UE10 Extraction et programmation statistique de l'information

LAI Khang Duy
Mazen TEBIB

Université Paris-Saclay
Faculté des sciences d'Orsay

université
PARIS-SACLAY

23-03-2023

Contents

1 Data exploration **2**

2 Data preprocessing **2**

2.1 Clean raw texts 3

2.2 TF-IDF Vectorizer 3

2.3 Encoding labels 4

2.4 Extracting features 4

2.4.1 Lexical features 4

2.4.2 Syntatic features 4

2.4.3 Syntatic features 4

2.4.4 Best feature to add 5

3 Models **5**

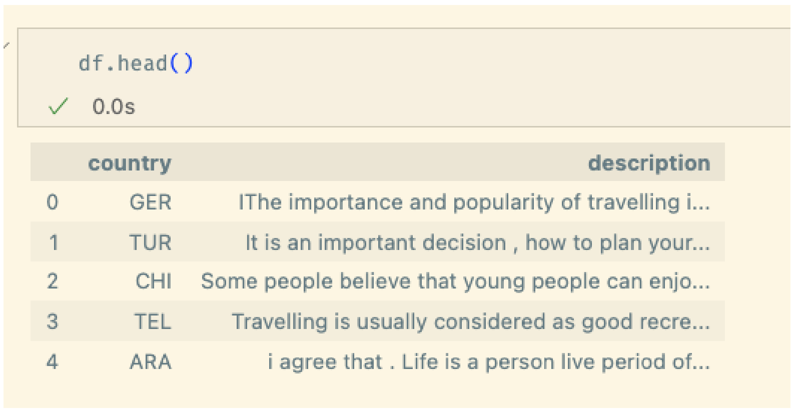
3.1 Model selection 5

3.2 Cascade method 7

4 Hyperparameter tuning **9**

1 Data exploration

First, we want to see which exactly in the corpus in order to further do the data preprocessing. As you can see in the image of Raw data set, we have 2 columns in the corpus, `country` and the `description`. `country` is functioned as the label `y`, whereas `description` is the data itself.

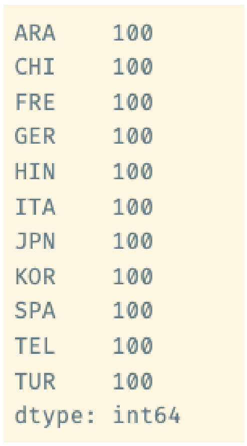


```
df.head()
✓ 0.0s
```

	country	description
0	GER	IThe importance and popularity of travelling i...
1	TUR	It is an important decision , how to plan your...
2	CHI	Some people believe that young people can enjo...
3	TEL	Travelling is usually considered as good recre...
4	ARA	i agree that . Life is a person live period of...

Figure 1: Raw data set

We have 10 labels stand for 10 different native languages. The number of data points in each label is equal.



```
ARA    100
CHI    100
FRE    100
GER    100
HIN    100
ITA    100
JPN    100
KOR    100
SPA    100
TEL    100
TUR    100
dtype: int64
```

Figure 2: Number of data point in each class

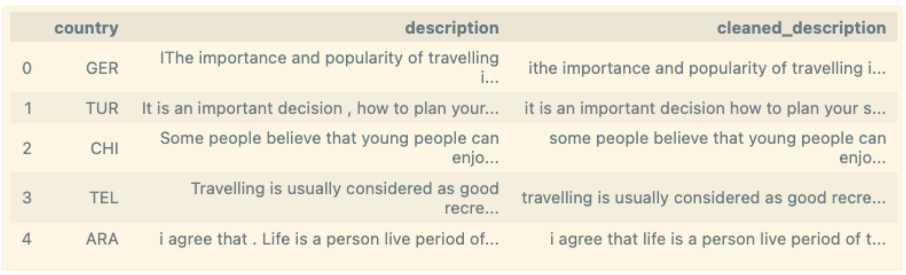
2 Data preprocessing

In this step, we apply a few data preprocessing techniques before feed the processed data to the model. These steps include cleaning the texts, using IF-IDF to vertorized it, encode the labels in the column `country` and Extract important features.

