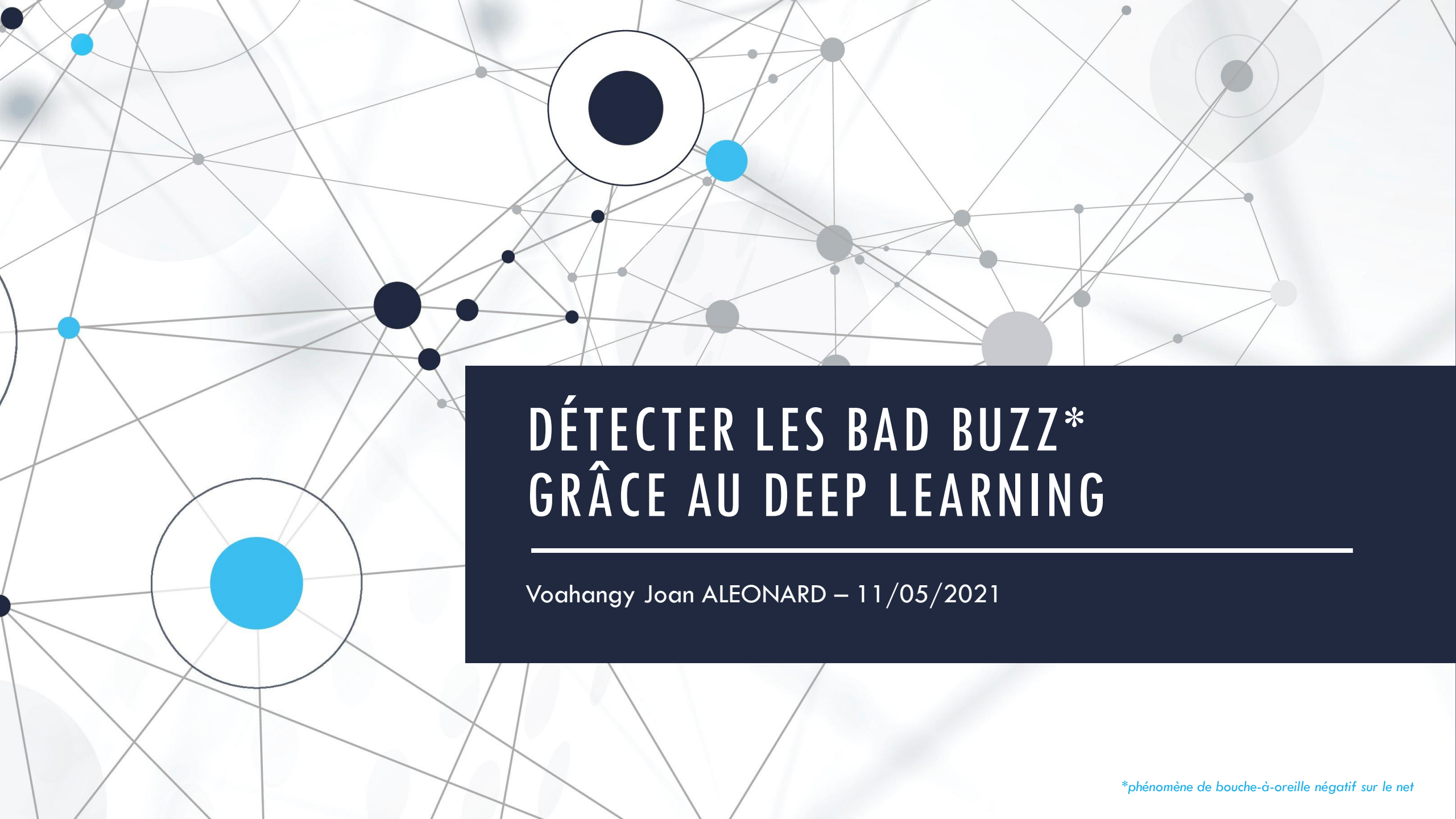


DÉTECTER LES BAD BUZZ* GRÂCE AU DEEP LEARNING

Voahangy Joan ALEONARD – 11/05/2021

*phénomène de bouche-à-oreille négatif sur le net

AGENDA DU JOUR

PROJET ET DÉMARCHE

PRÉ-TRAITEMENT DES DONNÉES

PRÉSENTATION DES APPROCHES

CHOIX MODÈLE ET DÉPLOIEMENT

PROJET ET DÉMARCHE

CONTEXTE PROJET ET DÉMARCHE

Problématique

Surveiller la réputation sur les réseaux sociaux

Prédire le sentiment associé à un tweet

Livrer un prototype fonctionnel du modèle

Source de travail

Utiliser des données open source d'analyse de sentiment

Trouver des tweets annotés du sentiment exprimé (👍 ou 👎)

Jeu de données : Sentiment140

Démarche

Analyser et prétraiter les données

Explorer les différentes approches

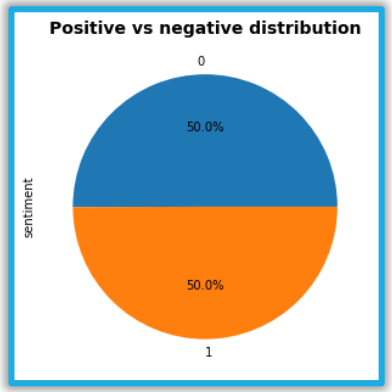
Déployer le meilleur modèle

PRÉ-TRAITEMENT DES DONNÉES

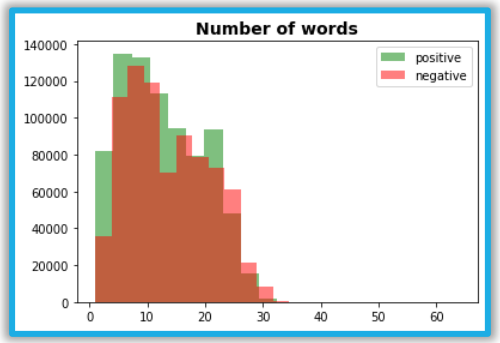
EXPLORATION DES DONNÉES

1,6 millions de Tweets collectés sur 3 mois: avril-juin 2009

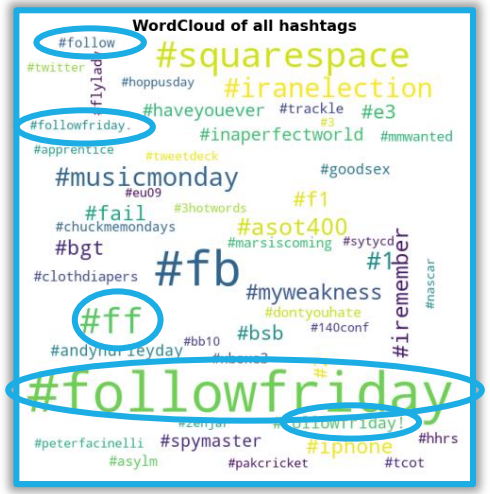
Classes cible équilibrées



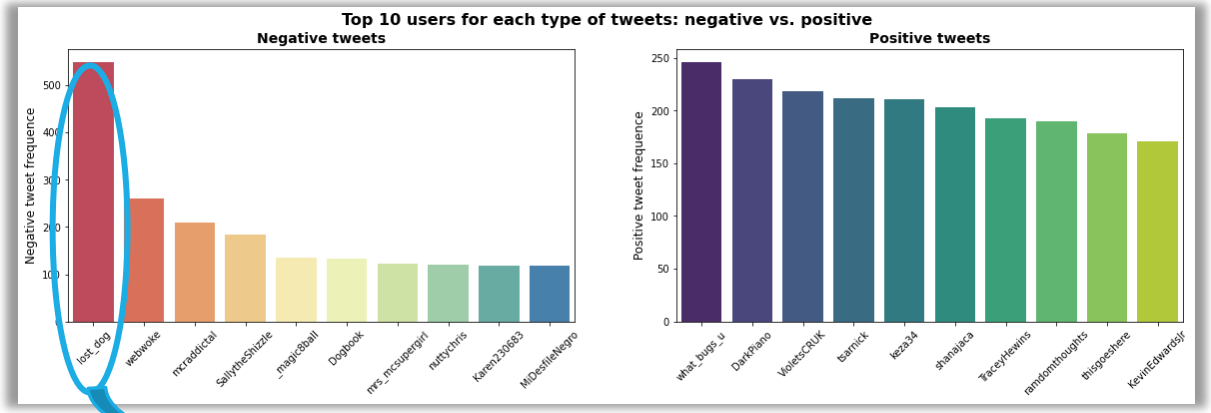
13/14 mots en moyenne



Hashtags peu informatifs (top 50)



