Projet 8 OC : Compétition kaggle - Jigsaw Multilingual Toxic Comment Classification

Présenté par : NAMA NYAM Guy Anthony

Mentor : Julien Hendrick

24 Juin 2020

OPENCLASSROOMS

Sommaire

Mise en situation

- Dans les conversations en ligne, Il suffit d'un seul commentaire toxique pour aigrir une discussion.
- La toxicité est définie comme tout ce qui est grossier, irrespectueux ou autrement susceptible de pousser quelqu'un à quitter une discussion.
- Ses toxicités dans les conversations en ligne peuvent être détectées par les modèles de machine learning.
- Les utilisateurs des plateformes de discussion en ligne proviennent de tous les quatre coins du monde, c'est dire la diversité des commentaires multilingue qu'on y retrouvent.

Objectif

Contribuer à l'identification de commentaires toxiques dans différentes langues pour un Internet plus sûr et collaboratif.

Approches

- Pour ce faire, nous utilisons les représentations vectorielles denses du langage naturel par les architectures de réseaux de neurones profond pré-entraînés.
- Dans ce contexte multilingue, nous utilisons les capacités croissantes et impressionnantes de modèles des derniers innovations.
- Il s'agit principalement des modèles multilingues ***BERT*** et ***XLM***
- Nous tirons profit des ressources informatiques grandissantes ***TPU*** pour améliorer notre capacité de modélisations.

Sommaire

Description

- Nous avons 4 jeux de données :
  1. ***jigsaw-toxic-comment-train.csv*** : dataset de commentaires en anglais provenant des discussions Wikipedia(Dataset d'entraînement)
  2. ***jigsaw-unintended-bias-train.csv*** : second dataset de commentaires provenant de commentaires civils(Dataset d'entraînement)
  3. ***validation.csv*** : commentaires des pages de discussion Wikipedia dans différentes langues non anglaises(Dataset de validation)
  4. ***test.csv*** : commentaires des pages de discussion Wikipedia dans différentes langues non anglaises(Dataset de test)

  Colonnes prépondérantes à la suite du travail :
- **id** - identifiant à l'intérieur de fichier correspondant au commentaire.
- **comment_text** - le texte du commentaire à classer
- **lang** - le langage du commentaire
- **toxic** - si le commentaire est classé comme toxique ou non.

Preprocessing - fusion train dataset

- Taille des dataset et mise en évidence de la colonne cible *toxic*.

```
df_train_tc.shape
```

```
(223549, 8)
```

```
df_train_tc.sample(10)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
10692	1c469963e53675e1	U smell like a rat ctcher? \n\nhey! don't bluf...	1	0	0	0	1	0
28847	4c740bed4ce3b3f9	Life of About Tanaya \n\nshe is mean I hate her	1	0	0	0	0	0
5840	0f9c3ecf18ba5190	Problem solved. The person who originally put ...	0	0	0	0	0	0

```
df_train_ub.shape
```

```
(1902194, 45)
```

```
df_train_ub.sample(5)
```

	id	comment_text	toxic	severe_toxicity	obscene	identity_attack	insult	threat
507574	865362	Really? Who made you an expert? Read Letters...	0.400000	0.1	0.0	0.2	0.200000	0.0
526254	887460	Brian Mulroney is hardly a "neophyte negotiator".	0.000000	0.0	0.0	0.0	0.000000	0.0
944627	5274242	http://nfs.sparknotes.com/juliuscaesar/page_13...	0.166667	0.0	0.0	0.0	0.166667	0.0

Preprocessing - fusion train dataset

- Mettre 0 si probabilité inférieur à 0.5, 1 sinon

```
for f in ['toxic']:
    df_train_ub.loc[df_train_ub[f] >= 0.5, f] = 1
    df_train_ub.loc[df_train_ub[f] < 0.5, f] = 0
```

- Observation des proportions de labels dans chaque dataset

```
df_train_tc.toxic.value_counts()
```

```
0    202165
1     21384
Name: toxic, dtype: int64
```

```
df_train_ub.toxic.value_counts()
```

```
0    1750083
1     152111
Name: toxic, dtype: int64
```

- Fusionner l'ensemble du dataset de *wikipédia* avec une partie du dataset *comment civils*.