2.1 Clean raw texts

Normally, this process includes also removing stop words, lemmatization and stemming. However, due to the fact that removing such thing might remove very important features when the people who use English as second language could do such mistake too, so in this case we are not going to include it it. We only include:

- Lowering all texts
- Remove numbers
- Remove punctuations
- Remove special characters



	country	description	cleaned_description
0	GER	lThe importance and popularity of travelling i...	ithe importance and popularity of travelling i...
1	TUR	It is an important decision , how to plan your...	it is an important decision how to plan your s...
2	CHI	Some people believe that young people can enjo...	some people believe that young people can enjo...
3	TEL	Travelling is usually considered as good recre...	travelling is usually considered as good recre...
4	ARA	i agree that . Life is a person live period of...	i agree that life is a person live period of t...

Figure 3: After cleaning corpus

2.2 TF-IDF Vectorizer

Split the data into test set and train set before use tf-idf to vectorize it. If you learn tf-idf also on the test set you will have data leakage. In example, you won't have out-of vocabulary words in inference on the test set, what might happen on the real world.

```
#would be better if u split data before then fit then transform
tf = TfidfVectorizer(stop_words='english', ngram_range=(1,2))
tf_Xtrain = tf.fit_transform(X_train)
tf_Xtest = tf.transform(X_test)
```

We have tried to adjust the `ngram_range` between `ngram_range=(2,3)` and `ngram_range =(1,2)`. The best one is `(1,2)`.

2.3 Encoding labels

We cannot use the direct label to feed the model. Before feeding the model, it is necessary to encode it into the form of numbers. We added a new column in the table. `country` is the label and `y_country` is the encoded one, from 0 to 9 according to 10 labels.

country	y_country

2.4 Extracting features

we thought about adding features that represents the writing style.

2.4.1 Lexical features

- total number of words
- total number of characters
- total number of alphabetic characters
- total number of upper-case characters
- total number of syllables
- average word length
- average sentence length by characters
- average sentence length by words
- ratio of unique words

2.4.2 Syntatic features

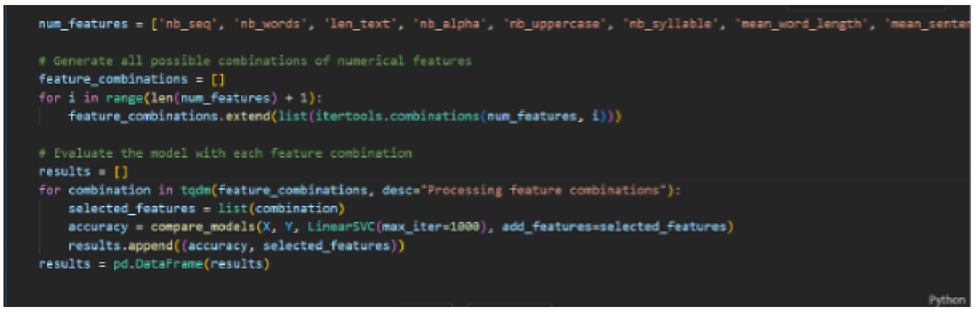
- frequency of special characters
- frequency of punctuations
- frequency of functional words
- part of speech tags

2.4.3 Syntatic features

- number of sentences

2.4.4 Best feature to add

Then to get which is the best features to add



```python
num_features = ['nb_seq', 'nb_words', 'len_text', 'nb_alpha', 'nb_uppercase', 'nb_syllable', 'mean_word_length', 'mean_senter

# Generate all possible combinations of numerical features
feature_combinations = []
for i in range(len(num_features) + 1):
    feature_combinations.extend(list(itertools.combinations(num_features, i)))

# Evaluate the model with each feature combination
results = []
for combination in tqdm(feature_combinations, desc="Processing feature combinations"):
    selected_features = list(combination)
    accuracy = compare_models(X, Y, LinearSVC(max_iter=1000), add_features=selected_features)
    results.append((accuracy, selected_features))
results = pd.DataFrame(results)
```

