces 15% sont remplacés par un mask token***
      - ***10% de ces 15% sont remplacés par un mot aléatoire***
      - ***10% de ces 15% sont remplacés par le vrai mot***
    - ***Cette approche bidirectionnelle converge moins vite qu'une approche unidirectionnelle classique mais donne de meilleurs résultats***
  - ***Next sentence prédiction:***
    - ***Des paires de phrases sont envoyées en entrée et le modèle tente de dire si la seconde phrase est vraiment connectée à la première***

C- Transformer BERT
Focus sur le MLM

- **MLM**
  - Consiste à obtenir la même phrase en sortie qu'en entrée
  - La phrase en entrée est partiellement masquée et bert doit trouver les mots manquants grâce au contexte
    - Exemple: « en automne, les _____ tombent des arbres
  - Le token le plus probable sera celui du dictionnaire qui aura la plus grande probabilité
- **Processus**
  - **Tokenization -> on récupère 3 tenseurs**
    - *Input_ids -> représentation tokenizée du texte , représente également le tenseur de label dans le cas du MLM*
    - *attention_mask-> permet de dire ou se situe les mots après padding de séquences de tailles différentes*
    - *token_type_ids (peu important pour le MLM)*
  - **Labels -> copie de l'input_ids**
  - **Masking**
  - **Calcul de la fonction de perte**



C- Transformer BERT

Analyse de sentiment: Fine tuning sur machine locale avec GPU

Bidirectional Encoder Representations from Transformers: Validation Accuracy autour de 0,81

- Pourquoi utiliser distilbert?
  - **Modèle Transformer petit, rapide, bon marché et leger**
  - **40% de paramètres en moins que le *bert-base-uncased*,**
  - **60% plus rapide en ne perdant que 3% de performance**

Auto tokenization
- *Longueur max des séquences correspondant au quantile 95% du nombre de mots par doc pour tout le corpus*
- *Padding*
- *Masque d'attention*

Split Train/Test 80/20
- *Corpus de sequences*
- *Matrice des masques d'attention*
- *Labels de sentiments*

Chargement du modèle distilbert préentrainé
- *TFAutoModelForSequenceClassification à 2 labels*

Compliation du modèle distilbert préentrainé
- *Avec callback permettant de sauver les meilleurs poids en terme de validation loss*
- *Loss=Sparse Categorical Crossentropy*
- *metric =.Sparse Categorical Accuracy('accuracy')*
- *optimizer =.Adam(learning_rate=2e-5,epsilon=1e-08)*

Fine tuning du modèle préentrainé
- *Avec les tweets preprocéssés*
- *4 epochs (pas besoin de plus car déjà pré entrainé)*

```
Model: "tf_distil_bert_for_sequence_classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 distilbert (TFDistilBertMai  multiple                 66362880
 nLayer)

 pre_classifier (Dense)      multiple                  590592

 classifier (Dense)          multiple                  1538

 dropout_19 (Dropout)        multiple                  0

=================================================================
Total params: 66,955,010
Trainable params: 66,955,010
Non-trainable params: 0
```

```
F1 score 0.8149879303187317
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.78      0.80    158066
           1       0.79      0.84      0.81    158067

    accuracy                           0.81    316133
   macro avg       0.81      0.81      0.81    316133
weighted avg       0.81      0.81      0.81    316133
```

C

ANALYSE DE SENTIMENT AVEC MICROSOFT AZURE

A- Utilisation de azure cognitives services

Une solution très facile à implémenter mais avec de nombreuses contraintes

- tarification:
  - https://azure.microsoft.com/fr-fr/pricing/details/cognitive-services/language-service/
  - gratuits 5000 textes (de 1000 caractères) / mois
  - 90 centimes par 1000 textes pour les premiers 500000
    - Très couteux-> 450€ pour 500000 textes de 1000 caracteres

