PARCOURS DATASCIENTIST

PROJET KAGGLE

Détermination de l’indice de toxicité de textes issus d’un réseau social

septembre 10, 2019

CONTENTS

Table of Contents

1 Introduction

2 Analyse du corpus

2.1 Analyse descriptive du corpus

Un texte d'une conversation est dit toxique si son contenu amène des participants à quitter la conversation.
Le corpus est constitué de près de 2 millions de textes et de 45 colonnes.

Un indice de toxicité allant de 0.0 à 1.0 est associé à chacun des textes. L'indice 1.0 indique la plus forte toxicité.

Des sous-catégories sont associées à un texte :
La sous catégorie de l'origine ethnique comprend les colonnes:

```
asian
latino
black
white
other_race_or_ethnicity
```

La sous catégorie religion comprend les colonnes:

```
atheist
buddhist
christian
hindu
jewish
muslim
other_religion
```

la sous catégorie du genre comprend les colonnes:

```
female
male
other_gender
transgender
```

La sous catégorie orientation sexuelle comprend les colonnes:

```
bisexual
heterosexual
homosexual_gay_or_lesbian
other_sexual_orientation
```

La sous catégorie handicapes comprend les colonnes:

```
intellectual_or_learning_disability
other_disability
physical_disability
psychiatric_or_mental_illness
```

Les colonnes dites de signalisation, donnant de l'information sur la création des commentaires :
- `created_date`
- `publication_id`
- `parent_id`
- `article_id`

Les colonnes relatant les sentiments des commentaires :
- `likes`
- `disagree`
- `sexual_explicit`
- `funny`
- `wow`
- `sad`

Les colonnes liées aux informations sur les comunity managers :
- `rating`
- `identity_annotator_count`
- `toxicity_annotator_count`

L'indice de toxicité d'un texte est du fait de la colonne `target`.

Les textes sont sous la colonne `comment_text`

2.2 Description statistique

La description statistique du corpus est focalisée sur la structure des textes et les indices de toxicité.

2.2.1 Structure des textes

10 textes pris aléatoirement dans le corpus sont exposés ci-dessous :

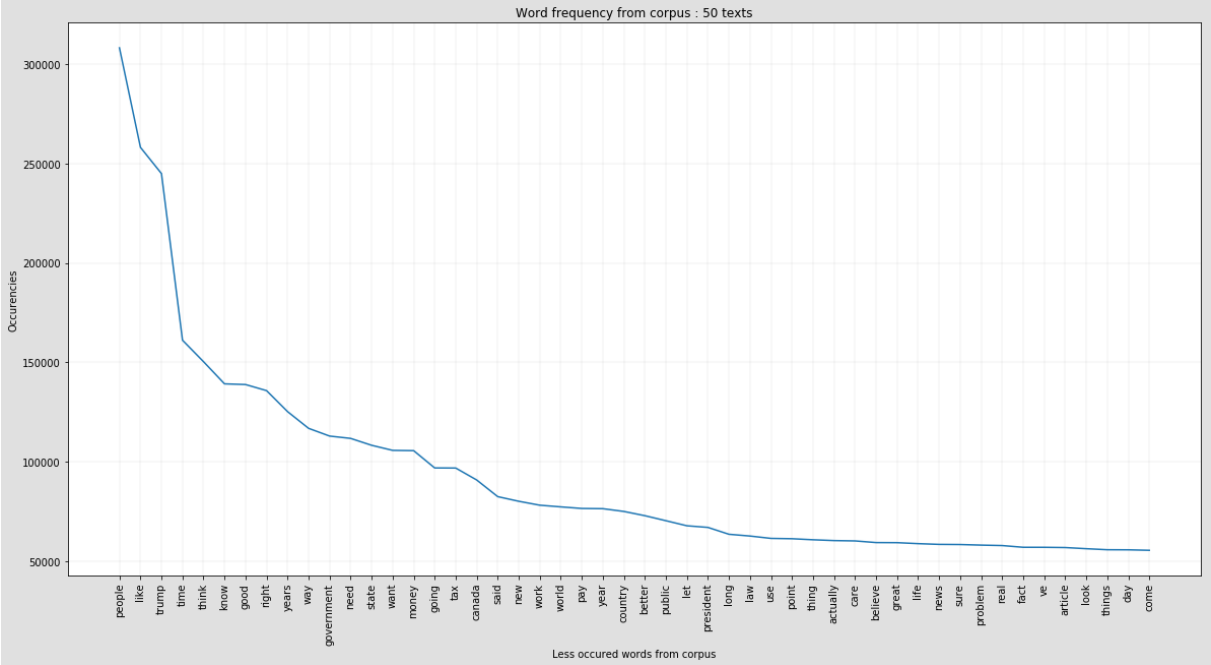
```
Corpus texts= 1804874

    target                          comment_text
   ------------------------------------------------------------
0  0.000000  this is so cool. it's like, 'would you want yo...
1  0.000000  thank you!! this would make my life a lot less...
2  0.000000  this is such an urgent design problem; kudos t...
3  0.000000  is this something i'll be able to install on m...
4  0.893617              haha you guys are a bunch of losers.
5  0.666667                                ur a sh*tty comment.
6  0.457627                         hahahahahahahhha suck it.
7  0.000000                                 ffffuuuuuuuuuuuuuu
8  0.000000  the ranchers seem motivated by mostly by greed...
9  0.000000  it was a great show. not a combo i'd of expect...
```