Python

Figure 4: Best feature to add

We created an array with all possible combinations of features and we looked at the score of each combination to find the best features which are with 0.74 accuracy score.

```
[['nb_seq', 'nb_words', 'len_text', 'nb_uppercase', '
    mean_sentence_length_byWords', 'tot_diff_words']]
```

3 Models

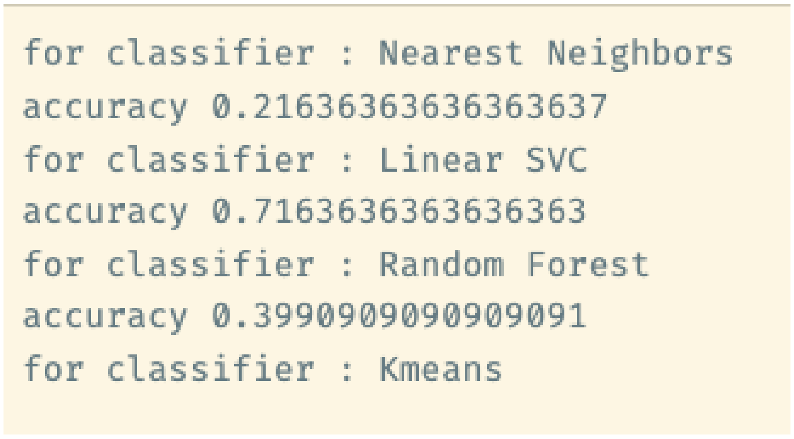
We have tried several method for the model of this subject. First, we tried to determine which is the bes classification model. Second, we apply the hierachy method (cascade method), in order to see if there are any improvements.

3.1 Model selection

In order to find the best model that work with the corpus, we must run several models and then select the model with the best score overall.

The list of models that we used including:

- Nearest Neighbors
- Linear SVC
- Random Forest
- Kmeans
- GaussianNB

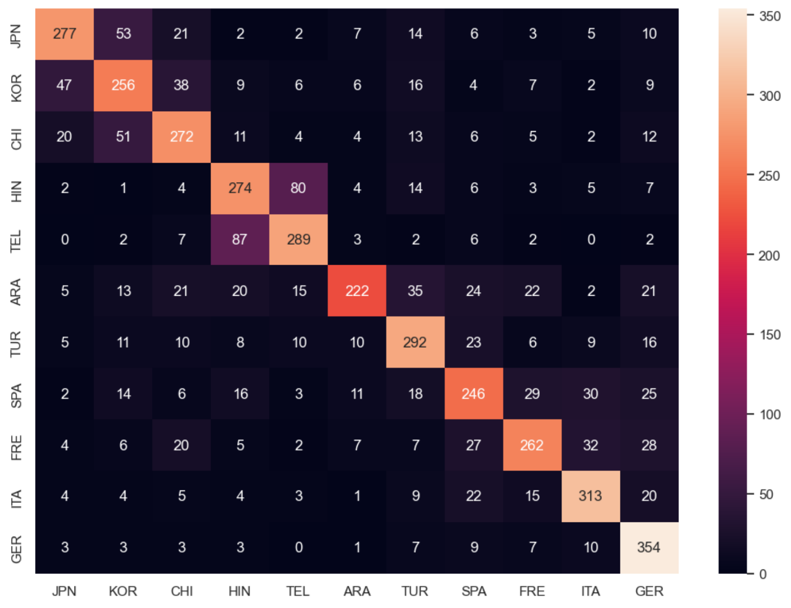


```
for classifier : Nearest Neighbors
accuracy 0.2163636363636637
for classifier : Linear SVC
accuracy 0.7163636363636363
for classifier : Random Forest
accuracy 0.3990909090909091
for classifier : Kmeans
```

Figure 5: Model selection results

With the LinearSVC, we have the best score overall so in the next cascade method I will use LinearSVC as our model.