```
df_train = tm.pd.concat([df_train_tc, df_train_ub[df_train_ub.toxic == 1]], axis=0, join='inner', ignore_index=True)
df_train.shape
```

```
(375660, 3)
```

Preprocessing - multiprocessing

- Train dataset

```
df_train.head(3)
```

	id	comment_text	toxic
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0

```
df_train.toxic.value_counts()
```

```
0    202165
1    173495
Name: toxic, dtype: int64
```

- L'ensemble des tâches de nettoyage sont en ***multiprocessing***.

```
from multiprocessing import  Pool

def parallelize(data, func, num_of_processes=4):
    data_split = np.array_split(data, num_of_processes)
    pool = Pool(num_of_processes)
    data = tm.pd.concat(pool.map(func, data_split))
    pool.close()
    pool.join()
    return data
```

```
df_train = parallelize(df_train, emoji_replace)
```

Preprocessing - nettoyage urls

- Suppression des urls

```
import re

def remove_URL(df):
    """Remove URLs from a sample string"""
    for i, row in df.iterrows():
        text = row["comment_text"]
        text = re.sub(r'http\S+', "", text)
        text = re.sub(r'www.\S+', "", text)
        df.loc[i, "comment_text"] = text
    return df
```

- Exemples textes avant et après suppression urls.

id	comment_text
)dfd995809fa	"\n\n Snowflakes are NOT always symmetrical! \n\nUnder Geometry it is stated that ""A snowflake always has six symmetric arms."" This assertion is simply not true! According to Kenneth Libbrecht, ""The rather unattractive irregular crystals are by far the most common variety"" http://www.its.caltech.edu/~atomic/snowcrystals/myths/myths.htm#perfection Someone really need to take a look at his site and get FACTS off of it because I still see a decent number of falsities on this page. (forgive me Im new at this and dont want to edit anything)"
3e1dbe91225	I was able to post the above list so quickly because I already had it in a text file in my hard drive I've been meaning to get around to updating the sound list for some time now. \nAs far as generating interest I've spent four years trying to drum up more interest in freely licensed full length classical music. Unfortunately, my attempts failed - I'm still effectively the only one who does it. The classical music wikiproject was not interested, (Wikipedia_talk:WikiProject_Classical_music/Archive_5#Need_help.21Wikipedia_talk:WikiProject_Music/Archive_3#I_could_use_some_helpWikipedia_talk:WikiProject_Music/Archive_2#Raulbot.2C_and_the_music_list) So I really had given up trying to interest others. \nThe sound list was featured on digg a while back - http://digg.com/music/Wikipedia_has_free_classical_music_downloads It got 1600 diggs, which is IMO very impressive.

id	comment_text
)dfd995809fa	"\n\n Snowflakes are NOT always symmetrical! \n\nUnder Geometry it is stated that ""A snowflake always has six symmetric arms."" This assertion is simply not true! According to Kenneth Libbrecht, ""The rather unattractive irregular crystals are by far the most common variety:"" Someone really need to take a look at his site and get FACTS off of it because I still see a decent number of falsities on this page. (forgive me Im new at this and dont want to edit anything)"
3e1dbe91225	I was able to post the above list so quickly because I already had it in a text file in my hard drive I've been meaning to get around to updating the sound list for some time now. \nAs far as generating interest I've spent four years trying to drum up more interest in freely licensed full length classical music. Unfortunately, my attempts failed - I'm still effectively the only one who does it. The classical music wikiproject was not interested, (Wikipedia_talk:WikiProject_Classical_music/Archive_5#Need_help.21Wikipedia_talk:WikiProject_Music/Archive_3#I_could_use_some_helpWikipedia_talk:WikiProject_Music/Archive_2#Raulbot.2C_and_the_music_list) So I really had given up trying to interest others. \nThe sound list was featured on digg a while back - It got 1600 diggs, which is IMO very impressive.

Preprocessing - nettoyage contract forms

- Forme longue des contractions

```
import contractions

def contract(df):
    """Resolving contractions"""
    for i, row in df.iterrows():
        df.loc[i, "comment_text"] = contractions.fix(str(row["comment_text"]), slang=False)
    return df
```

- Exemples textes avant et après remplacement forme contractée.

	id	comment_text	toxic
0	0000997932d777bf	Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.	0

	id	comment_text	toxic
0	0000997932d777bf	Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They were not vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please do not remove the template from the talk page since I am retired now.89.205.38.27	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0
2	000113f07ec002fd	Hey man, I am really not trying to edit war. It is just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.	0

Preprocessing - nettoyage emoji

- Remplacement des ***emoji*** par la forme textuelle.

