# Translation Source Dialect Identification Documentație

**Fărcășanu Tudor Andrei**

**Grupa 363**

## 1. Preprocesarea datelor

Pentru soluțiile finale, am tradus dataset-ul competiției în engleză, folosind serviciul Document Translation din oferta Azure Cognitive Services. Pentru a folosi etichetarea limbilor prezente în setul de antrenare, l-am împărțit în documente csv separate. De asemenea am aplicat o funcție pentru a curăța datele (am înlăturat linkurile, \n, \t):

```
import numpy as np

import pandas as pd

import re

import string

def clean_text_tran(text):

    # remove links

    text = re.sub('https?:\/\/\S+', '', text)

    #special for iloc[32827]

    text = text.replace('\\t', '')

    text = text.replace('_blank', '')

    # remove next line

    text = re.sub(r'[^ \w\.]', '', text)


    return text

clean_test = pd.read_csv("test_data.csv")

clean_test['Text'] = clean_test.text.apply(lambda x: clean_text_tran(x))


clean_test.drop(columns=["text"]).to_csv("clean_test.csv", index=True)


train_df_unspl = pd.read_csv("train_data.csv")

train_df_unspl['Text'] = train_df_unspl.text.apply(lambda x: clean_text_tran(x))
```

```
train_df_unspl.loc[train_df_unspl.language=="dansk"].drop(columns=["language","text","label"]).to_csv("danish.csv", index=True)

train_df_unspl.loc[train_df_unspl.language=="Deutsch"].drop(columns=["language","text","label"]).to_csv("german.csv", index=True)

train_df_unspl.loc[train_df_unspl.language=="español"].drop(columns=["language","text","label"]).to_csv("spanish.csv", index=True)

train_df_unspl.loc[train_df_unspl.language=="italiano"].drop(columns=["language","text","label"]).to_csv("italian.csv", index=True)

train_df_unspl.loc[train_df_unspl.language=="Nederlands"].drop(columns=["language","text","label"]).to_csv("dutch.csv", index=True)

print(train_df_unspl.Text)
```

Am apelat apoi endpoint-ul furnizat de Azure pentru a traduce pe rând din fiecare limbă în engleză, pentru setul de testare nespecificând limba sursă:

```
from azure.core.credentials import AzureKeyCredential

from azure.ai.translation.document import DocumentTranslationClient


endpoint = "https://datasetrans.cognitiveservices.azure.com/"

credential = AzureKeyCredential("<insertkey>")

source_container_sas_url_en =
"https://translationdat.blob.core.windows.net/sursa..(truncated)"

target_container_sas_url_es =
"https://translationdat.blob.core.windows.net/rezult..(truncated)"


document_translation_client = DocumentTranslationClient(endpoint, credential)


poller = document_translation_client.begin_translation(source_container_sas_url_en,
target_container_sas_url_es, target_language="en", source_language="<insert
language>")


result = poller.result()


print(f"Status: {poller.status()}")

print(f"Created on: {poller.details.created_on}")

print(f"Last updated on: {poller.details.last_updated_on}")
```

```
print(f"Total number of translations on documents:
{poller.details.documents_total_count}")


print("\nOf total documents...")
print(f"{poller.details.documents_failed_count} failed")
print(f"{poller.details.documents_succeeded_count} succeeded")


for document in result:
   print(f"Document ID: {document.id}")
   print(f"Document status: {document.status}")
   if document.status == "Succeeded":
       print(f"Source document location: {document.source_document_url}")
       print(f"Translated document location: {document.translated_document_url}")
       print(f"Translated to language: {document.translated_to}\n")
   else:
       print(f"Error Code: {document.error.code}, Message:
{document.error.message}\n")
```