Here is the confusion matrix of LinearSVC with the average accuracy of 70% if we classify language directly.

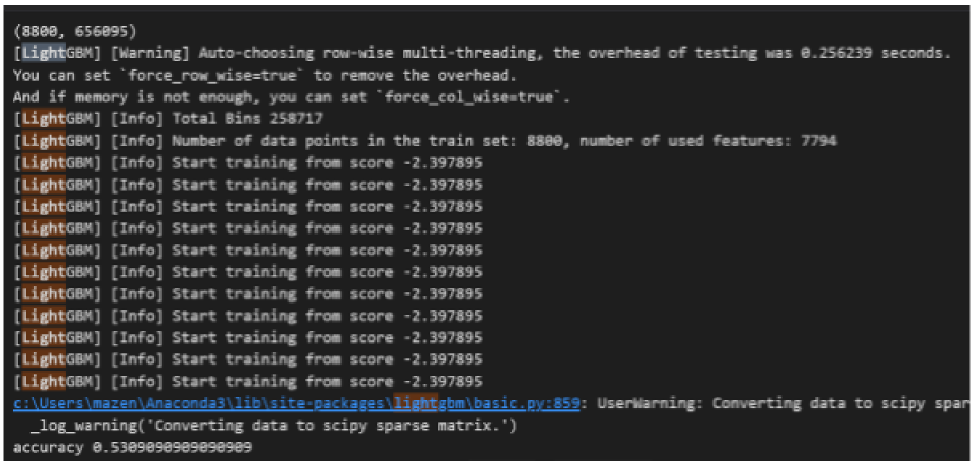


	JPN	KOR	CHI	HIN	TEL	ARA	TUR	SPA	FRE	ITA	GER
JPN	277	53	21	2	2	7	14	6	3	5	10
KOR	47	256	38	9	6	6	16	4	7	2	9
CHI	20	51	272	11	4	4	13	6	5	2	12
HIN	2	1	4	274	80	4	14	6	3	5	7
TEL	0	2	7	87	289	3	2	6	2	0	2
ARA	5	13	21	20	15	222	35	24	22	2	21
TUR	5	11	10	8	10	10	292	23	6	9	16
SPA	2	14	6	16	3	11	18	246	29	30	25
FRE	4	6	20	5	2	7	7	27	262	32	28
ITA	4	4	5	4	3	1	9	22	15	313	20
GER	3	3	3	3	0	1	7	9	7	10	354

Figure 6: LinearSVC confusion matrix

I re run the test with randomly train test split 4 times to take the average heat map confusion matrix.

We tried using lightgmb since it's known for being fast and reliable however we got an accuracy of 0.53.



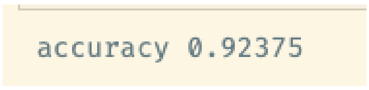
```
(8800, 656095)
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.256239 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 258717
[LightGBM] [Info] Number of data points in the train set: 8800, number of used features: 7794
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
[LightGBM] [Info] Start training from score -2.397895
c:\Users\mazen\Anaconda3\lib\site-packages\lightgbm\basic.py:859: UserWarning: Converting data to scipy spar
  _log_warning('Converting data to scipy sparse matrix.')
accuracy 0.5309090909090909
```

Figure 7: LightGBM

3.2 Cascade method

After we find the best model, we implement cascade method on the project. The idea is simple. We have 2 layers of models. First, we train the model so that it could well separate the language groups. We devided 10 languages into 4 language groups as below:

- Asian language group
- Roman language group
- Indian language group
- Other languages



```
accuracy 0.92375
```

Figure 8: Language group confusion matrix

We can achieve the confident of 93% with group separation. Below is the confusion matrix bettween language groups.