```
import emoji

def emoji_replace(df):
    for i, row in df.iterrows():
        text = str(row["comment_text"])
        if emoji.emoji_count(text):
            dem = emoji.demojize(text)
            tab = dem.split(":")
            tab = [' '.join(tab[j].split('_')) for j in range(len(tab))]
            k = 0
            #  Delete emoji successive
            while k < (len(tab) - 2):
                if (tab[k+1] == '') and (tab[k] == tab[k+2]):
                    del tab[k+2]
                    del tab[k+1]
                else:
                    k += 1
            res = ' '.join(tab).strip()
            df.loc[i, "comment_text"] = res
    return df
```

- Exemples textes avant et après remplacement émoji.

	id	comment_text	toxic
268766	298792	I love this on so many levels. What an awesome project. OMGosh the smiles on those kids faces 😍😍🚲🚲🚲🚲🚲🚲🚲🚲	0
272781	303810	Lol " he just pooed"💩	0
273790	305425	All the more reason to vote Bernie🔥🔥🔥🔥. When all the other countries provide healthcare as a basic right for their people, there can only be one reason the United States does Not....IT IS CALLED GREED!!!!!	0

	id	comment_text	toxic
268766	298792	I love this on so many levels. What an awesome project. OMGosh the smiles on those kids faces smiling face with heart-eyes bicycle	0
272781	303810	Lol " he just pooed" pile of poo	0
273790	305425	All the more reason to vote Bernie fire. When all the other countries provide healthcare as a basic right for their people, there can only be one reason the United States does Not....IT IS CALLED GREED!!!!!	0

Preprocessing - nettoyage Unidecode

- Représentation l'ensemble du texte dans l'encodage **ASCII**

```
from unidecode import unidecode

def unicode_characters(df):
    for i, row in df.iterrows():
        df.loc[i, "comment_text"] = unidecode(str(row["comment_text"]))
    return df
```

- Exemples textes avant et après encodage *ASCII*

	id	comment_text	toxic
161462	080ede0b55b91cfd	salât-ı cum'a ve salât-ı îdeyn diye bir tabir varmış. \n\n burada da ikame-i salât-i Cuma ve ideyni...	0
193508	88c82bd9c92ed310	merhaba \n\n eklemeleriniz için çok teşekkür ederiz	0
222787	ff0136d35909156c	We can use numbers in Yurtdışında Yaşayan Türk Vatandaşları (T.C. Dışişleri Bakanlığı) for Turkish citizen (Türk vatandaşı). But these number are invalid for Turkish People (Türkiye Türkleri).	0

	id	comment_text	toxic
161462	080ede0b55b91cfd	salat-i cum'a ve salat-i ideyn diye bir tabir varmis. \n\n burada da ikame-i salat-i Cuma ve ideyni...	0
193508	88c82bd9c92ed310	merhaba \n\n eklemeleriniz icin cok tesekkur ederiz	0
222787	ff0136d35909156c	We can use numbers in Yurtdisinda Yasayan Turk Vatandaslari (T.C. Disisleri Bakanligi) for Turkish citizen (Turk vatandasi). But these number are invalid for Turkish People (Turkiye Turkleri).	0

Preprocessing - nettoyage misspelling words

- Détection des mots "supposés" mal orthographiés.

```
from spellchecker import SpellChecker
spell = SpellChecker()

def suggest_misspelled(df):
    misspelt = dict()
    for i, row in df.iterrows():
        doc = nlp(str(row["comment_text"]))
        misspelled = spell.unknown([token.text for token in doc])
        for unknown in misspelled:
            if unknown in misspelt:
                misspelt[unknown] += 1
            else:
                misspelt[unknown] = 1

    with open('miss{}.txt'.format(i), 'w') as outfile:
        json.dump(misspelt, outfile)
```

- Remplacer les mots d'occurrences inférieurs ou égale à 3(+400 milles)

```
def misspelling_replace(words):
    misspelling_repl = []
    for word in words:
        misspelling_repl.append((word, spell.correction(word)))
    return misspelling_repl
```

```
misspelling_replace
```

```
 ('tapdance', 'tapdance'),
 ('zumpta', 'humpty'),
 ('doli', 'doll'),
 ('chiffer', 'schiffer'),
 ('nabila', 'nabil'),
 ('|result', 'result'),
 ('alfentanil', 'alfentanil'),
 ('toomas', 'thomas'),
```

Preprocessing - nettoyage astericks words

- Détection des mots avec astérisques

```
import spacy
nlp = spacy.load('en')

def asterick_words_detect(df):
    asterick_words = []
    for i, row in df.iterrows():
        doc = nlp(str(row["comment_text"]))
        for token in doc:
            if '*' in token.text[1:-1]:
                asterick_words.append(token.text)
    with open('ast{}.txt'.format(i), 'w') as outfile:
        json.dump(misspelt, outfile)
```

- Remplacement à la main les mots avec astérisques. Exemple d'extrait.