Identification d'utilisateur étrange

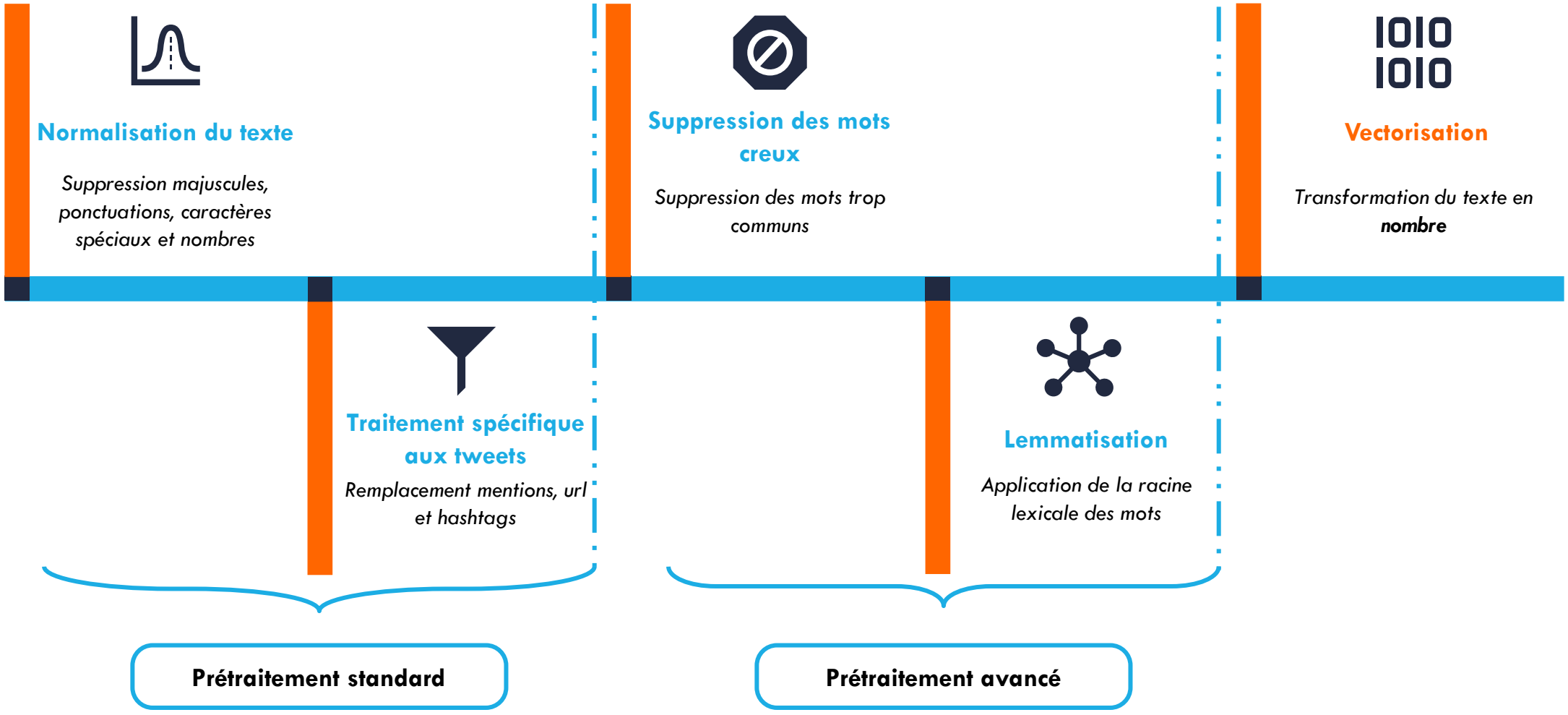


N°1 NEGATIVE MOOD
(549, 3)

	sentiment	user	tweet
43935	0	lost_dog	@NyleW I am lost. Please help me find a good home.
45574	0	lost_dog	@SallyD I am lost. Please help me find a good home.
46919	0	lost_dog	@zuppaholic I am lost. Please help me find a good home.
47949	0	lost_dog	@LOSTPETUSA I am lost. Please help me find a good home.
50572	0	lost_dog	@JeanLevertHood I am lost. Please help me find a good home.

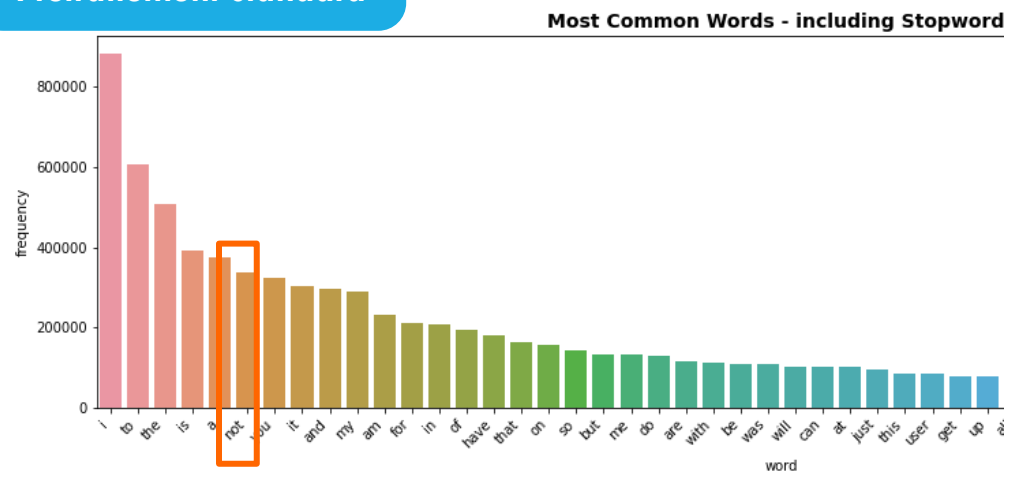
APPLICATION DE 2 TYPES DE PRÉTRAITEMENT

Nettoyage préliminaire : Suppression des doublons sur le sous-ensemble [Utilisateur – Tweet] et des lignes vides



COMPARAISON DES PRÉTRAITEMENTS

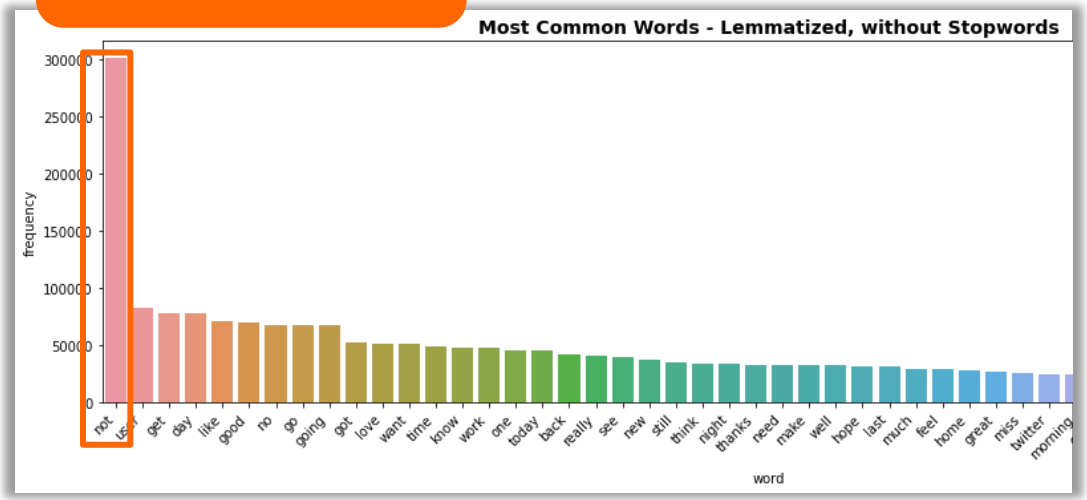
Prétraitement standard



'user nope they did not have it'

'user no it is not behaving at all i am mad why am i here because i can not see you all over there'