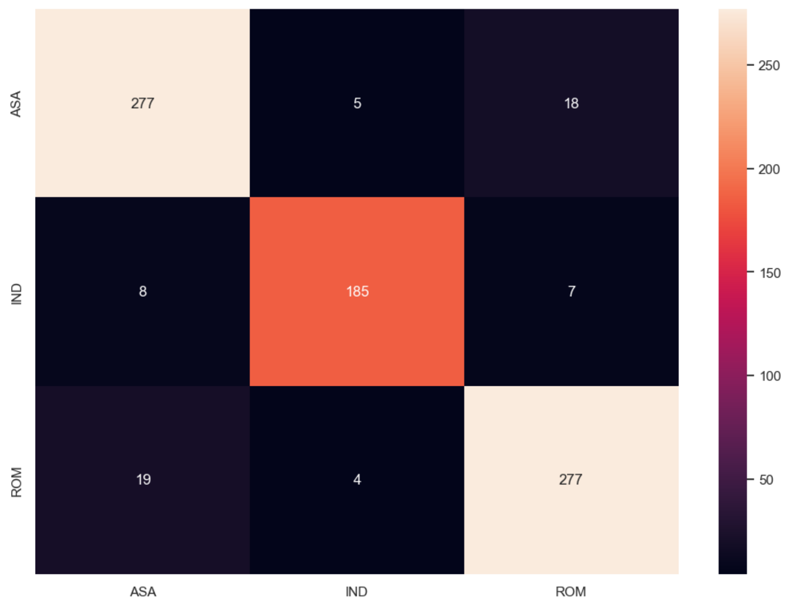


Figure 9: Language group confusion matrix

After that, which data point have the highest confident we will pass it in to the language specific layer. You can see the visualization of it as the diagram below.

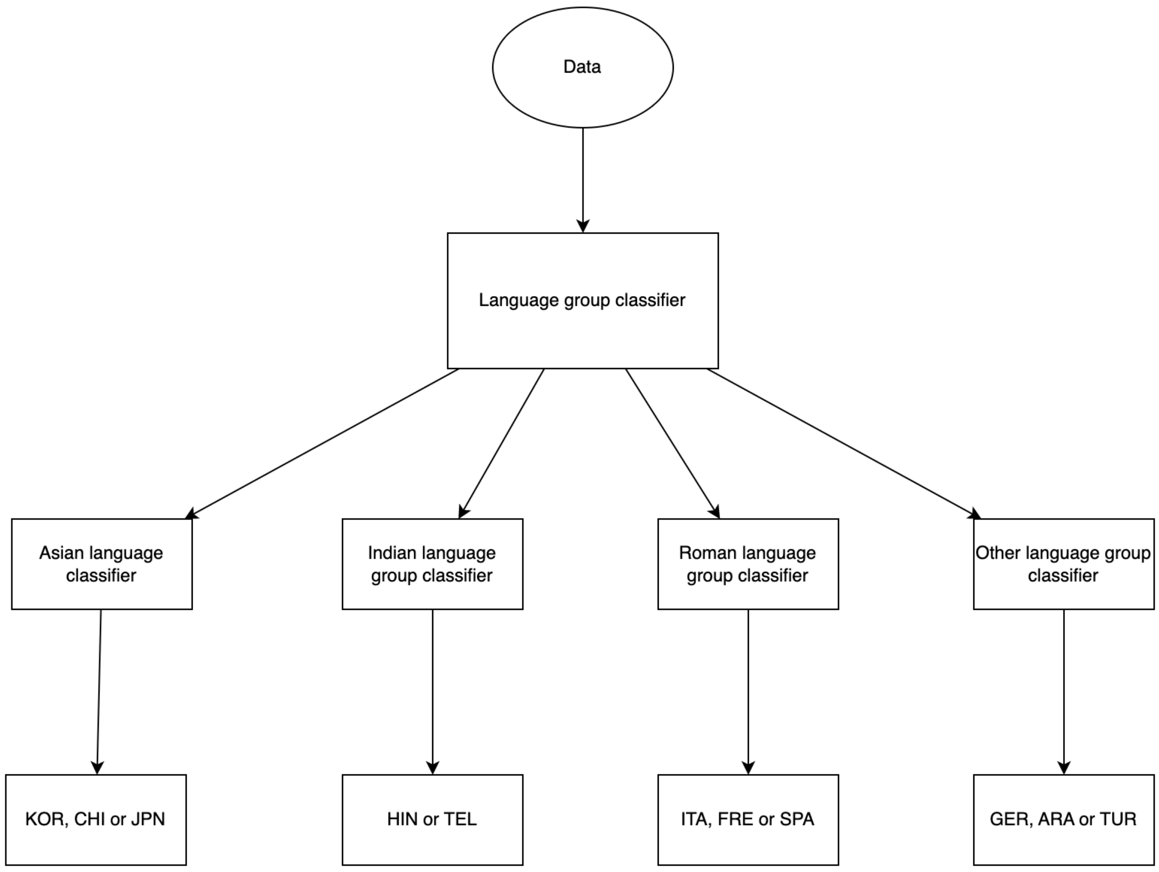
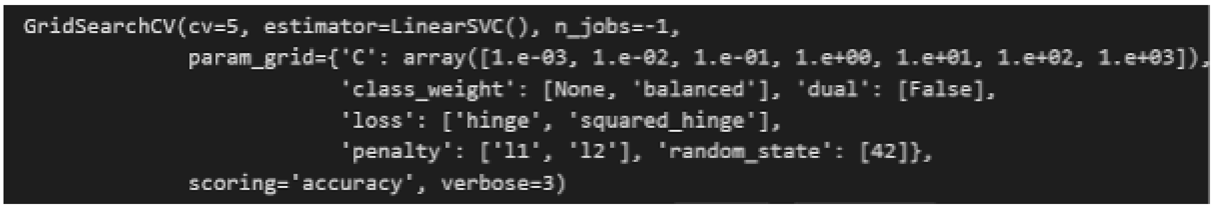