```
len(asterick_replace)
```

```
1075
```

```
asterick_replace
```

```
 ('d**n', 'damn'),
 ('big*ted', 'bigoted'),
 ('ClusterF**K', 'ClusterFucK'),
 ('N*gga', 'Nigga'),
 ('a*sholes', 'assholes'),
 ('di*ks', 'dicks'),
 ('F*uck', 'Fuck'),
 ('sc*m', 'scam'),
 ('G***Y', 'GAY'),
 ('l*es', 'lies'),
 ('fu**ing', 'fucking'),
 ('n****s', 'niggers'),
```

Preprocessing - nettoyage multiple space

- Exemples textes avant et après remplacement des mots avec astérisques.

	id	comment_text	toxic
201	007f1839ada915e6	Your blatant POV pushing Neither of you guys has made any contribution to this Italian history article other than to shove your unhistorical unconstructive modern POV in my face. This is a HISTORY article. HISTORY. Have you heard of that? This is the reason why so many people get pissed off about the pedantry and idiocy and triviality of Wikipedia. J*sus. Get a f*cking life.	1
16248	2adc0c27c1165eef	To whoever said most people in arab are homosexuals or other bad things: Go to D**n F***in' Hell you motherf***in' d**n son of a d**n f***ing c**t of a b**ch.	1
21977	39f43b3a2e2f45ee	Go away and stop bothering me i do not want your slimy s**t all over my page, so why do not you f**k off and leave the rest of us to celebrate the fact we are all going to DIE??? YAYYYYYY grinning face with smiling eyes	1
42504	7174f61591c8be78	Lia! You lie. Nyanpire might not be a boy,but you unregistered user,are one big B**ch!	1

	id	comment_text	toxic
201	007f1839ada915e6	Your blatant POV pushing Neither of you guys has made any contribution to this Italian history article other than to shove your unhistorical unconstructive modern POV in my face . This is a HISTORY article . HISTORY . Have you heard of that ? This is the reason why so many people get pissed off about the pedantry and idiocy and triviality of Wikipedia . Jesus . Get a fucking life .	1
16248	2adc0c27c1165eef	To whoever said most people in arab are homosexuals or other bad things : Go to Damn Fuckin ' Hell you motherfuckin ' damn son of a damn fucking cunt of a bitch .	1
21977	39f43b3a2e2f45ee	Go away and stop bothering me i do not want your slimy shit all over my page , so why do not you fuck off and leave the rest of us to celebrate the fact we are all going to DIE ? ? ? YAYYYYYY grinning face with smiling eyes	1
42504	7174f61591c8be78	Lia ! You lie . Nyanpire might not be a boy , but you unregistered user , are one big Bitch !	1

- Suppression des espaces multiples.

```
def remove_space_multiple(df):
    """Remove space multiple"""
    for i, row in df.iterrows():
        text = str(row["comment_text"]).strip()
        df.loc[i, "comment_text"] = ' '.join(text.split())
    return df
```

Sommaire

FullTokenizer

- Récupération de l'objet de la classe *FullTokenizer* qui contient la méthode de *tokenisation* et l'encodeur de token en *id*.

  Paramètres : bert_path : str

  Chemin vers Bert Saved Model

  Retour : tokenizer : Objet bert.tokenization.FullTokenizer

  Tokenizer bert