Prétraitement avancé



user nope not

'user no not behaving mad not see'

Utilisation des données avec Prétraitement standard : garder la séquence des mots, qui est importante dans l'analyse de sentiment

PRÉSENTATION DES APPROCHES

3 APPROCHES DIFFÉRENTES

Azure Cognitive Services

Modèle « Prêt-à-l'emploi »

AZURE Machine Learning

Bibliothèque de modules pour construire des modèles

TensorFlow

Modèles « propriétaire » de réseaux de neurones profonds

Choix de taille d'échantillon sur les tweets pré-traités pour un prototypage rapide

Entraînement et prédiction sur environ 10min par modèle

1.000 tweets

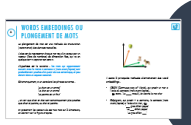
30.000 tweets

400.000 tweets

**Vectorisation si applicable*

Hachage des caractéristiques

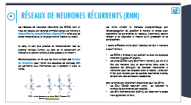
Extraction des caractéristiques n-grams



Word Embeddings (plongement de mots)

**Entraînement si applicable*

Régression logistique



Réseaux de neurones récurrents

Prédiction

Evaluation des 3 approches + Déploiement du meilleur modèle

MODÈLE CLÉ-EN-MAIN : API SENTIMENT ANALYSIS DE MICROSOFT AZURE

Modèle « Prêt-à-l'emploi »

Azure Cognitive Services

Prérequis:

Ouverture d'un compte Azure (sur le Portail)

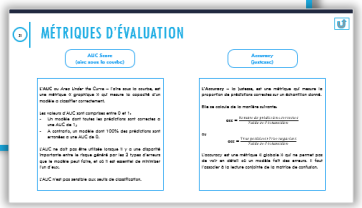
Souscription au service **Text Analytics** de la collection Azure Cognitive Services

Récupération du point de terminaison et de la clé de souscription pour utiliser le service

Utilisation du Kit de Développement Logiciel (SDK) Azure Machine Learning pour Python pour appeler le service

Pour valider notre hypothèse sur le prétraitement des données, nous avons testé l'API (Interface de programmation d'application avec les données :

- **Brutes,** sans aucun traitement;
- **Prétraitées,** standard;
- **Prétraitées,** avancé.

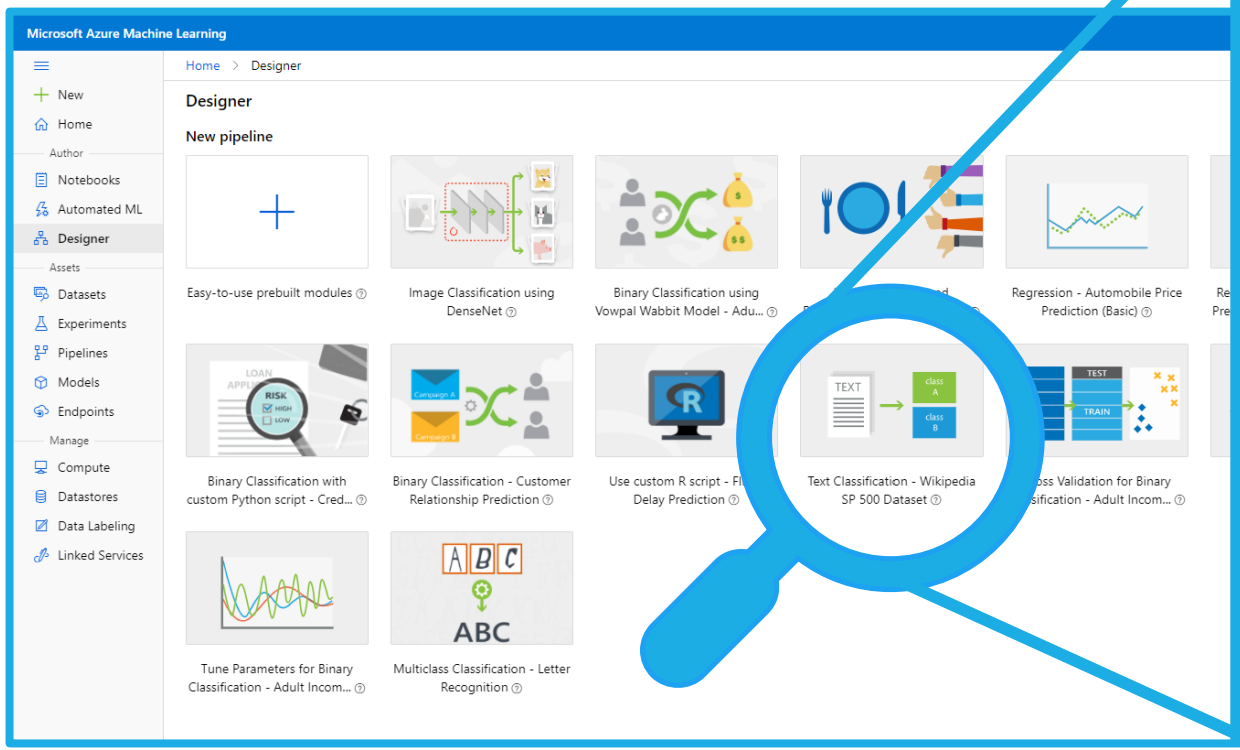


Model	Predict_time	AUC_Score	Accuracy
Original tweets	212.5	71.081%	70.860%
Cleaned tweets	197.2	76.187%	76.148%
Lemmatized tweets	198.2	74.595%	74.633%

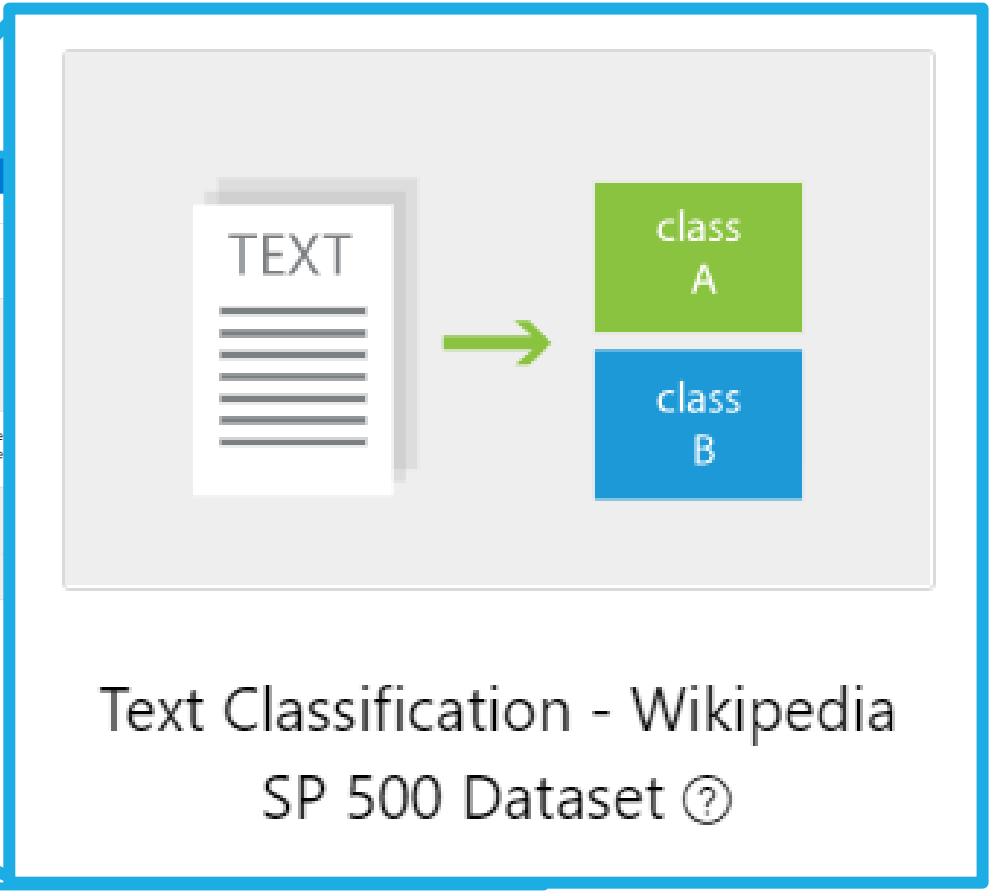
MODÈLE BOÎTE-À-OUTILS : AZURE MACHINE LEARNING STUDIO (AMLS)