Data
Language group classifier
Asian language classifier
Indian language group classifier
Roman language group classifier
Other language group classifier
KOR, CHI or JPN
HIN or TEL
ITA, FRE or SPA
GER, ARA or TUR

Figure 10: Cascade diagram

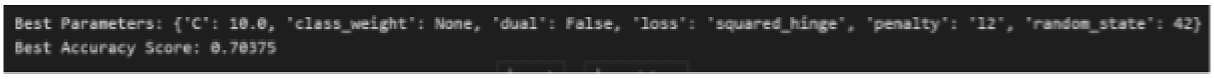
4 Hyperparameter tuning

Best accuracy yet feels like nothing changed when it comes to score.



```
GridSearchCV(cv=5, estimator=LinearSVC(), n_jobs=-1,
             param_grid={'C': array([1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03]),
                         'class_weight': [None, 'balanced'], 'dual': [False],
                         'loss': ['hinge', 'squared_hinge'],
                         'penalty': ['l1', 'l2'], 'random_state': [42]},
             scoring='accuracy', verbose=3)
```

Figure 11: Hyper 1



```
Best Parameters: {'C': 10.0, 'class_weight': None, 'dual': False, 'loss': 'squared_hinge', 'penalty': 'l2', 'random_state': 42}
Best Accuracy Score: 0.70375
```

Figure 12: Hyper 2

We can change the threshold of the language group layer and language specific layer until we have the best score.

We don't see a big improvement when it comes to score.

Tried we run it through calibrated classifier to prepare it for the Cascade method.

```
hierarchical_predict(X, group_clf, classifiers, threshold_group=0.8,
    threshold_specific=0.5):
```

the `threshold_group` and the `threshold_specific` varies from 0.5 to 0.8, and the impact on the last score is minor. So we keep the score as 0.5.