Cet échantillon met en évidence :
- des mots n'appartenant pas au vocabulaire du langage naturel
- des mots masqués comme le texte #7, qui est référencé avec un indice de toxicité à 0.
- les mots ne suffisent pas à identifier un texte toxique, comme le montre le texte #4. L'ensemble de ces mots n'appartiennent à aucune des sous catégories pouvant conduire à un indice de toxicité élevé (insulte, origine ethnique, orientation sexuelle….)
- La différence de longueur des différents textes.

2.2.2 Fréquence des mots dans le corpus

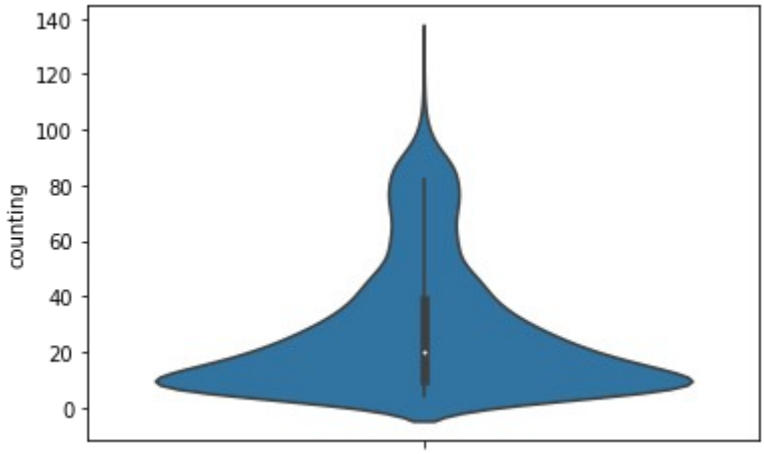
La figure ci-dessous montre les 50 mots les plus fréquents dans le corpus.



Le mot « Trump » arrive dans les tous premiers mots. La syntaxe de ce nom propre peut être confondu avec le nom commun trompette traduit en anglais, soit, trump.

2.2.3 Distribution de la taille des textes

La taille des textes est mesurée avec le nombre de tokens les composants. Cette mesure nécessite la tokenization des textes au préalable.
La figure ci-dessous montre la distribution de la taille des tokens après un processus de tokenization réalisé avec la librairie Spacy.



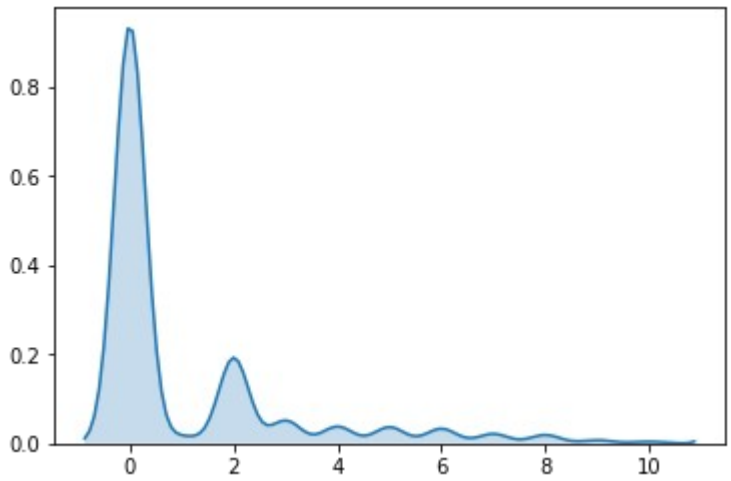
mean	*27.991291*
std	*23.849611*
min	*4.000000*
25%	*10.000000*
50%	*20.000000*
75%	*39.000000*
max	*129.000000*

Distribution de la aille des textes tokenisés sur un échantillon de 15K textes

Cette distribution laisse apraître deux modes : les textes ayant un nombre de tokens autour de 10 et un autre mode ayant un nombre de tokens autour de 80.