AZURE Machine Learning

Prérequis:
Ouverture d'un compte Azure (sur le Portail)
Souscription d'un espace de travail **Azure Machine Learning**

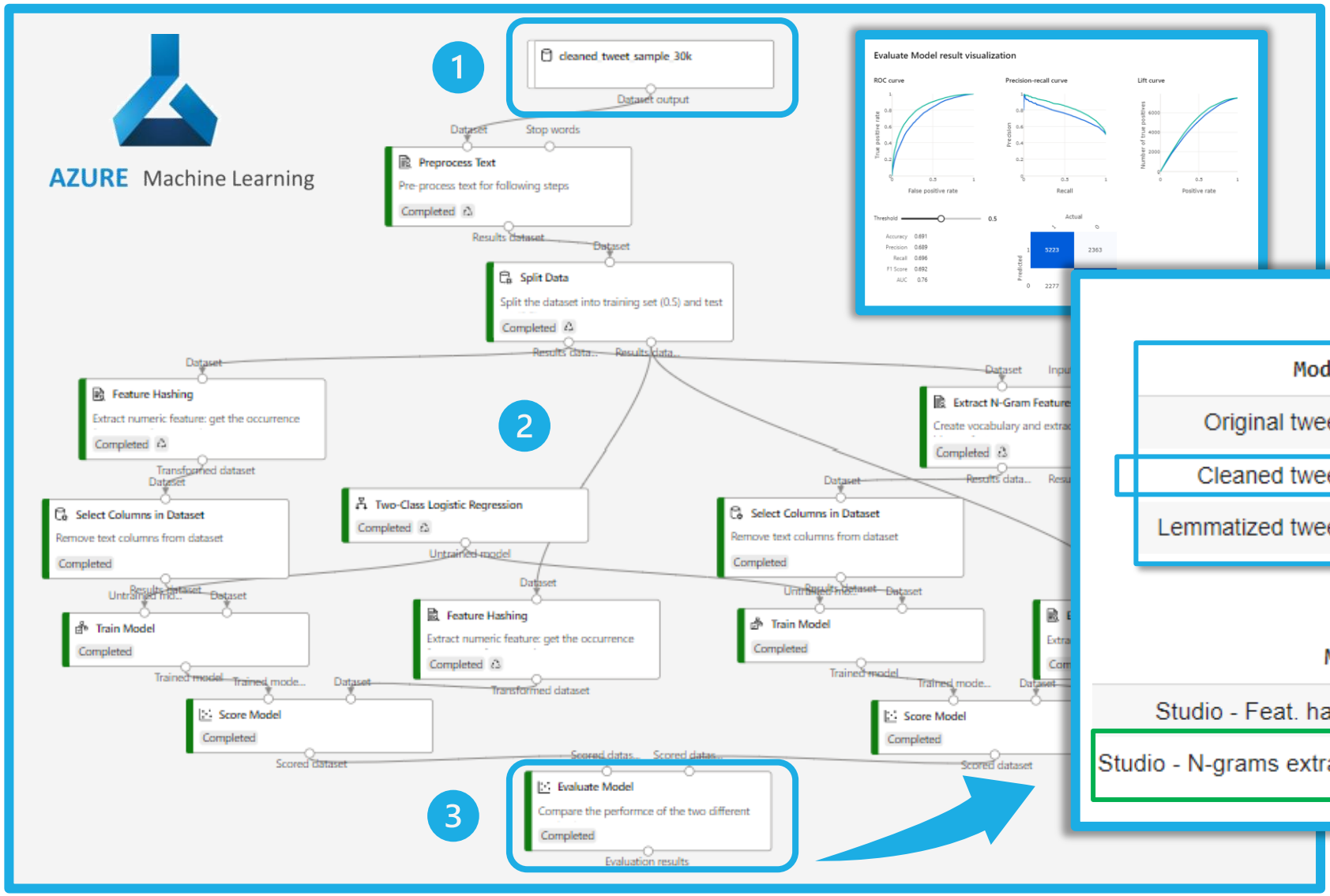


Clonage d'un pipeline existant



AMLS : CONCEPTEUR ET PIPELINE DE MODULES

Bibliothèque de modules



1. Remplacement du jeu de données
2. Correction des paramètres des modules
3. Exécution et évaluation des modèles

Approche 1

Model	Predict_time	AUC_Score	Accuracy
Original tweets	212.5	71.081%	70.860%
Cleaned tweets	197.2	76.187%	76.148%
Lemmatized tweets	198.2	74.595%	74.633%

Approche 2

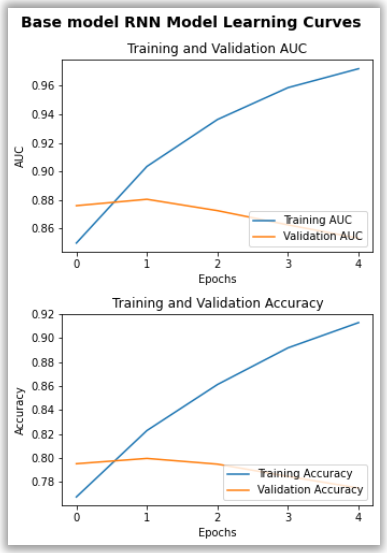
Model	Predict_time	AUC_Score	Accuracy
Studio - Feat. hashing	81.5	76.900%	70.000%
Studio - N-grams extraction	103.4	84.200%	76.300%

MODÈLE AVANCÉ: RÉSEAUX DE NEURONES PROFONDS

TensorFlow

Optimisation du modèle de base (*baseline*)

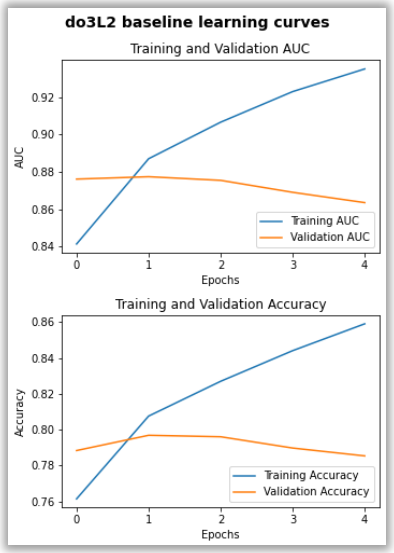
```python
# Add sequential model
base_model = Sequential()
base_model.add(Embedding(input_dim=VOCAB_SIZE,
                         output_dim=EMBEDDING_DIM,
                         input_length=MAX_LEN))
base_model.add(SimpleRNN(128, dropout=0.3))
base_model.add(Dense(64, activation='relu'))
base_model.add(Dense(1, activation='sigmoid'))
base_model.summary()
```



	Model	AUC_Score	Accuracy	Loss
0	SimpleRNN	85.245085	77.525002	56.828576

```python
# Add sequential model
do3l2_model = Sequential()
do3l2_model.add(Embedding(input_
                          output_
                          input_length=MAX_LEN))
do3l2_model.add(SpatialDropout1D(0.3))
do3l2_model.add(SimpleRNN(128, dropout=0.3))
do3l2_model.add(Dropout(0.3))
do3l2_model.add(Dense(64, activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)))
do3l2_model.add(Dropout(0.3))
do3l2_model.add(Dense(1, activation='sigmoid'))
do3l2_model.summary()
```

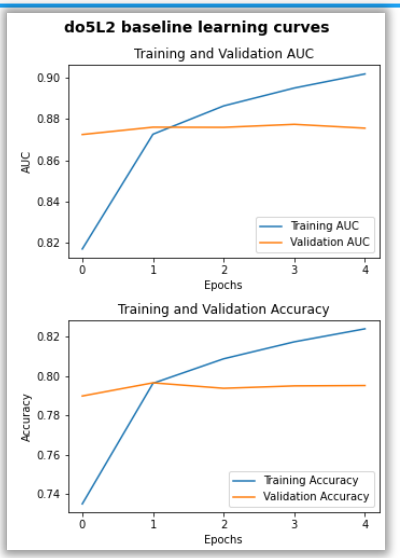
Ajout de Dropout (0,3) et Regularizers (L2)