```python
def get_tokenizer(bert_path=BERT_PATH_SAVEDMODEL):
    """Obtenez le tokenizer pour une couche BERT."""
    bert_layer = tf.saved_model.load(bert_path)
    bert_layer = hub.KerasLayer(bert_layer, trainable=False)
    vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
    cased = bert_layer.resolved_object.do_lower_case.numpy()
    tf.gfile = tf.io.gfile  # bert.tokenization.load_vocab dans tokenizer
    tokenizer = bert.tokenization.FullTokenizer(vocab_file, cased)

    return tokenizer

tokenizer = get_tokenizer()
```

Encodage

- Convertir chaque token en id avec ajout de tokens spéciaux.

Paramètres : sentence : str - Texte à encoder

max_seq_length : int - Longueur maximale de la séquence encodée

tokenizer : Objet bert.tokenization.FullTokenizer - Tokenizer bert

Retour : input_ids : list - Liste contenant ids des tokens

input_mask : list - Liste contenant 1 pour les tokens et 0 pour le rembourrage

segment_ids : list - Liste de zéros

```
def process_sentence(sentence, max_seq_length=SEQUENCE_LENGTH, tokenizer=tokenizer):
    """Convertit la phrase sous la forme ['input_word_ids', 'input_mask', 'segment_ids']."""
    # Tokenize, et tronque à max_seq_length si necessaire.
    tokens = tokenizer.tokenize(str(sentence))
    if len(tokens) > max_seq_length - 2:
        tokens = tokens[:(max_seq_length - 2)]

    # Convertir les tokens de la phrase en IDs
    input_ids = tokenizer.convert_tokens_to_ids(["[CLS]"] + tokens + ["[SEP]"])

    # 1 pour les vrais tokens et un 0 pour les tokens de rembourrage.
    input_mask = [1] * len(input_ids)

    # Compléter par des zéros si la séquence est inférieur à max_seq_length
    pad_length = max_seq_length - len(input_ids)
    input_ids.extend([0] * pad_length)
    input_mask.extend([0] * pad_length)

    # Nous avons un seul segment d'entrée
    segment_ids = [0] * max_seq_length

    return (input_ids, input_mask, segment_ids)
```

Représentation vectorielle dense du texte

- Exemple de sortie après encodage du texte par m-Bert.

```
df_test.head(2)
```

	id	lang	input_word_ids	input_mask	all_segment_id	comment_text
0	0	tr	(101, 17376, 14516, 19165, 56324, 10116, 24542, 91727, 10186, 119, 26937, 11772, 10561, 56324, 10116, 82867, 10713, 32720, 42702, 16334, 19343, 61716, 18330, 119, 102884, 10917, 172, 78653, 12683, 10147, 119, 44798, 82350, 14434, 30471, 10126, 60906, 23760, 119, 152, 28217, 55743, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	(0, 0)	Doctor Who adlı viki başlığına 12. doctor olarak bir viki yazarı kendi adını eklemiştir. Şahsen düzelttim. Onaylarsanız sevinirim. Occipital
1	1	ru	(101, 511, 53204, 36689, 44504, 117, 11279, 572, 45899, 10375, 88504, 27557, 91765, 96195, 103185, 11258, 67409, 543, 78237, 44178, 50121, 10593, 119, 33463, 12709, 20064, 16417, 81350, 543, 12259, 22366, 10122, 48680, 26380, 22504, 14708, 11602, 38886, 10351, 10197, 118, 10244, 100, 31816, 10405, 117, 53300, 17257, 44030, 15356, 119, 11480, 10521, 11815, 35818, 543, 45184, 91258, 67927, 31870, 551, 10227, 16806, 10593, 50121, 10593, 524, 86310, 16417, 81350, 549, 70723, 10593, 10375, 23853, 551, 10227, 16806, 10593, 50121, 10593, 524, 86310, 16417, 81350, 543, 12259, 119, 18464, 10205, 95434, 22692, 117, 10791, 19558, 58099, 110530, 76021, 107960, 557, 31751, 18235, 10122, 29139, 75529, 511, 16027, 131, 525, 26313, 33930, 45030, 136, 14028, 10375, 23807, 29678, 69300, 17257, 44030, 14762, 117, 45899, 66799, 65086, 10375, 10344, 102)	(1, 1)	(0, 0)	Вполне возможно, но я пока не вижу необходимости выделять материал в отдельную статью. Если про правосудие в СССР будет написано хотя бы килобайт 20-30 — тогда да, следует разделить. Пока же мы в итоге получим одну куцую статью Правосудие и другую не менее куцую статью Правосудие в СССР. Мне кажется, что этот вопрос вполне разумно решать на основе правил ВП:Размер статей? которые не предписывают разделения, пока размер статьи не достигнет хотя бы 50 тыс. знаков.

Modèle multilingue Bert

- Construire l'architecture du modèle multilingue bert

Paramètres : max_seq_len : int

Utiliser pour la forme des entrées du modèle.

trainable_bert : bool

Entraîner la partie représentation ou pas.

Retour : _ : tf.keras.Model

Modèle *bert*