2.2.4 Analyse de l'indice de toxicité

La figure ci-dessous montre la distribution normalisée des indices de toxicité

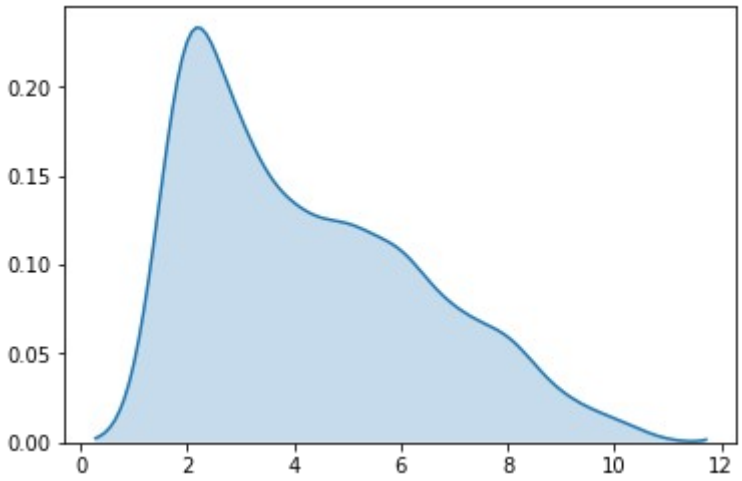


Distribution normalisée des indices de toxicité

On constate que :

- Les textes sains représentent près de 70 % du corpus contre 30 % pour les textes ayant un indice de toxicité supérieur à zéro.
- La fréquence de textes toxiques décroît avec l'indice de toxicité.
- Les textes fortement toxiques, avec un indice de 6 et plus, sont rares.

La représentation qui suit fait abstraction des textes sains. Elle représente la distribution des textes avec un indice de toxicité positif strictement.



Distribution normalisée des indices de toxicité positifs

2.2.5 Analyse multi-variée de l'indice de toxicité et des sous-catégories

Les sous-catégories sont les variables qualitatives conduisant à produire la valeur de l'indice de toxicité d'un texte.

3 Data-préparation

Le processus de data-préparation est réalisé avec la librairie Spacy.
Il consiste en les étapes suivantes :

- Le nettoyage des textes du corpus
- La standardisation des textes du corpus et leur tokenisation

3.1 La standardisation des textes

Cette étape est décomposée en les sous-étapes suivantes :

- Le remplacement des mots dits « suspects » par le mot clé « unknown ».
  L'analyse des textes au paragraphe 2.2.1 met en évidence des mots inconnus du dictionnaire du langage naturel. Ces mots sont du aux erreurs d'orthographe ou à la volonté de les masquer tout en faisant ressortir leur sens au lecteur.
- La détection des entités (pays, entreprises, organisations..) par le mot clé « entity ». Cette option est activable ou pas.
  Cette substitution permet de préserver la sémantique du texte. En substituant le nom d'un pays ou d'une organisation par un nom commun, l'information de la présence d'une entité dans le texte est préservé.
- Chacun des textes est limité à un nombre maximum de mots

A l'issue de ce processus, chacun des textes est représenté sous la forme d'une liste de tokens.

3.2 Le nettoyage des textes

Le nettoyage des textes. Cette étape consiste à

- ne préserver que les mots dans le vocabulaire de la librairie Spacy ;
- préserver les N mots les plus fréquents qui ne sont pas des stop-words ;
- supprimer du corpus les textes avec un nombre de mots inférieur à un seuil ;
- supprimer du corpus les textes avec un nombre de mots supérieur à un seuil. ;
- lemmatiser les mots des textes en dehors de la liste des mots les plus fréquents ;
- supprimer des mots les plus courants du langage (stop-words)
- supprimer de la ponctuation ;
- désaccentuer des caractères ;
- passer en minuscule de tous les mots.

3.3 La numérisation du corpus

La représentation numérique de chacun des textes tokenisés et formant le corpus, peut se faire selon deux modes :

- une représentation vectorielle 1D du texte
- une représentation matricielle 2D du texte.

Pour ces deux modes de représentation, la représentation vectorielle Glove, intégrée à la librairie Spacy a été mise en œuvre.

Chacune de ces deux représentations va être benchmarkée avec différents types d'estimateurs.