	Model	AUC_Score	Accuracy	Loss
0	do3L2 SimpleRNN	86.35236	78.49375	49.211758

```python
# Add sequential model
do5l2_model = Sequential()
do5l2_model.add(Embedding(input_
                          output_
                          input_length=MAX_LEN))
do5l2_model.add(SpatialDropout1D(0.5))
do5l2_model.add(SimpleRNN(128, dropout=0.5))
do5l2_model.add(Dropout(0.5))
do5l2_model.add(Dense(64, activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)))
do5l2_model.add(Dropout(0.5))
do5l2_model.add(Dense(1, activation='sigmoid'))
do5l2_model.summary()
```

Optimisation du Dropout à 0,5



	Model	AUC_Score	Accuracy	Loss
0	do5L2 SimpleRNN	87.459093	79.468751	44.949293

MODÈLE AVANCÉ: DEEP LEARNING & PLONGEMENT DE MOTS

Modèle avancé : RNN & Embeddings

TensorFlow

Paramètres communs

Configure all modeling variables

VOCAB_SIZE:	50000
MAX_LEN:	36
EMBEDDING_DIM:	256
DROP_OUT:	0.5
OPTIM_LR:	0.001
REGUL_LR:	0.001
NUM_EPOCHS:	5
BATCH_SIZE:	250

Modèle de base qu'on veut encore améliorer

```
Model: "base_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 36, 256)           12800000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 36, 256)           0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 128)               49280
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 12,857,601
Trainable params: 12,857,601
Non-trainable params: 0
_________________________________________________________________
```

Meilleure mémoire du passé proche

```
lstm (LSTM)                  (None, 128)               197120
```

Lecture gauche/droite et droite/gauche

```
bidirectional (Bidirectional (None, 256)               394240
```

Utilisation du modèle LSTM avec les Embeddings pré-entraînés

Word2Vec

```
embedding_3 (Embedding)      (None, 36, 300)           28044000
```

GloVe

```
embedding_4 (Embedding)      (None, 36, 200)           18696000
```

USE

```
keras_layer (KerasLayer)     (None, 512)               256797824
```

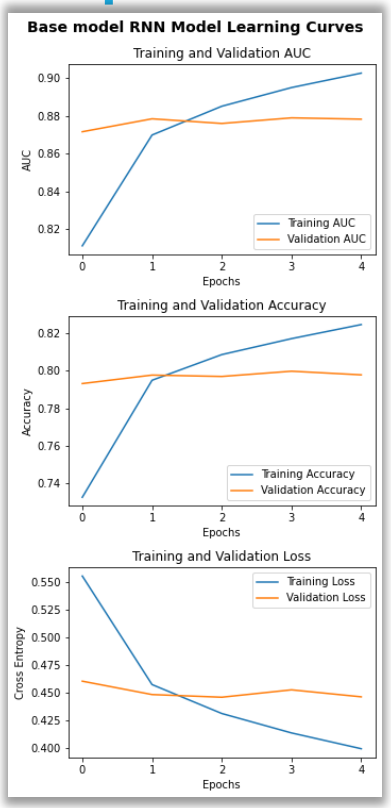
Transforme une phrase entière en vecteurs, n'utilise pas de LSTM

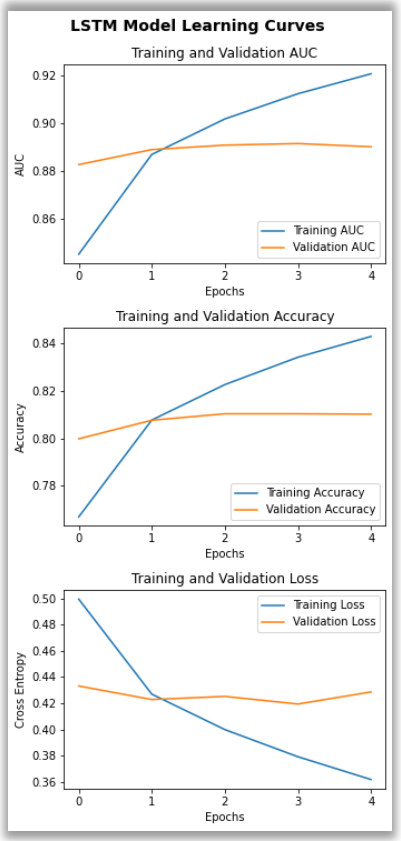
Différents réseaux de neurones : RNN → LSTM → LSTM bidirectionnel

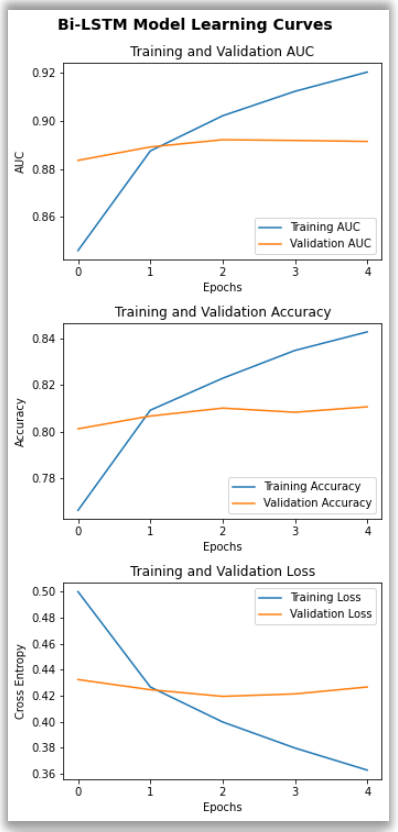
Différents embeddings : from scratch → pré-entrainés (Word2Vec, GloVe, USE…)