## 2. Submisie folosind KNN

Pentru una dintre submisii, am folosit metoda celor mai apropiați k vecini. Înainte de a aplica varianta algoritmului din librăria sklearn, am mai aplicat un pas de preprocesare, în care scot majusculele, semnele de punctuație, cuvintele ce conțin numere dar și stopwords, din librăria spacy:

```
import re
import string
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from spacy.lang.en.stop_words import STOP_WORDS as en_stop
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

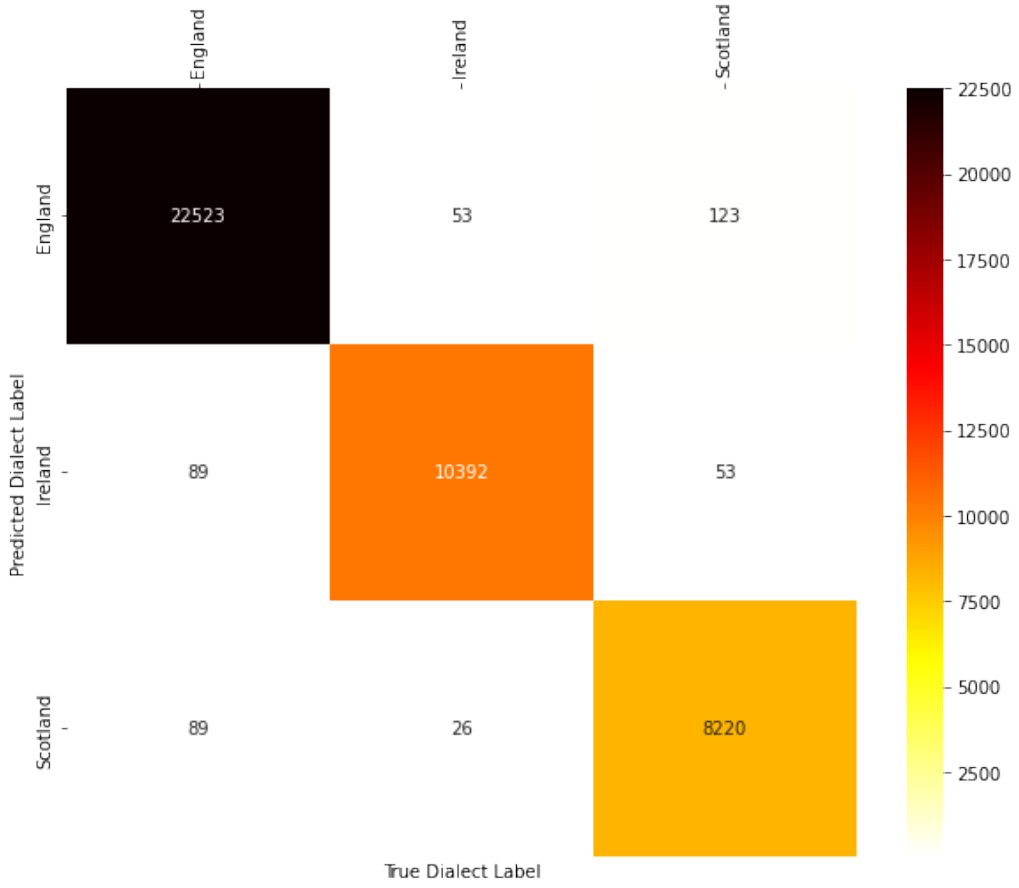
```
from sklearn import svm

import seaborn as sns

import pandas as pd



data_path = '../input/unibuccomptranslated'

train_df_unspl =
pd.read_csv('../input/unibuccomptranslated/translated_train_data.csv',quotechar="^"
,index_col="id_ref")

test_df =
pd.read_csv('../input/unibuccomptranslated/clean_test_fixed.csv',quotechar="^",inde
x_col="id_ref")



def clean_text(text,sw):

    # to lower case

    text = text.lower()

    # remove punctuation

    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)

    # remove words containing numbers

    text = re.sub('\w*\d\w*', '', text) #maybe not? la laborator - poate au sens?

    #remove stopwords

    words = [word for word in text.split() if word.lower() not in sw]

    text=" ".join(words)



    return text



train_df_unspl=train_df_unspl.dropna()

train_df_unspl['Text'] = train_df_unspl.Translated.apply(lambda x:
clean_text(x,list(en_stop)))

test_df.loc[7582].Text=''

test_df['PText'] = test_df.Text.apply(lambda x: clean_text(x,list(en_stop)))
```

Caracteristicile pe care modelul a fost antrenat sunt reprezentările vectoriale ale textelor prelucrate ca mai sus, obținute prin aplicare CountVectorizer() din librăria sklearn, cu opțiunea strip_accents='unicode' care aplică și normalizarea de tip NFKD.

Modelul "celor mai apropiați k vecini„ este un algoritm neparametrizat de învățare supervizată, care are ca singur hiperparametru k, numărul de vecini luați în considerare. Am ales hiperparametrul k experimental, încercând valori impare în intervalul [1,15], dar am obținut cea mai bună acuratețe cu k=1, fiindcă după traducerea setului de date propozițiile care original erau identice aveau reprezentări foarte asemănătoare:

```
Propozitia testata: mr president subject amendments group fully supports motion resolution tab
eference religion constitution union different religions religious pluralism people union reli

Propozitia din train: mr president amendments group fully supports motion resolution presented
mmission opposed wish refer religion constitution union diverse religions religious pluralism
0.989295164782295

Propozitia testata: madam president concerns vote fiori report brussels parliament worst tempo

Propozitia din train: madam president refer fiori report vote partsession brussels parliament
0.9904979552561943

Propozitia testata: speak foreign minister mr santer spoken point skip meeting turkish foreign