3.3.1 Représentation vectorielle des textes du corpus

Chacun des textes tokenisé du corpus va être représenté sous la forme d'un vecteur dont la dimension est la dimension des vecteurs de Glove, soit, 300.

Le processus commence par décrire sous forme matricielle la représentation tokenisée d'un texte :

Text tokensisé	dim_0	dim_1	dim_p
token_0	Text tokenisé projeté sur la dimension 0	M[0,1]	Text tokenisé projeté sur la dimension P
token_i		M[i,1]	
token_N		M[N,1]	
		$vector_1 = \sum_{i=1}^{N} M_{i,1}$	$vector_p = \sum_{i=1}^{N} M_{i,p}$

La représentation vectorielle d'un texte est obtenue en sommant la représentation vectorielle de chacun des tokens du texte.

3.3.2 Interprétation de la représentation vectorielle

Le vecteur résultat, $vector = (vector_{1,..} vector_p)$ peut être interprété en recherchant les mots similaires dans l'espace des mots vectorisé de Glove. La mesure de similarité utilisée emploie la fonction produit scalaire dont le résultat est normalisé (la similarité cosinus).

L'interprétation de la similarité de la somme algébrique des tokens d'un texte dans l'espace des mots de Glove :

Texte original	thank you for stating the fact that the israeli wall is there to protect the inhabitants of israel , some of whom are israeli arabs , from palestinian and other arab terrorist attacks .
Texte tokenisé	thank state fact israeli wall protect inhabitant israel israeli arab palestinian arab terrorist attack
Mots les plus similaires à vector	'israelian', 'Isreali', 'KIBBUTZ'

Le texte est vectorisé en l'équivalent d'un seul mot. Ce mot n'est pas dépourvu de sens au regard des trois mots les plus similaires et du texte original.

3.3.3 Représentation matricielle des textes du corpus

Chacun des textes tokenisé du corpus va être représenté sous la forme d'une matrice dont :

- le nombre de lignes est la dimension des vecteurs de Glove, soit, 300 et
- le nombre de colonnes est le nombre de tokens du texte tokenisé.

La standardisation des textes du corpus décrite en permet de s'assurer de l'identité du nombre de colonnes de tous les textes ainsi représentés, soit N ce nombre.

		Dim 0	...	DIM D
Text t	Token 0	$coeff_{t,0,0}$		$coeff_{t,0,D}$
	Token 1	$coeff_{t,1,0}$		$coeff_{t,1,D}$
	...			
	Token N			
		$coeff_{t,N,0}$		$coeff_{t,N,D}$

Le vecteur ($coeff_{t,0,0}$ … $coeff_{t,0,D}$) représente la projection du text t sur la première dimension de l'espace des mots de Glove, le texte t étant composé, dans l'ordre, des tokens énumérés de 0 à N.

- Text t représente le $t^{ème}$ texte du corpus, $\in$ [0,M-1]

- Les colonnes DIM 0, …, DI M représentent les dimensions de l'espace des mots de Glove. nombre maximum de tokens par texte, soit N.
- Token 0, … Token N représentent la décomposition en N tokens du texte t.

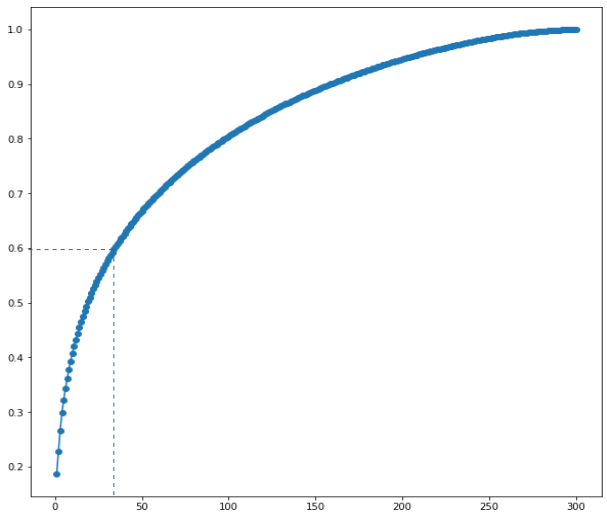
3.3.4 Réduction de dimension des textes du corpus

L'espace des mots permet une représentation numérique d'un mot avec 300 coefficients. Il peut s'avérer nécessaire de réduire le nombre de coefficients nécessaires à représenter un mot.

La réduction de la dimension de l'espace des mots est mis en œuvre avec un algorithme de réduction PCA (analyse en composantes principales) incrémental.