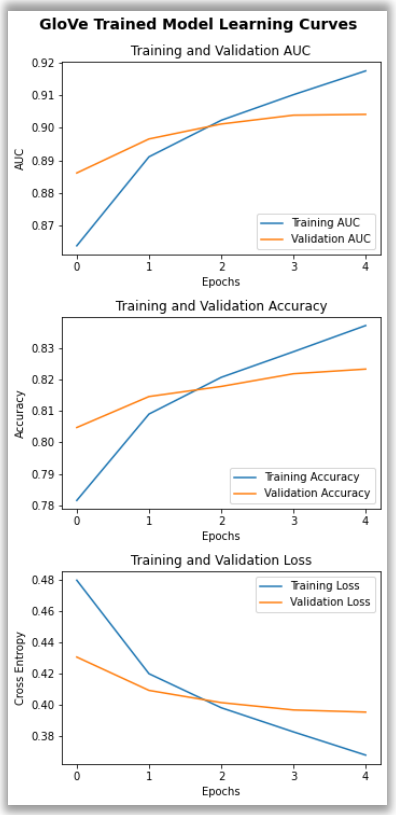
PERFORMANCE DES MODÈLES AVANCÉS

TensorFlow

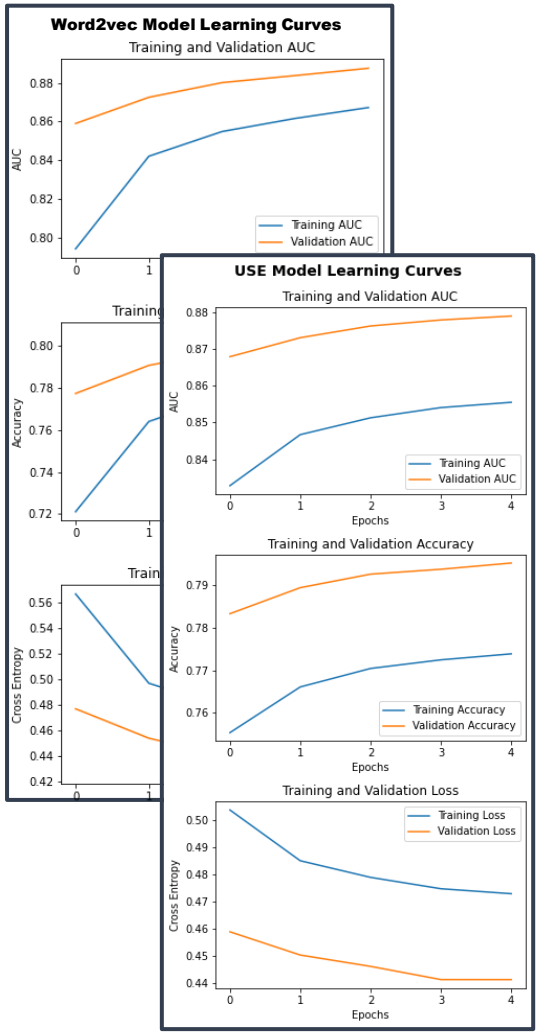


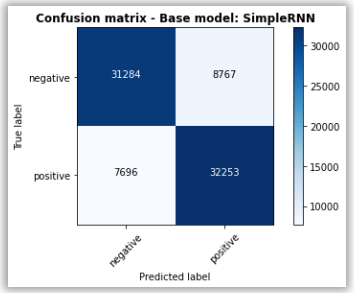


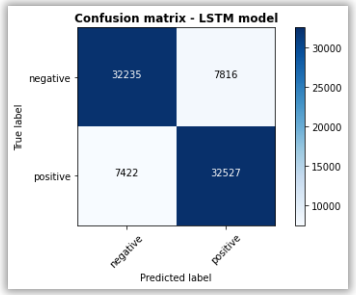


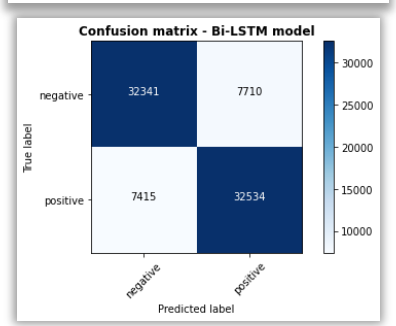


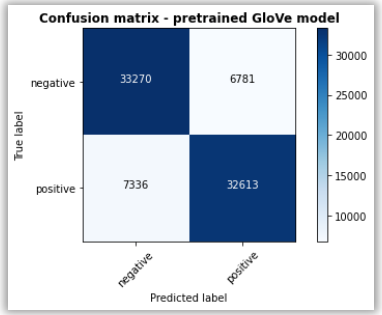
Word2Vec et USE en sous-apprentissage (underfit)











CHOIX MODÈLE ET DÉPLOIEMENT

SYNTHÈSE DES PERFORMANCES DES APPROCHES

Vectorisation	Modèle	Machine	Dimension de vecteurs	Echantillon	Paramètres entraînables	Temps d'exécution	AUC Score	Accuracy
N.A.	API Sentiment	CPU	N.A.	1.000	N.A.	3,28 min	76,187 %	76,148 %
Hachage de caractéristiques	Régression logistique	vCPU	N.A.	30.000	N.A.	~8 min	76,900 %	70,000 %
Extraction de caract. n-grams	Régression logistique	vCPU	N.A.	30.000	N.A.	~7,30 min	84,200 %	76,300 %
Embeddings propriétaire (Keras)	RNN Simple	GPU	256	400.000	12.857.601	8,10 min	87,653 %	79,421 %
Embeddings propriétaire (Keras)	LSTM	GPU	256	400.000	13.005.441	7,41 min	88,955 %	80, 953%
Embeddings propriétaire (Keras)	LSTM bidirectionnel	GPU	256	400.000	13.210.753	7,23 min	89,091 %	81,094 %
Embeddings pré-entraînés: Word2Vec	LSTM	GPU	300	400.000	227.969	1,20 min	88,685 %	80,586 %
Embeddings pré-entraînés: GloVe	LSTM	GPU	200	400.000	176.769	1,04 min	90,395 %	82,354 %
Embeddings pré-entraînés: USE	N.A.	GPU	512	400.000	131,585	4,16 min	87,912 %	79,594%

EXEMPLES D'ERREURS DU MEILLEUR MODÈLE : TWEETS LABELISÉS « NÉGATIF » PRÉDITS « POSITIF »

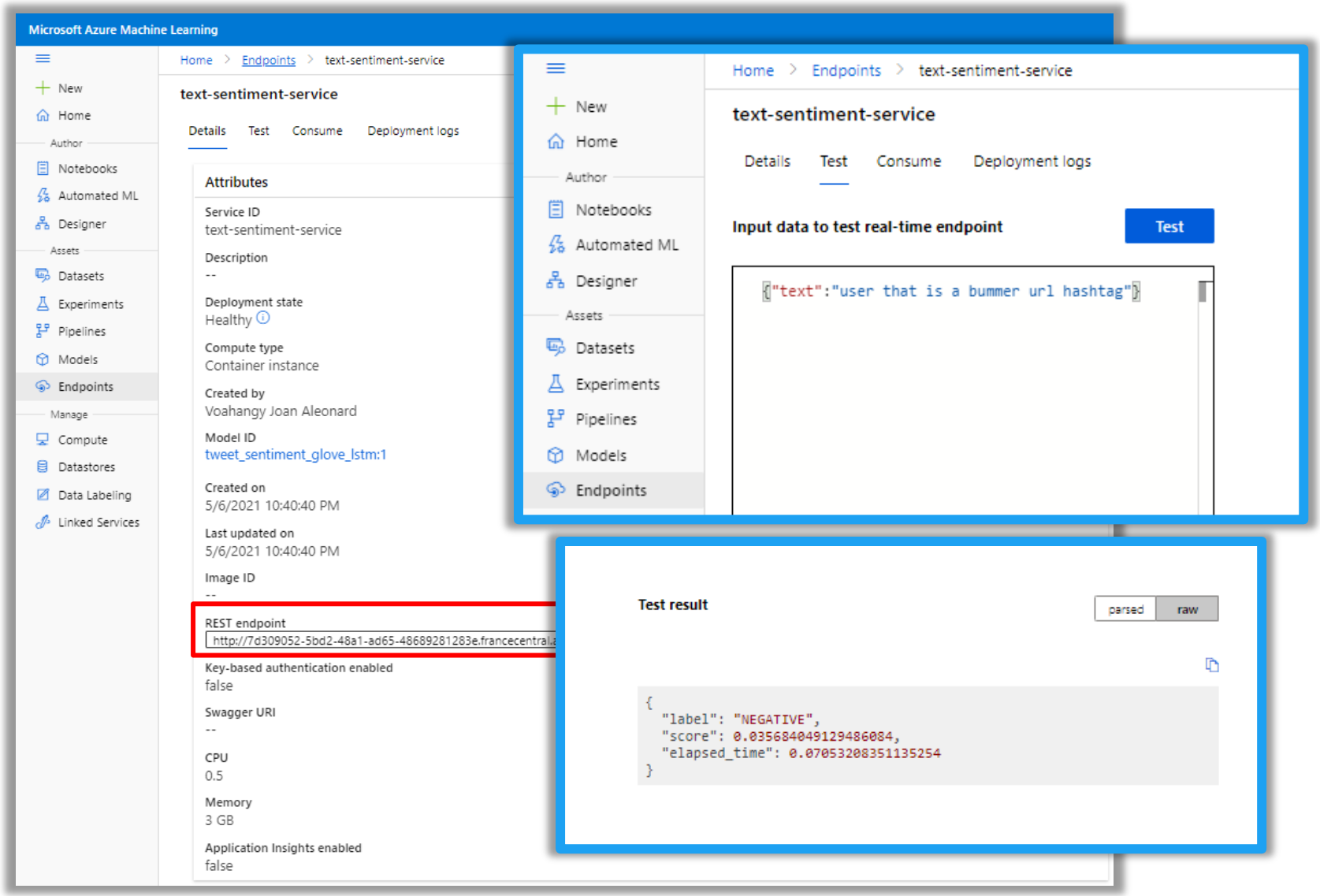
Housemate has told me to shut up with the hysterical laughing.	housemate has told me to shut up with the hysterical laughing	Erreur réelle du modèle
Holiday Was FANTASTIC	holiday was fantastic	Erreur de label à l'origine? Ce tweet semble très positif
Sugarsnap peas, carrots and califlower. what a lunch!	sugarsnap peas carrots and califlower what a lunch	Détection de sarcasme compliquée
kind of very pissed off that my parentals are going to NYC in Septmember without me. but now I just figured out what they can buy me!	kind of very pissed off that my parentals are going to nyc in septmember without me but now i just figured out what they can buy me	Mix d'émotions: joie, peine…
Rona Howarda? S tim to jde z kopce. Kdyz vezmu Beautiful Mind ... Sifra ... Andele. Od deviti k peti	user rona howarda s tim to jde z kopce kdyz vezmu beautiful mind sifra andele od deviti k peti	Tweets qui sont dans une langue autre que l'Anglais (ici, le tchèque)