Propozitia din train: time mr spring meeting turkish foreign minister dublin assure house conc
0.9878518162136156

Propozitia testata: mr president areas exert influence change regime drugs oil tourism feature

Propozitia din train: mr president things need influence burma order change regime drugs oil t
0.9903765187056418

Propozitia testata: british conservatives support view emergencies member states wish european

Propozitia din train: british conservatives support idea event major emergencies member states
0.9898953446409239
```

Mai jos am obținut acuratețea folosind StratifiedKFold cu un număr de 5 fold-uri:



| Fold | Acuratețe |
|---|---|
| 1 | 0.989295164782295 |
| 2 | 0.9904979552561943 |
| 3 | 0.9878518162136156 |
| 4 | 0.9903765187056418 |
| 5 | 0.9898953446409239 |
| Medie | 0.9895833599197 |

Antrenarea acestui model a durat în total aproximativ 336 secunde.

Acest model a obținut un scor public de 0.46, fiindcă în datele de test propozițiile nu erau repetate în mai multe limbi, la fel ca în setul de date de antrenare.

### 3. Submisie folosind RoBERTa

Modelul RoBERTa (Robustly Optimized BERT Pretraining Approach) este o formă îmbunătățită a modelului BERT (Bidirectional Encoder Representations from Transformers), care a fost pre-antrenat doar pentru modelarea mascată a limbajului (MLM – Masked Language Modeling) și care în loc să mascheze doar în stadiul de pregătire a datelor, introduce generarea măștii în timpul antrenării, astfel încât modelul să nu vadă de mai multe ori aceeași mască pentru aceeași propoziție în timpul antrenării.

Pentru această submisie am folosit tokenizatorul asociat modelului, care folosește o abordare de encodare la nivel de perechi de octeți. Astfel, caracteristicile folosite pentru antrenare sunt 85% din toate datele prelucrate cum este precizat în secțiunea 1, împărțite stratificat la întâmplare în funcție de etichetă, cu starea generatorului 42, cărora a fost aplicat tokenizatorul și trunchiate la 512 reprezentări.