Le diagramme ci-dessous montre que le taux de variance expliqué en fonction du nombre de composantes principales retenues.

L'échantillon de textes traités est de 20K. Le taux de variance expliqué retenue, 60 %, amène à définir un espace de dimension de l'ordre de 40.



Réduction de dimension pour une variance de 60% sur 20K textes

Le diagramme ci-contre montre que le taux de variance expliqué en fonction du nombre de composantes principales retenues.

Le nombre de documents traités numérisés est de 20K. Le taux de variance expliqué retenue, 60 %, amène à définir un espace de dimension de l'ordre de 40.

La réduction de la dimension de l'espace des mots est mis en œuvre avec un algorithme de réduction PCA (analyse en composantes principales) incrémental

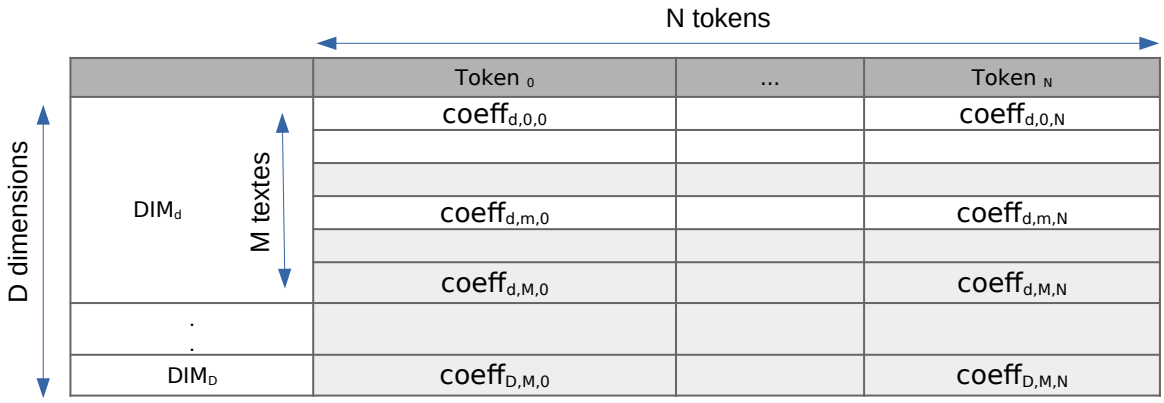
.

Pour calculer le taux de variance de l'échantillon, le tenseur de dimensions (M,N,D) a été transformé en une matrice de dimensions (MxN, D).

Le même opérateur de réduction de dimension est utilisé pour les dataset de test et validation. Cela garantie une dimension identique pour l'espace de mots, même si le nombre d'échantillons du dataset de validation est inférieur à celui du dataset d'entraînement.

3.3.5 Multiplexage de dimension des textes du corpus

L'étape suivante consiste à multiplexer par les dimensions, la représentation matricielle décrite ci-dessus. Cette nouvelle représentation est schématisée ci-dessous :



La représentation numérique finale du corpus est donc un tenseur de dimensions (D,M,N) où :

- D est la dimension réduite de l'espace des vecteurs de mots ;
- M est le nombre de textes (documents) numérisés du corpus ;
- N est le nombre de tokens pour chacun des documents du corpus. Le processus de standardisation décrit en 3.1 a conduit à ce que ce nombre soit égale pour tous les textes.

Il s'ensuit les notations suivantes, adoptées pour la suite:

- DIM$_d$: représente la d^{ème} dimension de l'espace des mots, d $\in$ [0, D-1]
- Token $_n$: représente le n^{ème} token d'un texte , n $\in$ [0, N-1]
- coeff$_{m,d,0}$: représente le coefficient du token 0 du texte m, m $\in$ [0, M-1] selon la dimension d de l'espace des mots réduits.