INTERPRETATION

DÉPLOIEMENT SUR AZURE MACHINE LEARNING

Flux de travail

- Installation de Azure SDK pour Python
- Inscription du modèle sur Azure
- Préparation d'un script pour initier le déploiement et prédire avec le modèle
- Définition des configurations: environnement, dépendances, inférence
- Exécution du déploiement
- Utilisation du service web déployé



SYNTHÈSE

SYNTHÈSE DES INVESTIGATIONS

Flux d'un projet IA (vision 360°)

Les promesses du Cloud

- Facilité de création de flux de travail en Machine Learning
- Mise à l'échelle (scale up / down) des services en fonction des besoins
- Disponibilité permanente des services
- Affranchissement des coûts élevés de machines
- Paiement à l'usage
- Accès aux dernières mises à jour, innovations et technologies

Focus sur les objectifs

- **Vision d'ensemble** du projet de Machine Learning : place du projet dans la stratégie globale, usage, délai de mise en marché, …
- Prototypage en local pour contrôler la **faisabilité** du projet
- Analyse de toutes les **options** disponibles
- Vérification de la capacité (interner) à **implémenter et à manager** les services dans le cloud

Approche 1

Outil clé en main, idéal pour commencer (petit)

Approche 2

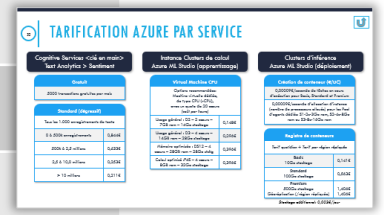
Utilisation de services managés afin de profiter de l'existence d'algorithmes déjà éprouvés

Approche 3

Utilisation de modèles propriétaires avancés à déployer dans le Cloud

PROPOSITION DE CRITÈRES DE SÉLECTION

	Approche « Clé en main »	Approche « Boîte à outils »	Approche « propriétaire »
Connaissance en IA	Peu de connaissance	Connaissances de base	Expertise
Prise en main	Simple	Relativement simple	Complexe
Adaptabilité	Aucune	Possible	Complète
Local / Cloud	Non / Oui	Non / Oui	Oui / Oui
Performance	Moyen	Bon	Très bon
Investissement matériel	Inclus	Inclus	(à définir)*
Investissement temps	Faible	Modéré	Conséquent
Coûts Cloud	Usage en volumétrie	Usage horaire	(à définir)*



**Le coût d'un GPU est moindre par rapport à un vrai coût matériel*

CONCLUSION

Un **choix d'approche** dépendant principalement des besoins exprimés par les équipes en interne ET des ressources disponibles (humain, financier, temps).

Une **performance du modèle** dépendant principalement :
- La qualité initiale des données ;
- Du prétraitement effectué.

QUESTIONS / RÉPONSES

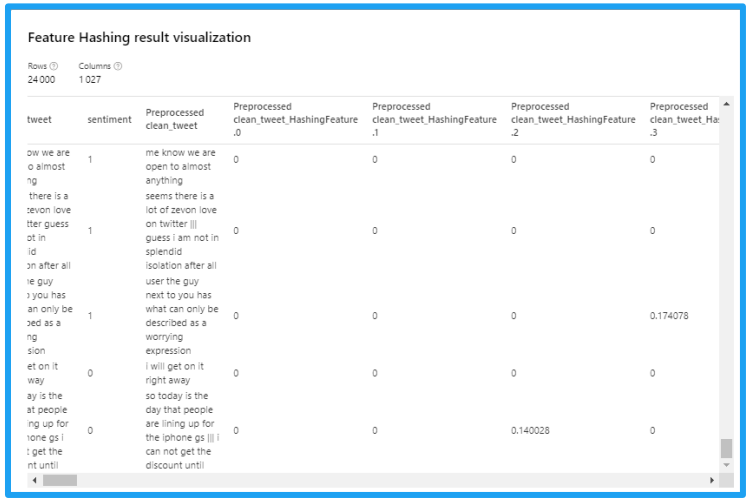
ANNEXES

VECTORISATION DANS AZURE MACHINE LEARNING STUDIO (AMLS)

La **vectorisation** est le fait de transformer du texte en nombre pour qu'il puisse être ingérer par un modèle de Machine Learning.

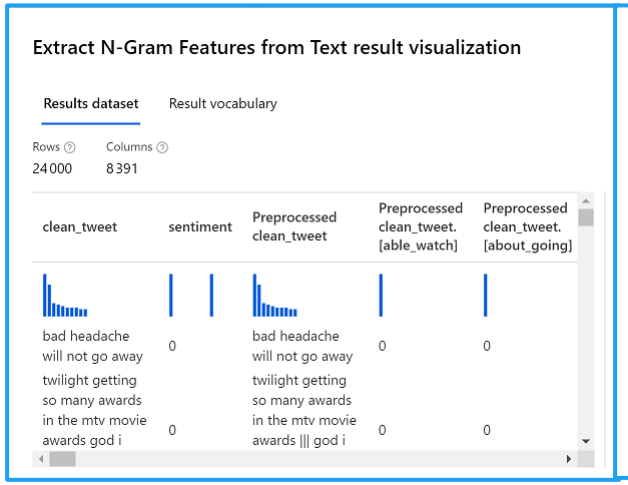
Hachage de caractéristiques

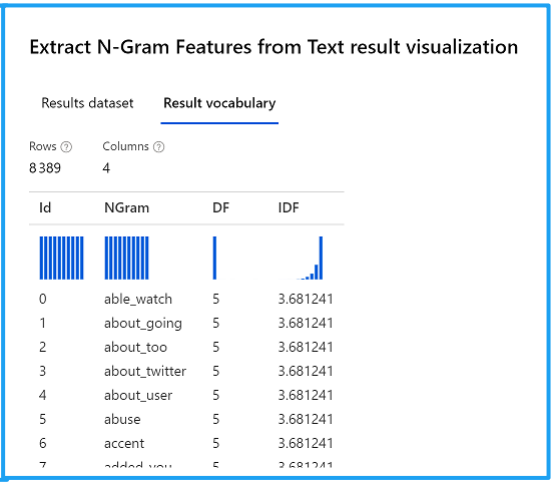
Le Hachage de caractéristiques transforme un flux de texte en un ensemble de caractéristiques numériques, similaire à du cryptage. Un espace de hachage de taille fixe est définie (par exemple, 1.024 – et autant de colonnes) et le texte est converti en une séquence d'index dans cet espace.



Extraction de caractéristiques n- grams

L'extraction de caractéristiques de n-grams du texte extrait d'un flux de texte un vocabulaire, affecte un valeur de présence (1) aux n-grams extraits lorsqu'ils existent dans le document, et attribue un score de fréquence (TF-IDF) à ces mêmes n-grams.





WORDS EMBEDDINGS OU PLONGEMENT DE MOTS

Le **plongement de mot** est une méthode de structuration (vectorisation) des données textuelles.

L'idée est de représenter chaque mot issu d'un corpus par un **vecteur (liste de nombres) de dimension fixe,** qui va en quelque sort « capturer son sens ».

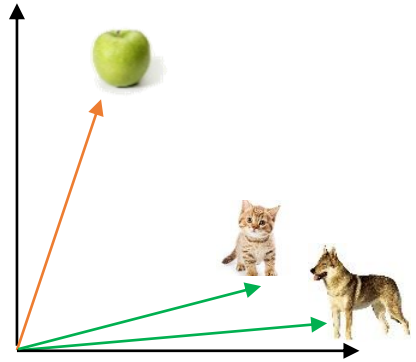
L'hypothèse est la suivante : les mots qui apparaissent souvent avec le même « contexte » (mots avant/après) sont probablement proches d'un point de vue sémantique, et peu distant dans un espace vectoriel.

Schématiquement, si on considère les phrases suivantes :

'Le chien est un animal'
'Le chat est un animal'
'La pomme est un fruit'

, on sait que *chien* et *chat* sont sémantiquement plus proches que *chien* et *pomme*, ou *chat* et *pomme*.

En présentant les vecteurs de ces trois mots sur 2 dimensions, on devrait voir la figure ci-après.