- limitations:
  - les performances dépendent d'un certain nombre de facteurs tels que :
    - le domaine du sujet
    - les caractéristiques du texte traité
    - le cas d'utilisation du système
    - la façon dont les gens interprètent la sortie du système
    - Le modèle est entrainé sur les avis de produits et de services
    - aucune compréhension de l'importance relative des différentes phrases qui sont envoyées ensemble
    - difficultés à reconnaître le sarcasme.
    - le score de confiance ne reflète pas l'intensité du sentiment mais la confiance du modèle pour un sentiment particulier (positif, neutre, négatif).
    - limite de données: https://docs.microsoft.com/fr-fr/azure/cognitive-services/language-service/sentiment-opinion-mining/how-to/call-api
    - limite de taux de transfert

A- Utilisation de azure cognitives services

WORKFLOW

Création des services azure nécessaires:
Azure Key Vault (pour stocker les crédenciales)
Azure cognitives services (pour analyser les sentiments)

Création des variables environnementales
permettant de se connecter à l'azure key Vault

Extraction de 500 tweets à tester
De façon aléatoire
Avec des labels équilibrés

Connexion à l'azure key Vault
Grâce à des variables environnementales
Afin de récupérer les crédenciales du cognitives services

Récupération des scores de confiance pour chaque tweet via cognitives services:
3 scores de confiance (négatif-neutre-positif)

Analyse de l' accuracy

- **des scores pas exceptionnels sur le peu de tweets testé**
  - Accuracy avec prise en compte du neutral:

	precision	recall	f1-score	support
0	0.79	0.57	0.67	235
1	0.00	0.00	0.00	0
2	0.76	0.60	0.67	265
accuracy			0.59	500
macro avg	0.52	0.39	0.44	500
weighted avg	0.77	0.59	0.67	500

  - Accuracy sans prise en compte du neutral:

	precision	recall	f1-score	support
0	0.74	0.68	0.71	235
1	0.73	0.78	0.76	265
accuracy			0.73	500
macro avg	0.73	0.73	0.73	500
weighted avg	0.73	0.73	0.73	500

B-Test de azure ML designer:

Le concepteur de modèle en glisser-déplacer

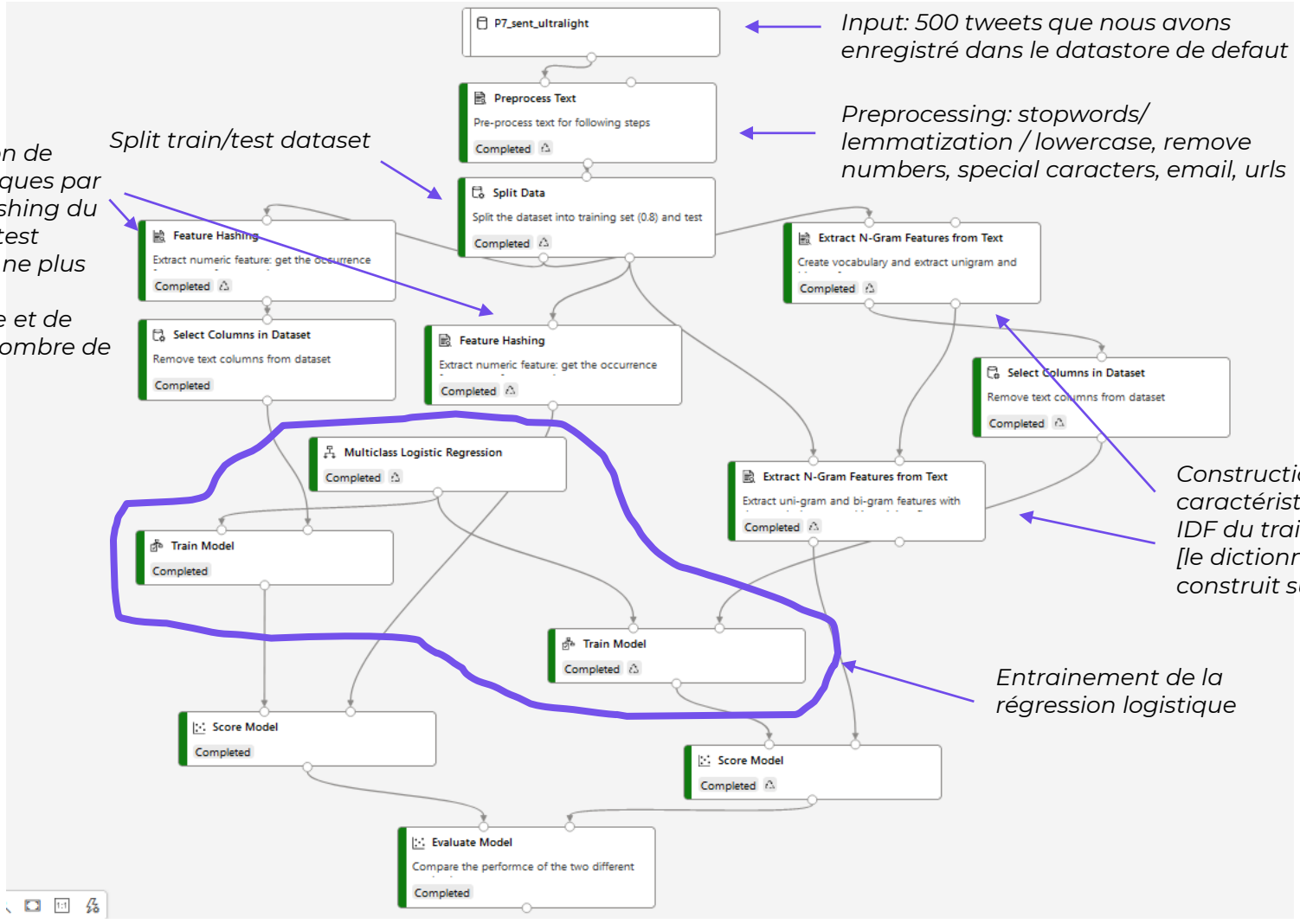
- Une interface de glisser-déplacer pour accélérer la création et le déploiement de modèles
  - Permet de se connecter à n'importe quelle source de données:
    - Stockage Blob Azure, Azure Data Lake Storage, Azure SQL, ou chargez des données à partir d'un fichier local
  - Permet de préparer et prétraiter des données à l'aide d'un large éventail de modules intégrés:
    - transformation des données et l'ingénierie de caractéristiques
  - Permet de créer et d'entrainer des modèles visuellement à l'aide des derniers algorithmes Machine Learning et Deep Learning:
    - notamment ceux pour la vision par ordinateur, l'analyse de texte, les recommandations et la détection d'anomalies.
  - Utilise des modules de glisser-déplacer pour valider et évaluer des modèles
    - Glissez-déplacez des modules pour des modèles **sans code** ou personnalisez à l'aide de code Python et R
    - Exécutez de façon interactive des pipelines Machine Learning. Effectuez une validation croisée des modèles et des jeux de données pour plus de précision. Accédez aux visualisations de données pour évaluer les modèles en quelques clics
  - Permet de déployer et publier des points de terminaison d'inférence en temps réel ou par lots en quelques clics
    - Générez automatiquement des fichiers de scoring et l'image de déploiement

B-Test de azure ML designer:
Le concepteur de modèle en glisser-déplacer

- **Forces**
  - Peu ou pas de code: bien pour les personnes non techniques
  - Facile à déployer en production dans l' environnement du cloud azure
- **Faiblesses**
  - Peu d'algo intégrés : modèles / transformations
    - On peut coder ses propres modèles mais on perd l aspect no code et donc l'utilité du service
  - Prix de l'environnement azure
  - Faible capacité de storage dans la version gratuite

B-Test de azure ML designer:

Adaptation de l'exemple en ligne à notre problématique



B-Test de azure ML designer:

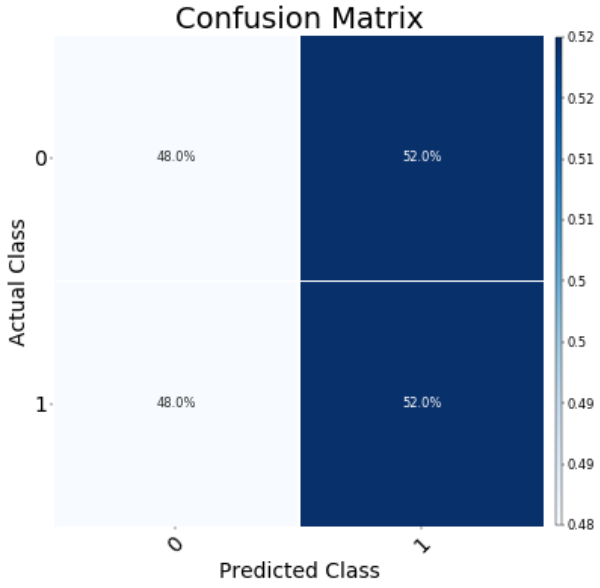
Résultat après avoir lancé une instance de calcul

Metriques de résultats pour les 2 modèles: 1ere ligne feature hashing / 2eme ligne tf-idf

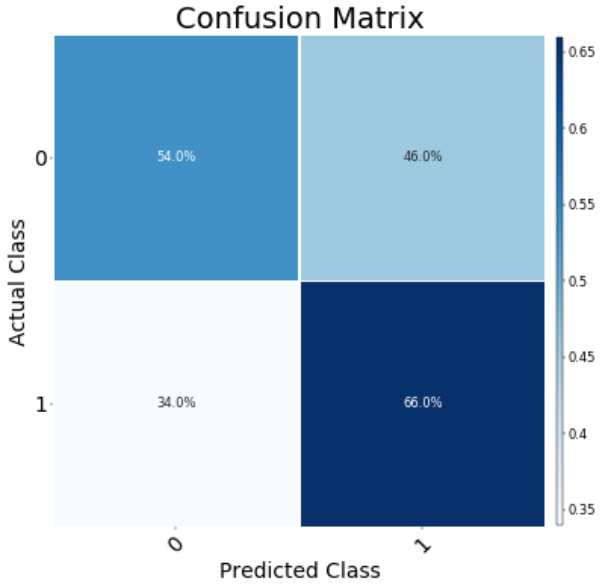
Overall_Accuracy	Micro_Precision	Macro_Precision	Micro_Recall	Macro_Recall
0.5	0.5	0.5	0.5	0.5
0.6	0.6	0.601461	0.6	0.6

Rappel: Le test se fait sur trop peu de données pour en tirer des conclusions

Matrice de confusion pour la regression logistique à feature hashing



Matrice de confusion pour la regression logistique à tf-idf



C-Entrainement du modele LSTM avec AZURE:

WORKFLOW de la préparation des données

Connexion à L'azure workspace

Récupération du datastore
de défaut lié au workspace

Upload des fichiers vers le datastore
Sentiment et glove50d

Passage en tabular dataset
du fichier Sentiment
Passage en file dataset
du fichier we glove50d

Registration des datasets

Preprocessing avec split train/test
Voir partie A pour le détail

Upload des binaries vers le datastore
Xtrain/Xtest/ytrain/ytest/glove50d weight matrix

Registration des binaries datasets

WORKFLOW de la préparation et du run de l'expérience d'entrainement

Création d'une cible de calcul
(vm_size='Standard_DS3_v2')

Création d'un training folder en local avec download des binaries necessaires à l'entrainement
Xtrain/Xtest/ytrain/ytest/glove50d weight matrix

Ecriture du training script
définissant le modèle LSTM à entrainer ainsi que ses entrées/sorties
Sauvegarde du training script dans le training folder