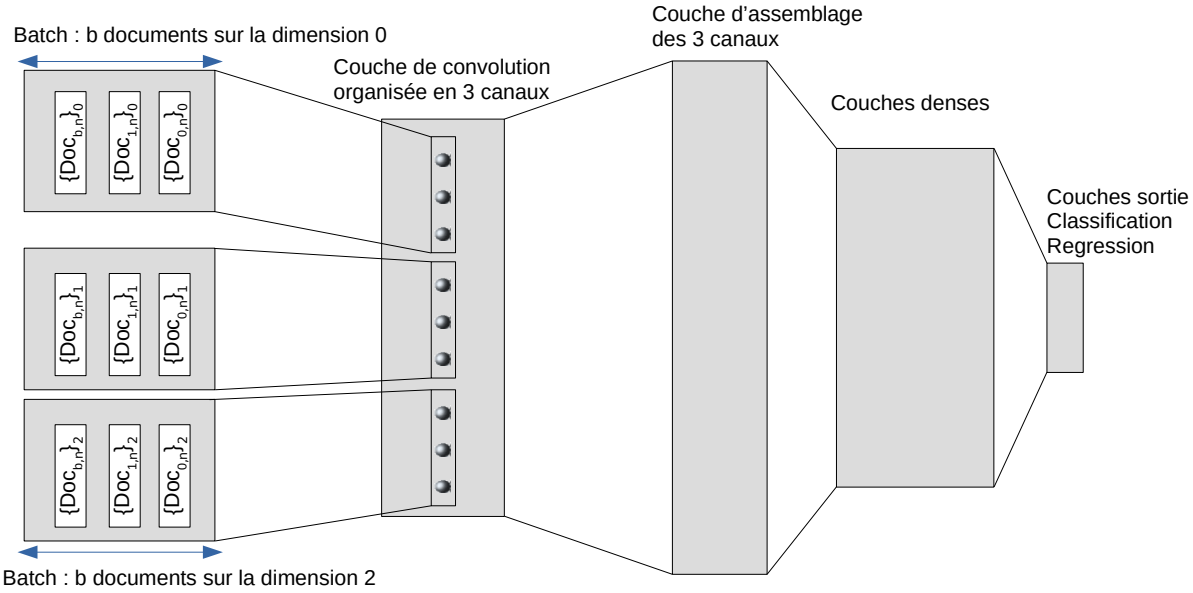
3.3.6 Interprétation de la représentation matricielle multiplexée des textes du corpus

Cette représentation revient à multiplexer les données numérisées du corpus sur les dimensions de l'espace des mots.

Ainsi, le vecteur de composantes $\{coeff_{d,m,0}, ..., dim_{tk,d,N}\}$ représente la projection sur la dimension d des N tokens numérisés du document (texte) Tk .

Notons $\{Doc_{m,n}\}_d$ les coefficients vecteur projeté sur la dimension d, du $m^{ème}$ document composé de N tokens numérisés et indicés par n.

Cette représentation va permettre d'alimenter un réseau de neurones de convolutions dans lequel, chacune des projection d'un texte sera traité par un sous-réseau du réseau de convolution.
Le schéma ci-dessous permet d'illustrer ce propos avec trois dimensions.

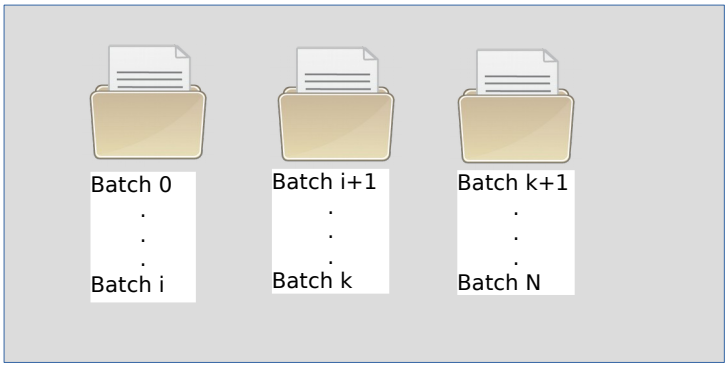


Architecture en canaux dun CNN réalisé avec Keras

4 L’organisation des données

Afin d’optimiser l’utilisation de la mémoire vive (RAM) dans la phase d’apprentissage des modèles en réseaux de neurones, les dataset issus de la préparation sont stockés sous forme de partitions.

Une partition est un ensemble de fichiers associés à un dataset.
Le fichier est parcouru par plusieurs batchs.



Organisation d'un dataset en partitions

La figure ci-dessus décrit l’organisation d’un dataset en partitions composé de trois fichiers, chacun d’eux décrit par une énumération de batchs.

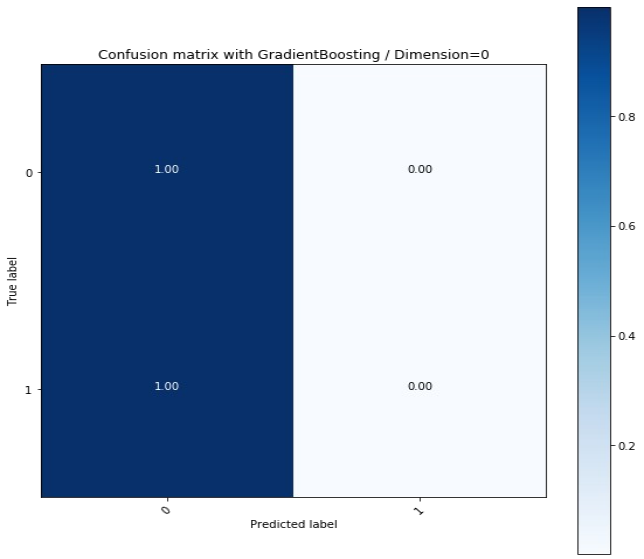
Une epoch est l’ensemble des batchs nécessaires au parcours de tous les items du dataset.