```
def multilingual_bert_model(max_seq_length=SEQUENCE_LENGTH, trainable_bert=True):
    """Construit et retourne un modèle Bert multilingue"""
    input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_word_ids")
    input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input_mask")
    segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="all_segment_id")

    bert_layer = tf.saved_model.load(BERT_GCS_PATH_SAVEDMODEL)
    bert_layer = hub.KerasLayer(bert_layer, trainable=trainable_bert)

    pooled_output, _ = bert_layer([input_word_ids, input_mask, segment_ids])
    output = tf.keras.layers.Dense(32, activation='relu')(pooled_output)
    output = tf.keras.layers.Dense(1, activation='sigmoid', name='labels')(output)

    return tf.keras.Model(inputs={'input_word_ids': input_word_ids,'input_mask': input_mask,
                                  'all_segment_id': segment_ids},outputs=output)
```

TPU

- Détection TPUs ou GPUs

```
# Detect hardware, return appropriate distribution strategy
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None

if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
else:
    # Default distribution strategy in Tensorflow. Works on CPU and single GPU.
    strategy = tf.distribute.get_strategy()

print("REPLICAS: ", strategy.num_replicas_in_sync)
```

```
Running on TPU  grpc://10.0.0.2:8470
REPLICAS:  8
```

- Chargement et compilation du modèle sur TPU.

```
with strategy.scope():
    multilingual_bert = multilingual_bert_model()
    # Compile le modèle.
    multilingual_bert.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                              optimizer=tf.keras.optimizers.Adam(lr=1e-4),
                              metrics=[tf.keras.metrics.AUC()])
```

Training and predict

- Notre stratégie d'entraînement est d'adaptée au dataset en anglais puis au dataset de validation multilingue.

```
# Train on dataset.
multilingual_bert.fit(english_train_dataset, steps_per_epoch=4000, epochs=2, verbose=1,
                      validation_data=nonenglish_val_datasets['Combined'], validation_steps=100)

multilingual_bert.fit(nonenglish_val_datasets['Combined'], steps_per_epoch=100, epochs=2)

# Prédictions
probabilities = np.squeeze(multilingual_bert.predict(test_sentences_dataset))
```

- Format du contenu du fichier soumission.

```
Generating submission file...
id,toxic
0,0.083437
1,0.128149
2,0.256390
3,0.082323
4,0.005595
5,0.358424
```

Encodage AutoTokenizer

- ***AutoTokenizer*** une classe de tokeniseur générique en fonction du modèle. *XLM-Roberta*, MODEL = 'jplu/tf-xlm-roberta-large'.

```
def get_tokenizer(MODEL):
    """Obtenez le tokenizer pour XLM-Roberta"""
    tokenizer = AutoTokenizer.from_pretrained(MODEL)

    return tokenizer

tokenizer = get_tokenizer(MODEL)
```

- Conversion des tokens en ids. Méthode *encode_plus* du tokeniseur.

```
def encode(texts, tokenizer, maxlen=512):
    enc_di = tokenizer.batch_encode_plus(
        texts,
        return_attention_masks=False,
        return_token_type_ids=False,
        pad_to_max_length=True,
        max_length=maxlen
    )
    return np.array(enc_di['input_ids'])
```

Représentation vectorielle dense du texte

- Encodage

```
x_train = encode(train.comment_text.values, tokenizer, maxlen=SEQUENCE_LENGTH)
x_valid = encode(valid.comment_text.values, tokenizer, maxlen=SEQUENCE_LENGTH)
x_test = encode(test.comment_text.values, tokenizer, maxlen=SEQUENCE_LENGTH)

y_train = train.toxic.values
y_valid = values
```

- Exemple de sortie après encodage du texte par XLMRobertaTokenizer.