Il existe 2 principales méthodes d'entraînement des **word embeddings** :

- **CBOW** (Continuous Bag of Words), qui prédit un mot à l'aide du contexte (mots avant/après) ;
  Ex: Dans : '*Le* ____ *miaule*', on devine le mot *chat*

- **Skip-gram**, qui prédit – à contrario, le contexte (mots avant/après) à l'aide d'un mot ; Ex:
  '___ gros **chien** jappe'
  'Le ___ **chien** jappe'
  'Le gros **chien** ___'

WORDS EMBEDDINGS : À ENTRAÎNER OU PRÉ-ENTRAÎNÉS

Il existe 2 sortes de **Word Embeddings :**

- **Les embeddings « à entraîner » :** ils sont entraînés en temps réel sur les données d'entrée; on ajoute une couche d'Embedding à notre modèle, qui transforme les mots en vecteurs dans une matrice, celle-ci étant « apprise » au fur et à mesure que le modèle s'entraîne ;

- **Les embeddings « pré-entraînés » :** ils ont été pré-entraînés sur des corpus de texte très importants (Wikipedia, Google News, …) et ils permettent un temps d'entraînement réduit et souvent, de meilleurs performances globales.

Embeddings	Editeur	Année	Niveau	Téléchargement	Type	Architecture	Dimensions
Word2Vec	Mikolov et al	2013	Mots	~1.5GB	À entraîner	CBOW et SG	300
GloVe* (Global Vector for word representation)	Pennington et al	2014	Mots	~1.5GB	Pré-entrainé	Ratio de cooccurrence	25, 50, 100, 200
USE (Universal Sentence Encoder)	Google	2018	Phrases	~1GB	Pré-entrainé	DAN (Deep averaging network)	512

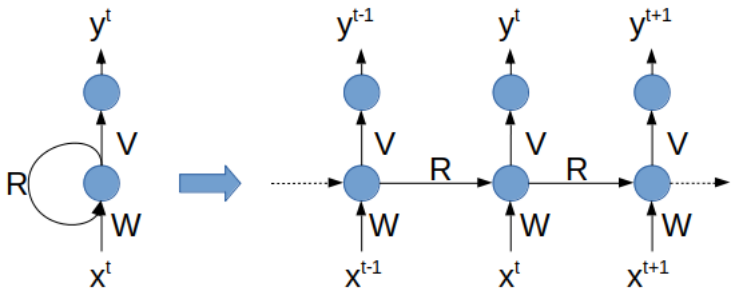
**Version Twitter*

RÉSEAUX DE NEURONES RÉCURRENTS (RNN)

Les **réseaux de neurones récurrents (ou RNNs)** sont un type de réseaux de neurones artificiels conçus de manière à **reconnaître les caractéristiques séquentielles** telles que les séries temporelles ou le langage naturel (textuel ou vocal).

En cela, ils sont plus proches du fonctionnement réel du système nerveux humain, au sens où ils **conservent en mémoire** un certain nombre d'états passés à un instant t.

Schématiquement, on dit que les RNNs utilisent des **boucles de rétroaction** pour traiter une séquence de données, afin de permettre aux informations de « persister » dans le temps.



RNN : vision récurrence vs. vision déplié (temporalité)
Source: Open Classrooms

Les RNNs utilisent la méthode d'apprentissage par **rétropropagation du gradient à travers le temps** pour apprendre les paramètres du réseaux. Néanmoins, ceux-ci tendent à se dégrader à mesure qu'ils grossissent et se complexifient.

Il existe différents RNNs selon l'étendue de leur « mémoire à court-terme »:

- Les **RNNs** « simples », qui oublient au bout de quelques itérations le passé dit proche ;
- Les unités **LSTMs** (Long Short-Term Memory), qui ont à la fois une mémoire long et court-terme, donc avec la capacité de distinguer les données importantes à mémoriser pour les réinjecter dans le réseau ; elles sont 2 fois plus lourdes que les couches récurrentes simples, compte tenu des connexions nécessaires.

Des variantes sont également disponibles pour les LSTMs:
- Les GRU (Gated Recurrent Units), qui réduisent le nombre de paramètres des modèles ;
- Les LSTM bidirectionnels (BLSTM), qui observent le passé mais également le futur.

MÉTRIQUES D'ÉVALUATION

AUC Score (aire sous la courbe)

L'AUC ou *Area Under the Curve* – l'aire sous la courbe, est une métrique « graphique » qui mesure la capacité d'un modèle a classifier correctement.

Les valeurs d'AUC sont comprises entre 0 et 1:

- Un modèle dont toutes les prédictions sont correctes a une AUC de 1;
- A contrario, un modèle dont 100% des prédictions sont erronées a une AUC de 0.

L'AUC ne doit pas être utilisée lorsque il y a une disparité importante entre le risque généré par les 2 types d'erreurs que le modèle peut faire, et où il est essentiel de minimiser l'un d'eux.

L'AUC n'est pas sensible aux seuils de classification.

Accuracy (justesse)

L'Accuracy – la justesse, est une métrique qui mesure la proportion de prédictions correctes sur un échantillon donné.

Elle se calcule de la manière suivante:

$$\textbf{acc} = \frac{Nombre\ de\ prédictions\ correctes}{Taille\ de\ l'échantillon}$$

ou

$$\textbf{acc} = \frac{True\ positives + True\ negatives}{Taille\ de\ l'échantillon}$$

L'accuracy est une métrique « globale » qui ne permet pas de voir en détail où un modèle fait des erreurs. Il faut l'associer à la lecture conjointe de la matrice de confusion.

TARIFICATION AZURE PAR SERVICE

Cognitive Services <clé en main>
Text Analytics > Sentiment

Gratuit
5000 transactions gratuites par mois

Standard (dégressif)	
Tous les 1.000 enregistrements de texte	
0 à 500k enregistrements	0,844€
500k à 2,5 millions	0,633€
2,5 à 10,0 millions	0,253€
> 10 millions	0,211€

Instance Clusters de calcul
Azure ML Studio (apprentissage)

Virtual Machine CPU	
Options recommandées: Machine virtuelle dédiée, de type CPU (vCPU), avec un quota de 20 cœurs *(coût par heure)*	
Usage général : D2 – 2 cœurs – 7GB ram – 14Go stockage	0,148€
Usage général : D3 – 4 cœurs – 14GB ram – 28Go stockage	0,296€
Mémoire optimisée : DS12 – 4 cœurs – 28GB ram – 28Go stckg	0,396€
Calcul optimisé :F4S – 4 cœurs – 8GB ram – 32Go stockage	0,396€

Clusters d'inférence
Azure ML Studio (déploiement)

Création de conteneur (€/UC)
0,00009€/seconde de tâches en cours d'exécution pour Basic, Standard et Premium
0,00009€/seconde d'allocation d'instance (nombre de processeurs alloués) pour les Pool d'agents dédiés: S1-2c-3Go ram, S2-4c-8Go ram ou S3-8c-16Go ram

Registre de conteneurs	
Tarif quotidien + Tarif par région répliquée	
Basic 10Go stockage	0,141€
Standard 100Go stockage	0,563€
Premium 500Go stockage Géoréplication (/région répliquée)	1,406€ 1,406€

Stockage additionnel: 0,003€/jour

RÉFÉRENCES

- Analyse de texte (text mining) avec Azure Cognitive Services - Text Analytics
- Solution Cloud de Machine Learning avec Azure Machine Learning et son concepteur (Drag and Drop)
- Deep Learning : Tensorflow - Réseaux de neurones récurrents, LSTM - Towards Data Science , word embeddings
- Pretrained word embeddings : word2vec, GloVe (Global Vectors for Word Representation , USE (Universal Sentence Encoder) - v4
- Déploiement : How and where to deploy Machine Learning models to Azure
- Autre : neptune.ai - structure and manage nlp projects
- Pour aller plus loin : Classification de texte avec BERT, Traduction de texte avec des modèles seq2seq (NMT: Neural machine translation with attention) , Traduction de texte avec des modèles Transformer

Ce document a été produit dans le cadre de la soutenance du projet n°7 du parcours Ingénieur IA d'OpenClassrooms :
« Détectez les Bad Buzz grâce au Deep Learning »

Mentor : Thierno DIOP
Evaluateur : Kezhan SHI

OPENCLASSROOMS
En partenariat avec
CentraleSupélec Microsoft