Lors de la phase d’apprentissage d’un modèle, ces fichiers sont lu successivement en fonction des indices de batch qui augmentent avec la progression de l’epoch.

5 Le problème binaire

Pour estimer la qualité de la data-préparation, des estimateurs de la famille du gradient boosting vont être appliqués à la data-préparation décrite au chapitre 3.

La représentation du multiplexage en dimension va être utilisée sur plusieurs dimensions séparément, ce, pour mettre en évidence la susceptibilité des estimateurs à la variance capturée par une dimension.



e

La projection des textes sur la dimension 0 montre que l’estimateur est susceptible de classifier les textes sains avec un résultat de vrais positifs de 100 %.

Cette dimension capture la plus de variance des dimensions du sample.

6 Estimateur CNN

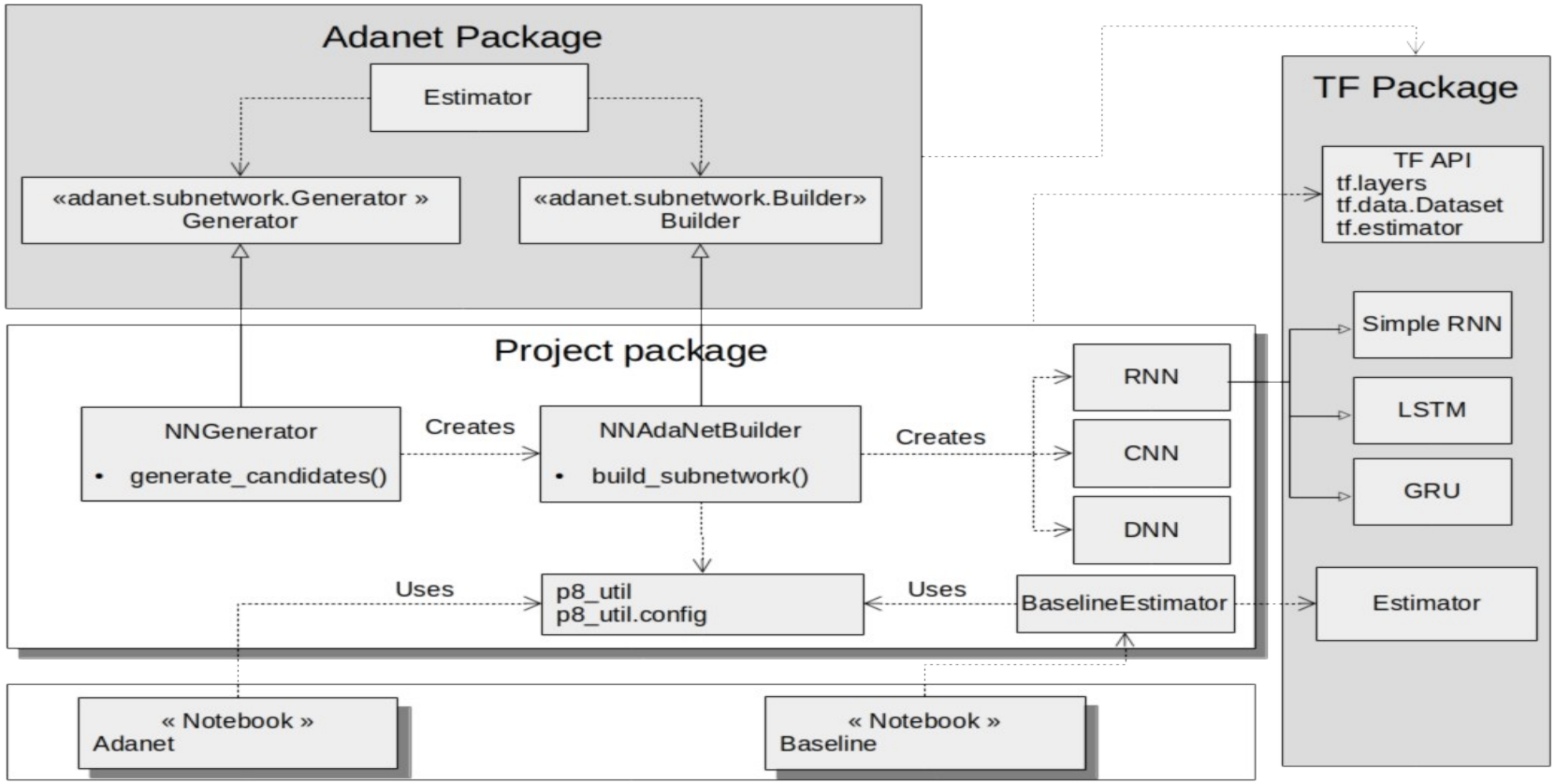
Il faut noter que le nombre de composantes nécessaire pour un taux de variance expliqué donné, croît avec le nombre d'échantillons.