```
x_valid[:1]
```

```
array([[     0,   3224,  54367,    300, 172297,  47612,    144, 161265,
             8,   3688,    236,      6,      5,   1818,    459,   4104,
         11035,   1651,    520,  10155,   1138,     22,     21,  91614,
           587,     11,    121, 217332,     42,    166,  22068,    113,
         13130, 129402, 178705,     31,  21959,    166,     23, 141933,
             5,  91958,     13,   3126,     73,   1039,  88684,  58744,
             4,   6053,      4,     95,   8090,     71,      5,    729,
         34605,    702,   1108,   2021,     15,  45962,     16,      2,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
             1,      1,      1,      1,      1,      1,      1,      1,
```

tf.data.Dataset

- L'utilisation efficace de la mémoire et TPUs avec l'api tf.data.Dataset

```
train_dataset = (
    tf.data.Dataset
    .from_tensor_slices((x_train, y_train))
    .repeat()
    .shuffle(2048)
    .batch(BATCH_SIZE)
    .prefetch(AUTO)
)

valid_dataset = (
    tf.data.Dataset
    .from_tensor_slices((x_valid, y_valid))
    .batch(BATCH_SIZE)
    .cache()
    .prefetch(AUTO)
)

test_dataset = (
    tf.data.Dataset
    .from_tensor_slices(x_test)
    .batch(BATCH_SIZE)
)
```

La classe AutoModel

- Construire le modèle grâce aux couches de l'architecture XLM-Roberta

```
def build_model(transformer, max_len=512):
    """
    Construire le modèle à partir des couches Roberta
    """
    input_word_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_word_ids")

    sequence_output = transformer(input_word_ids)[0]
    cls_token = sequence_output[:, 0, :]
    x = tf.keras.layers.Dropout(0.5)(cls_token)
    out = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=input_word_ids, outputs=out)
    model.compile(tf.keras.optimizers.Adam(lr=1e-5), loss='binary_crossentropy', metrics=[tf.ke
ras.metrics.AUC()])

    return model
```

- Load model into TPU

```
%%time
with strategy.scope():
    transformer_layer = TFAutoModel.from_pretrained(MODEL)
    model = build_model(transformer_layer, max_len=SEQUENCE_LENGTH)
```

Training

- Tout d'abord, nous entraînons notre modèle sur le dataset d'entrainement qui est entièrement en anglais.

```
Epoch 1/2
2873/2873 [==============================] - 1230s 428ms/step - auc: 0.9898 - loss: 0.1252 -
val_auc: 0.8980 - val_loss: 0.3206
Epoch 2/2
2873/2873 [==============================] - 1198s 417ms/step - auc: 0.9954 - loss: 0.0814 -
val_auc: 0.8975 - val_loss: 0.3283
```

- Puis, nous l'entraînons sur l'ensemble de validation, qui est beaucoup plus petit mais contient un mélange de différentes langues.

```
n_steps2 = x_valid.shape[0] // BATCH_SIZE
model.fit(valid_dataset.repeat(), steps_per_epoch=n_steps2, epochs=EPOCHS)
```

```
Epoch 1/2
62/62 [==============================] - 25s 409ms/step - auc: 0.9247 - loss: 0.2345
Epoch 2/2
62/62 [==============================] - 25s 407ms/step - auc: 0.9627 - loss: 0.1687
```

Sommaire

Résultats et discussions

- *AUC* public modèle *jplu/tf-xlm-roberta-large* est de : ***0.9238***
- Résultat obtenu au bout juste de 6 soumissions, nombre faible comparé au trois premiers de cette compétition respectivement (385, 295, 354).
- Le résultat jugé moyen car dut à une mauvaise organisation (méconnaissance de la plateforme kaggle, utilisation entière de la donnée, preprocessing de base, etc).
- La phase de preprocessing nécessitait davantage d'opérations avancées.
- Une connaissance accrue du fonctionnement du modèle *XLM-Roberta* aurait été un plus ainsi qu'une meilleure construction du modèle (partie classifieur).

Sommaire

Conclusion

- Expérience enrichissante d'échange avec la communauté Kaggle.
- Fondamentaux des compétitions Kaggle acquis et contraintes.
- Utilisation de deux hub importants de modèles NLU et NLG : ***transformers*** de HuggingFace et ***TFHub*** de TensorFlow.
- Utilisation des TPUs : accélérer le temps d'exécution d'entraînement.
- Construire efficacement les modèles multilingues - conversation IA
- Appropriation des opérations de pré-traitement de base
- Stratégie d'entrainement des modèles multilingues

Perspectives

- Intégrer davantage les lois de distributions dans la modélisation entre les différents dataset.
- Utiliser la technique des n-grammes et des features linguistiques lors de la phase de preprocessing de la data.