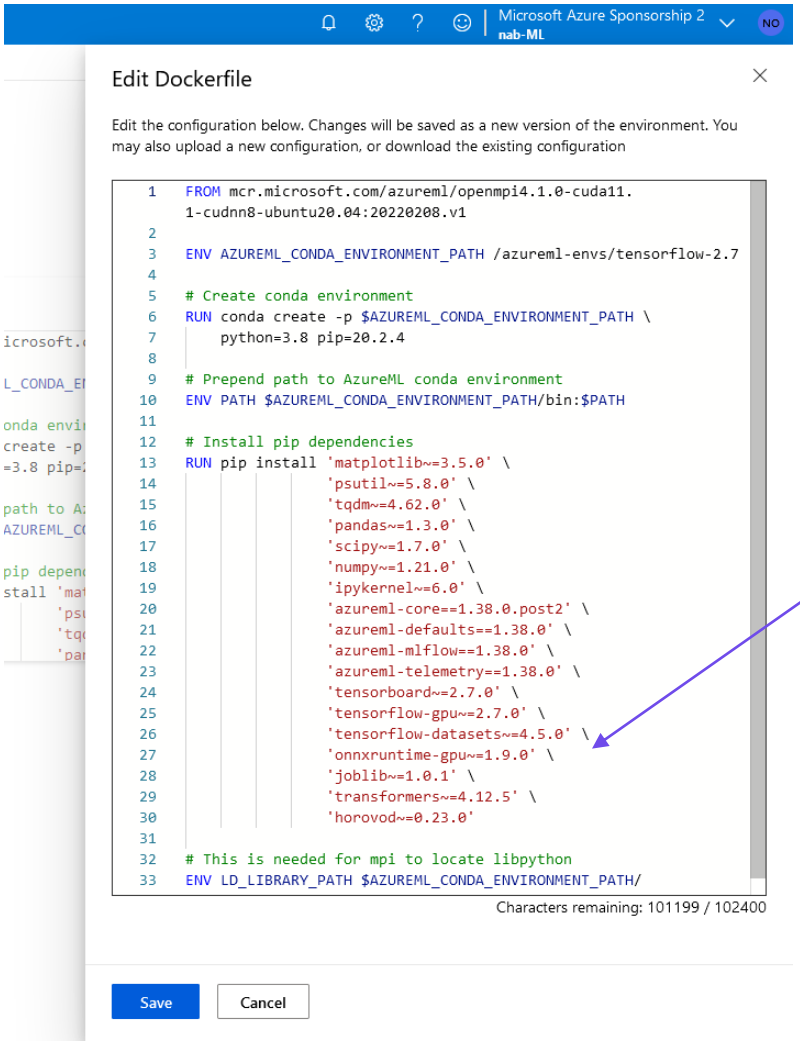
Definition de l'environnement de calcul:
Veiller à modifier manuellement un dockerfile de l'environnement désiré afin de lui ajouter les dépendances nécéssaires

Definition du script config
Source directory / environment / compute instance / training script