7 Résultats

7.1 Reseau CNN

7.2 Régularisation de la complexité de Rademacher

8 *Ingénierie logicielle*



Les notebook ***Adanet.ipynb*** et ***Baseline.ipynb*** permettent de produire des résultats éponymes.

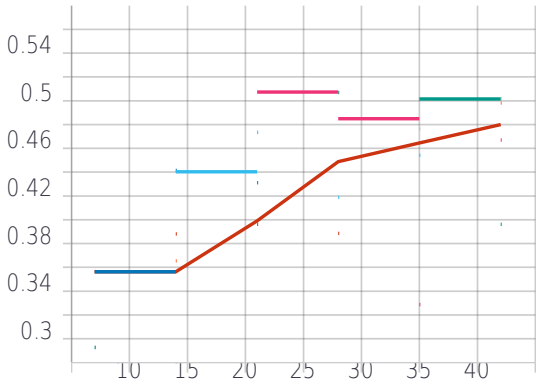
Ils utilisent tous deux le fichier ***p8_util_config.py*** fixant la configuration de chacune des expérimentations. Ce, dans la perspective de faciliter les comparaisons issues des expérimentations.

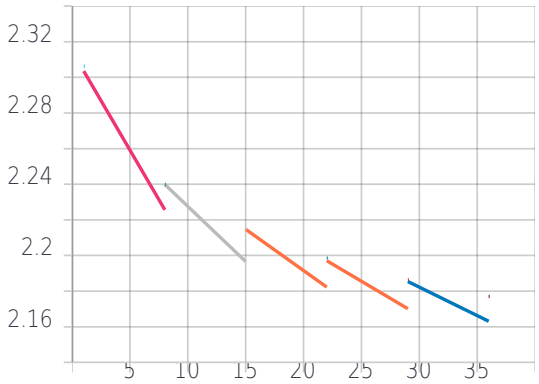
Le fichier ***p8_util.py*** implémente les fonctions utilitaires nécessaires aux expérimentations du projet.

Le fichier ***BaselineEstimator.py*** utilise la classe ***Estimator*** de Tensorflow. ***BaselineEstimator*** est une implémentation de l'estimateur customisé de la baseline, utilisant les API ***tf.estimator*** fournies par Tensorflow.

Le fichier ***NNGenerator.py*** implémente la classe ***NNGenerator*** qui est une extension de la classe ***Generator*** du framework ***Adanet***. Cette classe a la charge de renvoyer les sous-réseaux candidats à l'algorithme Adanet. Pour ce faire, elle délègue la fabrication des sous-réseaux à la classe ***NNAdanetBuilder*** implémentée dans le fichier ***NNAdanetBuilder.py.***

La classe ***NNAdanetBuilder*** construit les différents types de sous réseaux, RNN, CNN et DNN. Ces derniers utilisent l'API ***tf.layers*** de Tensorflow pour les classes du même nom.


0.54
0.5
0.46
0.42
0.38
0.34
0.3
10
15
20
25
30
35
40


2.32
2.28
2.24
2.2
2.16
5
10
15
20
25
30
35

```
Global steps  : ................................ 300
NN type              : ........................... RNN
Features shape       : ........................... (28, 28)
Adanet boosting iter.: ........................... 40
Adanet iter per boost: ........................... 7
Dropout rate         : ........................... 0.0
Seed value           : ........................... 42
Nb of classes (logit): ........................... 10
Adanet regularization: ........................... 1e-05
Weights initializer  : ........................... truncated_normal
Batch normalization  : ........................... True
Learn mixture weights: ........................... True
Cell type            : ........................... SGRU
Hidden units         : ........................... 128
Stacked cells        : ........................... 2
Time steps           : ........................... 28
```