PARTICIPEZ À UNE COMPÉTITION KAGGLE

Evaluating language knowledge of ELLs students from grades 8-12

Mestapha Oumouni

P8 OC parcours IML
Mentor: Samir Tanfous
31/10/2022

Sommaire

- Présentation de la compétition
- Chargement des données et pré-processing
- Modèles de prédiction
- Conclusion

Objectif du projet

Kaggle: plateforme organisant des compétitions de Data Science

- ➢ Mission: participer à une compétition réelle et en cours
- ➢ Partager ses résultats avec la communauté

Présentation de la compétition

Vue générale de la compétition

Context:

- L'expression écrite fondamentale dans l'apprentissage d'une seconde langue
- La population d'apprenants de l'anglais est en croissance rapide, l'évaluation des compétences linguistiques est une tâche fatigante pour les enseignants.
- Les outils de l' évaluation existants sont moins efficaces
- Évaluation parfois biaisée au détriment de l'apprenant.

Mission:

Développer un modèle de ML afin d'évaluer avec précision les compétences linguistiques des apprenants de la langue anglaise.

Données:

Des essais de rédaction (en anglais) notés selon six mesures analytiques : cohésion, syntaxe, vocabulaire, phraséologie, grammaire et conventions.

Chargement des données et pré-processing

Aperçu des données

```
df_train.tail()
```

	text_id	full_text	cohesion	syntax	vocabulary	phraseology	grammar	conventions
3906	FFD29828A873	I believe using cellphones in class for educat...	2.5	3.0	3.0	3.5	2.5	2.5
3907	FFD9A83B0849	Working alone, students do not have to argue w...	4.0	4.0	4.0	4.0	3.5	3.0
3908	FFDC4011AC9C	"A problem is a chance for you to do your best...	2.5	3.0	3.0	3.0	3.5	3.0
3909	FFE16D704B16	Many people disagree with Albert Schweitzer's ...	4.0	4.5	4.5	4.0	4.5	4.5
3910	FFED00D6E0BD	Do you think that failure is the main thing fo...	3.5	2.5	3.5	3.0	3.0	3.5

- 3911 Essais de textes
- Six variables d'évaluation dont les notes vont de 1,0 à 5,0 par incréments de 0,5

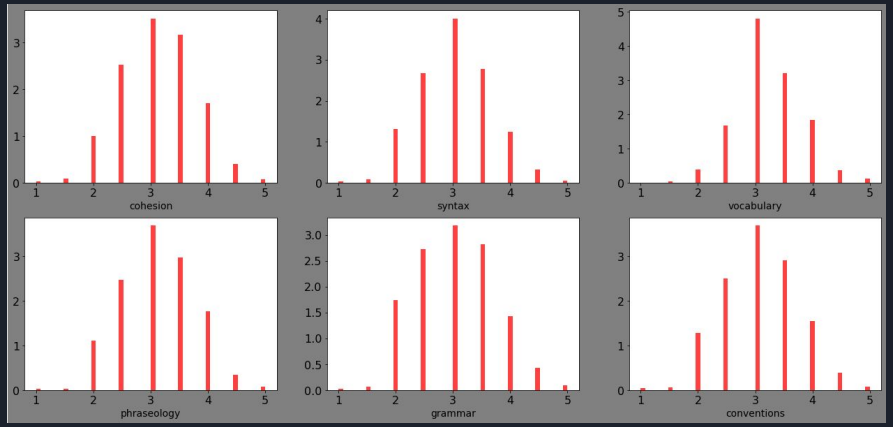
➢ Objectif: Prédire l'évaluation d'un test pour les six composantes:

➢ Métrique:

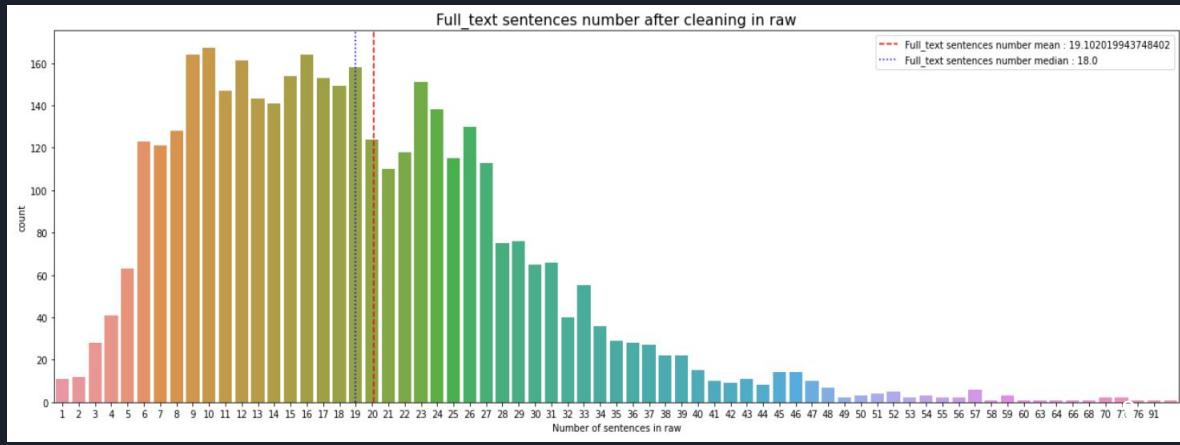
$$\frac{1}{6}\sum_{j=1}^{6}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{i,j}-\hat{y}_{i,j})^2}$$