Am preluat modelul pre-antrenat RoBERTa-large, iar pentru cele mai bune predicții am folosit următorii parametrii:

- Număr maxim de cicluri de antrenare: 50
- Starea generatorului de numere pentru antrenare: 12345
- Lungimea maximă a reprezentărilor textului luate în calcul: 512 (maximul care poate fi folosit)
- Dimensiunea loturilor de antrenare: 128 (16 * numărul de nuclee TPU disponibile)
- Optimizatori:
    1. Adam (Adaptive moment estimation):
        - Rata de învățare: $10^{-5}$
    2. Adamw (Adaptive moment estimation with weight decay)
        - Rata de învățare: $10^{-5}$

    - Rata de reducere a greutăților: $10^{-5}$
- Funcții de activare:
    1. Softmax

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,\text{K}$$

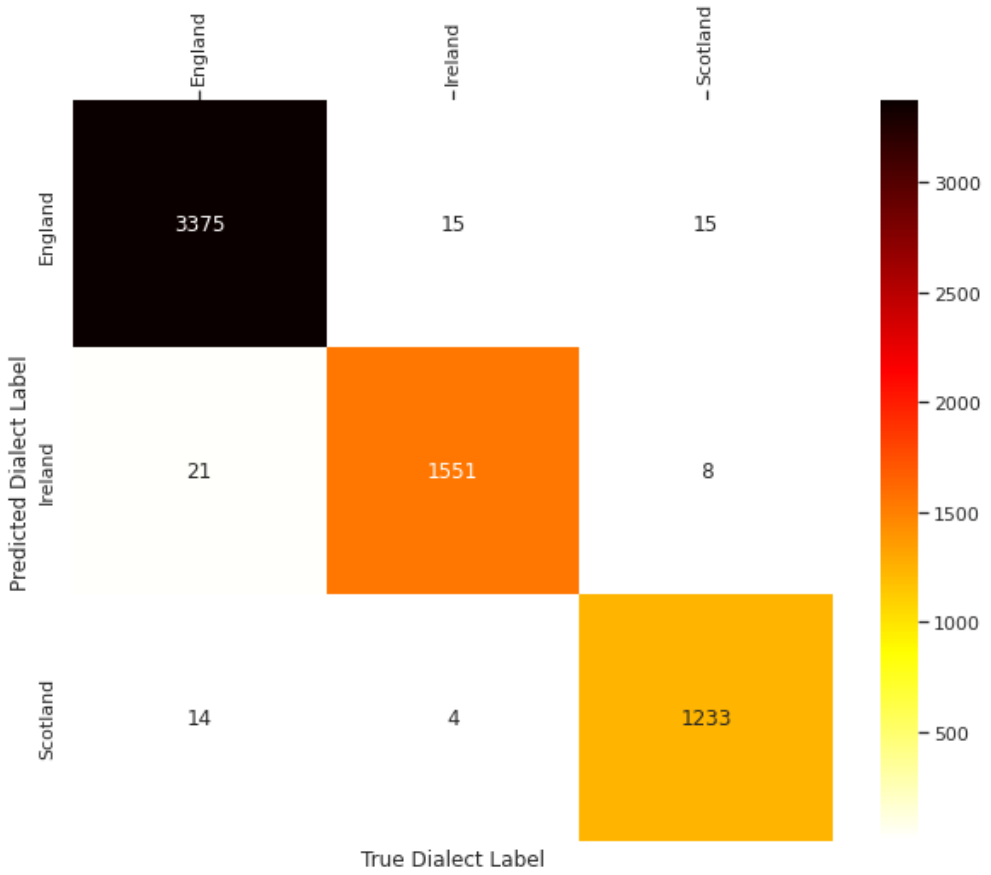
    2. Sigmoid (deși nu este potrivită pentru o problemă de clasificare pe mai multe clase, a dat rezultate similare cu cele obținute folosind Softmax)

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}}$$

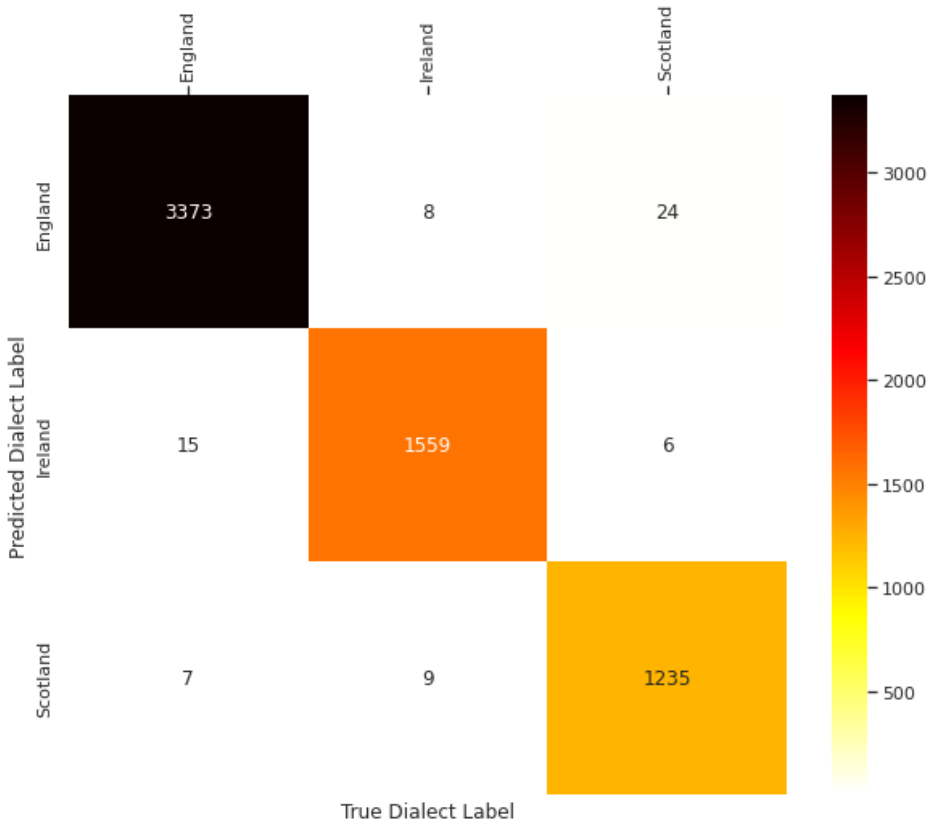
Antrenarea unui astfel de model poate dura între 4000 și 6600 de secunde, în funcție de parametrii folosiți, într-o sesiune de tip TPU Kaggle (16 GB RAM, 8 nuclee TPU v3).

Predicțiile selectate au obținut următoarele scoruri publice:

1. Optimizator AdamW, rata de reducere a greutăților $10^{-5}$, funcție de activare sigmoid: 0.76028



2. Optimizator Adam, rata de învățare $10^{-5}$, funcție activare softmax: 0.74837




## 4. Analiza clasificatorilor încercați

Deși soluția finală se bazează pe aplicarea RoBERTa, un model antrenat pe texte în engleză, pe textele traduse, pentru primele ~20 submisii foloseam XLM-RoBERTa, o variantă antrenată pe un corpus multi-lingual, aplicată pe textele originale, cu acuratete publică de până la 0.73