Run de l'expérience d'entrainement

C-Entrainement du modèle LSTM avec AZURE

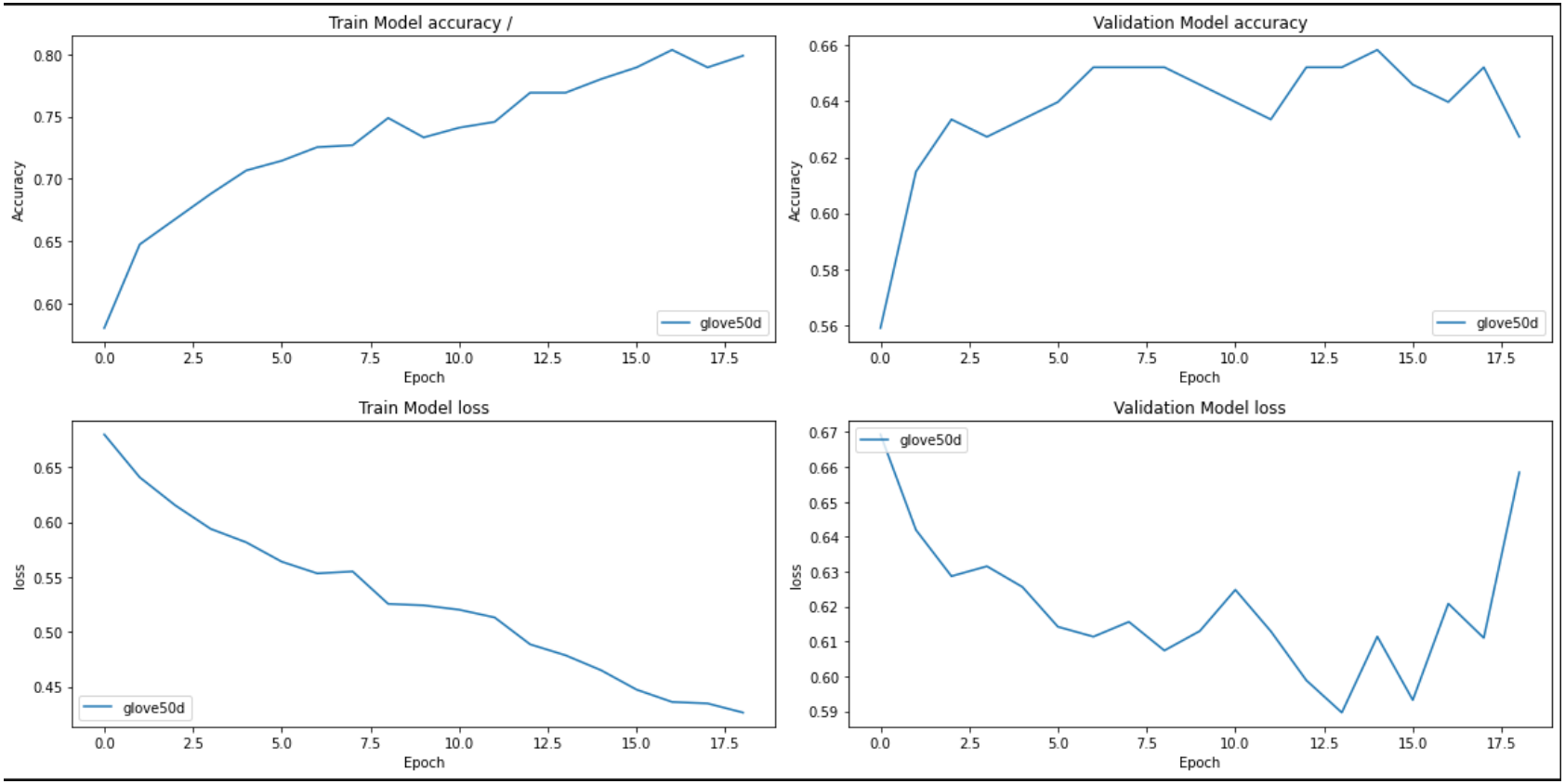
Ajout de dépendances au dockerfile de l' environnement ciblé



Joblib et transformers par exemple

C-Entrainement du modèle LSTM avec AZURE:

- **Par soucis de cout , l'entrainement du LSTM sur Azure ne s'est fait que sur moins d'un milliers de tweets, ce qui explique la faible accuracy de l'échantillon de validation**



D-Deploiement du modèle distilbert fine tuné

WORKFLOW de la préparation au déploiement

Chargement du modèle pré entrainé distilbert à partir du hub hugginfaces
TFAutoModelForSequenceClassification

Compilation du modèle pré entrainé
loss=SparseCategoricalCrossentropy
metric = SparseCategoricalAccuracy('accuracy')
optimizer = Adam(learning_rate=2e-5,epsilon=1e-08)