Exploration des données

- Pas de valeurs manquantes
- Pas de valeurs manquantes
- 3910 de textes de test de writing en anglais
- Pas de corrélation importante entre les variables
- En moyenne le text contient 20 phrases



	cohesion	syntax	vocabulary	phraseology	grammar	conventions
cohesion	1.000000	0.695459	0.666151	0.690058	0.638689	0.666151
syntax	0.695459	1.000000	0.680562	0.725467	0.709525	0.700025
vocabulary	0.666151	0.680562	1.000000	0.735261	0.654852	0.664292
phraseology	0.690058	0.725467	0.735261	1.000000	0.719746	0.666842
grammar	0.638689	0.709525	0.654852	0.719746	1.000000	0.673301
conventions	0.666151	0.700025	0.664292	0.666842	0.673301	1.000000



MODÈLES DE PRÉDICTION

- TfidfVectorizer + SVR
- Bert pré-entraîné
  - avec extraction des features

TfidfVectorizer + SVR

```
vectorizer = TfidfVectorizer(max_df=0.98, min_df=2, max_features=10000)
X = vectorizer.fit_transform(X)
# split data
X_train, X_test, y_train, y_test = train_test_split(X[0:len(df_train.full_text)],
                                   df_train.iloc[:,2:].values,test_size=0.20)
print("shape of vectorized train data:",X_train.shape)
print("shape of vectorized test data: ", X_test.shape)
data_test = X[len(df_train.full_text):]
```

```
shape of vectorized train data: (3128, 9380)
shape of vectorized test data:  (783, 9380)
```

```
param_grid = {'C': [ 0.1, 1, 10, 100],  'gamma': [1,0.1, 0.01],
      'kernel': ['rbf', 'poly', 'sigmoid'], 'epsilon':[1,0.1,0.01]}
svr_clf = SVR()
grid = GridSearchCV(svr_clf,param_grid,refit=True,verbose=2)
grid.fit(X_train,y_train[:,-1])
```

```
[ ]  grid.best_params_
```

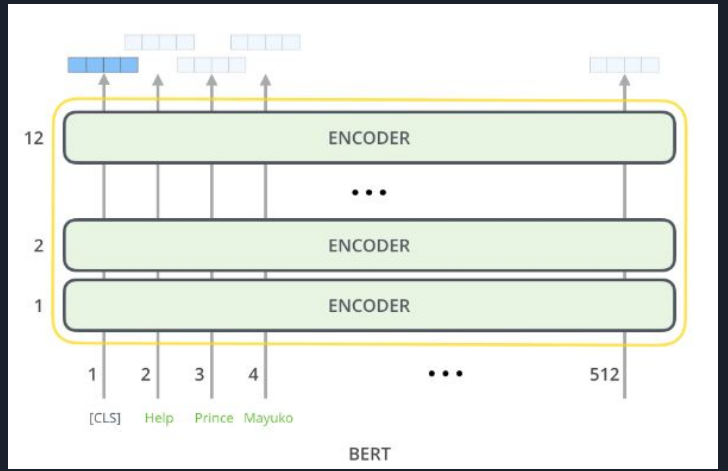
```
{'C': 10, 'epsilon': 0.1, 'gamma': 1, 'kernel': 'rbf'}
```

```
[ ]  %time
     svr_clf = SVR(**grid.best_params_)
     error = []
     for k in range(0,y_train.shape[1]):
       svr_clf.fit(X_train, y_train[:,k])
       rf_preds = svr_clf.predict(X_test)
       error.append(rmse(rf_preds,y_test[:,k],squared=False))
     np.mean(error)
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.72 µs
0.5490805237617936
```

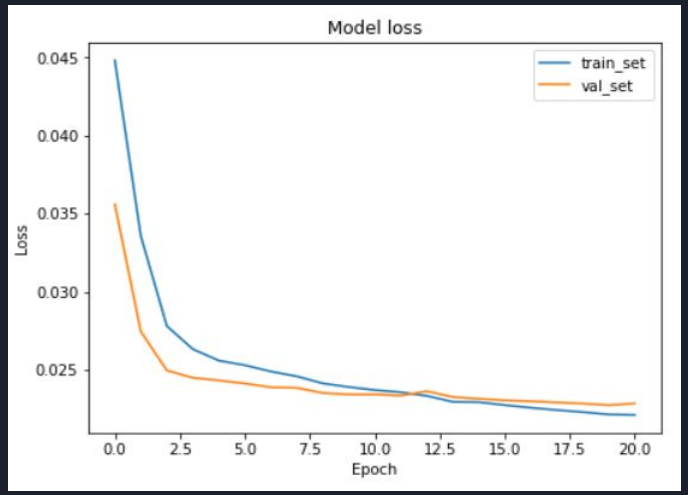
Bert pré-entraîné

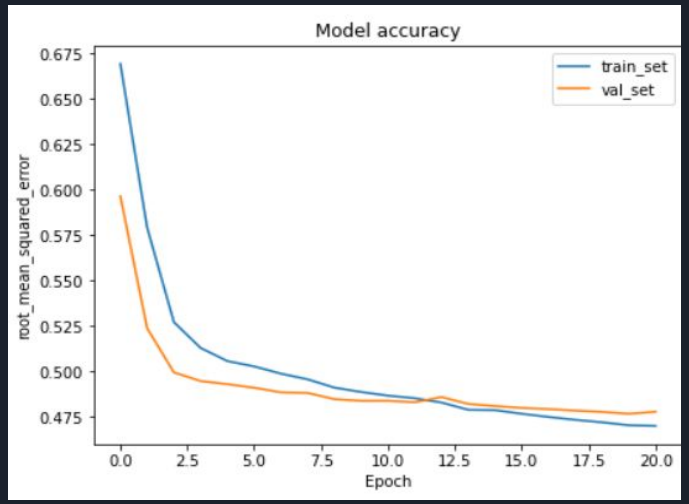
```
bert_model = transformers.TFBertModel.from_pretrained("bert-base-uncased",
    attention_probs_dropout_prob=0,hidden_dropout_prob=0)
# Freeze the BERT model to reuse the pretrained features without modifying them.
bert_model.trainable = False

bert_output = bert_model.bert(
    input_ids, attention_mask=attention_masks, token_type_ids=token_type_ids)
sequence_output = bert_output.last_hidden_state
pooled_output = bert_output.pooler_output
# ---------------------Add trainable layers on top of frozen layers
bi_lstm = tf.keras.layers.Bidirectional(
tf.keras.layers.LSTM(128, return_sequences=True))(sequence_output)
# Applying hybrid pooling approach to bi_lstm sequence output.
avg_pool = tf.keras.layers.GlobalAveragePooling1D()(bi_lstm)
max_pool = tf.keras.layers.GlobalMaxPooling1D()(bi_lstm)
concat = tf.keras.layers.concatenate([avg_pool, max_pool])
dropout = tf.keras.layers.Dropout(0.2)(concat)
x = tf.keras.layers.Dense(512, activation="relu")(dropout)
x = tf.keras.layers.Dense(512, activation="relu")(x)
dropout = tf.keras.layers.Dropout(0.1)(x)
output = tf.keras.layers.Dense(6)(dropout)
model = tf.keras.models.Model(
    inputs=[input_ids, attention_masks, token_type_ids], outputs=output
```



Bert pré-entraîné

accuracy : MRMSE
loss : huber_loss
Epochs =21
MRMSE = 0.47





Comparaison

	SVR	Bert
Score	0.55	0.47
temps (min) d'entraînement	2	59

SVR

cohesion	syntax	vocabulary	phraseology	grammar	conventions
2.866870	2.618208	2.991919	2.998066	2.531559	2.749256
2.936079	2.810271	2.859445	2.682595	2.800824	3.130826
3.555411	3.468012	3.620318	3.499729	3.429987	3.382408

Bert

cohesion	syntax	vocabulary	phraseology	grammar	conventions
2.920458	2.788875	3.050251	2.775156	2.665359	2.615009
2.958915	2.662622	2.901422	2.632174	2.451072	2.769485
3.692701	3.548685	3.664722	3.725807	3.507187	3.661301

Conclusion

- Compétences acquises:
  - Découverte et participation sur la plateforme Kaggle
  - Plus familier avec les algo NLP
- Le modèle Bert améliore nettement les résultats

Pistes d'améliorations:

- ❏ Tester d'autres modèles avec différentes stratégies de pooling
- ❏ Intégrer une équipe de travail
- ❏ Combinaison des modèles

Merci de votre attention