Chargement des poids issus du fine tuning déjà réalisé en local

Sauvegarde de l'autotokenizer correspondant et du modèle pré entrainé et fine tuné
en local grâce à la fonction de sauvegarde de la librairie transformers
save_pretrained('./test2/', push_to_hub=False)

Registration du modèle dans notre workspace azure
par passage du répertoire local de sauvegarde

Création du folder de déploiement en local

Ecriture du scoring script dans le folder de deploiement
Création des fonctions init et run

WORKFLOW de déploiement

Définition de l'environnement de déploiement et paramétrage pour l'inférence
- *on peut récupérer un environnement d'entrainement avec les bonnes dépendances*
- ***Ajouter le paramétrage d'inférence pour un environnement d'entrainement**: env.inferencing_stack_version = "latest"*

Definition de l'inférence_config
deployment directory / environment / scoring script

Définition de la deployment_config
(AciWebservice.deploy_configuration(cpu_cores=1, memory_gb=4)
suffit largement pour des petis modèles Interieurs à 1Go

Déploiement et test : *Attente du statut healthy*

```
Deploying model...
Tips: You can try get_logs(): https://aka.ms/debugimage#dockerlog or local deployment:
https://aka.ms/debugimage#debug-locally to debug if deployment takes longer than 10 min
utes.
Running
2022-02-26 15:44:22+01:00 Creating Container Registry if not exists.
2022-02-26 15:44:22+01:00 Registering the environment.
2022-02-26 15:44:24+01:00 Use the existing image.
2022-02-26 15:44:25+01:00 Generating deployment configuration.
2022-02-26 15:44:26+01:00 Submitting deployment to compute.
2022-02-26 15:44:31+01:00 Checking the status of deployment distilbert-sentiment-ft..
2022-02-26 15:52:16+01:00 Checking the status of inference endpoint distilbert-sentimen
t-ft.
Succeeded
ACI service creation operation finished, operation "Succeeded"
Healthy
```

17- test du deploiement

```
# Call the web service
service.run("we love you")
```

```
[{'label': 'LABEL_1', 'score': 0.8176222443580627}]
```

D RÉSUMÉ

résumé

	accuracy	*training speed*	*inference speed*	*training cost*	*inference cost*	*implémentation*	*deploiement*
cognitive services	0,6-0,73 (dépend proximité avec les données d'entrainement MS)		**relativement lent**		**elevé**	**facile**	**facile**
Modèle simple via Azure Machine Learning designer	0,5-0,6 (limité en ressources donc à voir sur un plus gros échantillon d'entrainement)	**lent**	**relativement lent**	**de moyen à élevé en fonciton des ressources**	**de moyen à élevé en fonciton des ressources**	**facile, demande des compétences en datascience mais pas forcément en code**	**facile**
LSTM unidirectionnel en local 16G/GPU 8G	0,75	~200sec / epoch pour tous les tweets (1M6) mais sans regularisation dropout- compter entre 5 et 10 epochs	**relativement rapide, quelques secondes pour un échantillon de test conséquent**	**faible**	**faible**	**demande de la technicité**	
LSTM unidirectionnel sur le cloud (limité par mes crédits)	on a atteint 0,66 mais en theorie en exploitant plus de données , on devrait obtenir le meme resultat que précédemment	**dépend des ressources selectionnés**				**demande de la technicité**	**demande une montée en compétence**
Distilbert en local GPU	**0,81**	le fine tuning sur une grosse partie des tweets dure 1h30 par epochs (4 suffisent sachant qu ele modèle est déjà préentrainé): donc 6h au total pour ma part	**relativement rapide**	**faible pour le fine tuning, le pre training n est pas à notre charge**	**faible**	**demande de la technicité**	
Distilbert sur le cloud	pas testé (limité par les ressources mises à dispo)	**dépend des ressources selectionnés**				**demande de la technicité**	**demande une montée en compétence**