مقدمه

در این پروژه به راهکارهایی برای دسته بندی متون در شش زبان انگلیسی، فرانسوی، فارسی، عربی، آلمانی و پشتو می‌پردازیم.

مراحل انجام کار به ترتیب عبارتند از جمع‌آوری داده مناسب، پیش پردازش و اعمال مدل‌های یادگیری متناسب که شرح تمام آن‌ها در ادامه گزارش آماده‌است.

جمع‌آوری داده

در این قسمت با استفاده از تکنیک Web Scraping و کتابخانه Selenium از دو وب‌سایت ویکی‌پدیا و خبرگزاری Deutsche Welle به ترتیب حدود ۳۰۰۰ و ۶۰۰۰ واژه برای ۶ زبان مدنظر استخراج شده‌است.

داده‌های خبرگزاری تمیزتر! و تقریبا عاری از هرگونه واژه خارجی هستند، در سمت دیگر در بسیاری از داده‌های ویکی‌پدیا چندین زبان (حتی خارج از محدوده زبانی مورد بررسی ما) دیده می‌شود از این رو

از ۹۰ درصد داده‌های دسته خبرگزاری برای آموزش و مابقی داده‌ها برای تست به کار می‌گیریم.

کد مربوط به این بخش با استفاده از Selenium و Jupyter Notebook نوشته شده است که ضمیمه است.

پیش‌پردازش

تمیز کردن متون

در این بخش به استاندارد سازی متون داده شده می‌پردازیم و موارد اضافی را از آن حذف می‌کنیم.

موارد اضافی چه هستند ؟

۱- کاراکترهای خاص

۲- کارکتری‌های تکی

۳- شماره‌های لاتین، فارسی، لاتین (به دلیل اینکه توزیع احتمال نصف می‌باشد.)

۴- آدرس وبسایت‌ها

۵- تگ‌های مرسوم HTML

۶- ایموجی‌ها

تمامی این موارد اطلاعات مفیدی برای تشخیص زبان مدنظر نمی‌دهند و در مواردی می‌توانند گمراه‌کننده و سرباری بر عملیات آموزش باشند پس آن‌ها را حذف می‌کنیم.

چگونه ؟

استفاده از Regex ها و پترن‌های مربوط به هر کدام از موارد بالا و در نهایت حذف آن با استفاده از تکنیک مپینگ در دیتاست.

لازم به ذکر است برای بهینگی و نتیجه بهتر تمامی کاراکترهای دیتاست را به حروف کوچک آن تبدیل می‌کنیم. (زبان‌های با حروف لاتین)

در این بخش به دلیل Language Specific نبودن فضای کار از حذف stop-word ها و برخی نرمالایزهای مرسوم در زبان‌های فارسی و عربی صرف‌نظر شده.

پیاده‌سازی شده در :

```
def preprocess
```

جداسازی داده‌ها

همان‌طور که گفته شد ۱۰ درصد از داده‌های خبرگزاری برای تست به کار می‌رود.

پس به شکل زیر عمل می‌کنیم:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.ID, df.Category, test
_size=0.1, random_state=42)
```

تبدیل متون به وکتورهای عددی با استفاده از روش tf-idf

در بخش به منظور فهماندن کلمات از طریق اعداد به کامپیوتر بر اساس قواعدی اطلاعات متنی هر ردیف را به برداری از اعداد تبدیل می‌کنیم.

روش مرسوم برای این بخش استفاده از کتابخانه sklearn.feature_extraction.text.TfidfVectorizer می‌باشد ولی در این پروژه به صورت دستی پیاده‌سازی شده‌است.

ابتدا به توضیح مختصری در رابطه با این روش می‌پردازیم

TF (Term Frequency): **نسبت تکرار کلمه مدنظر در تک تک ردیف‌های دیتاست به تعداد کل کلمات آن ردیف**

DF (Document Frequency): **نسبت تعداد ردیف‌هایی که کلمه مد نظر را دارند به تعداد کل ردیف‌های دیتاست**

IDF (Inverse Document Frequency): **دادن اهمیت بیشتر به واژه‌های خاص‌تر با معکوس مقدار DF**

مدل استاندارد:

$$IDF = \log(n/(DF+1))$$

مدل scikit-learn:

$$IDF = \log((n+1)/(DF+1)) + 1$$

پیاده‌سازی DF

ساختن دایره واژگان از کل دیتاست و نشان دادن اینکه هر کلمه را کدام ردیف‌ها دارند.

```
DF = {}
for index in X_train.index:
  for w in X_train[index].split():
    try:
      DF[w].add(index)
    except:
      DF[w] = {index}
```

پیاده‌سازی IDF

مقدار IDF به ازای هر کدام از DF ها محاسبه می‌شود.

هر دو روش استاندارد و scikit-learn در این بخش پیاده‌سازی شدند که در نتیجه نهایی تفاوت زیادی نداشتند.

```
N = len(X_train)
IDF = DF
for i in DF:
 # IDF[i] = math.log((N + 1) / float((len(IDF[i]) + 1))) + 1
 IDF[i] = math.log((N) / float((len(IDF[i]) + 1)))
```

پیاده‌سازی TF-IDF:

محاسبه مقدار TF-IDF در کل دیتاست با استفاده از مقادیر IDF بدست آمده و در نهایت محاسبه TF هر کلمه در سطر مورد نظر

```
TF_IDF = {}
```

```
VECTORIZE = dict()
for index in X_train.index:
  tokens = X_train[index].split()
  words_count = len(tokens)
  weight_list = []
  for token in tokens:
    TF = list(tokens).count(token) / float(words_count)
    RES = TF*IDF[token]
    TF_IDF[index, token] = RES
    weight_list.append(RES)
  try:
    VECTORIZE[index].append(weight_list)
  except:
    VECTORIZE[index] = weight_list
```

ساختن جدول وکتورایز شده برای دیتاست

	TF		
	Sent-1	Sent-2	Sent-3
John	1/2	0	0
Cat	1/2	1/3	0
Eat	0	1/3	1/3
Fish	0	1/3	1/3
Big	0	0	1/3

X

	IDF
John	Ln(4/2)+1=1.69
Cat	Ln(4/3)+1=1.28
Eat	Ln(4/3)+1=1.28
Fish	Ln(4/3)+1=1.28
Big	Ln(4/2)+1=1.69

Normalization by the Euclidean norm

	Big	Cat	Eat	Fish	John
Sent_1	0	1.28x0.5=0.640	0	0	1.69x0.5=0.845
Sent_2	0	1.28x0.3=0.384	1.28x0.3=0.384	1.28x0.3=0.384	0
Sent_3	1.69x0.3=0.507	0	1.28x0.3=0.384	1.28x0.3=0.384	0

شکل i

حالا تمامی مقادیر محاسبه شده باید به ردیف‌های متناسب به آن متصل شود.

```
index_mapper = {}
index_mapper_cnt = 0
for i in list(X_train.index):
 index_mapper[i] = index_mapper_cnt
 index_mapper_cnt += 1

print(index_mapper)

VOCAB_LIST = [x for x in DF]
D = np.zeros((len(X_train), len(VOCAB_LIST)))
for i in TF_IDF:
  # print(i)
  ind = VOCAB_LIST.index(i[1])
  D[index_mapper[i[0]]][ind] = TF_IDF[i]
print(D[0])

DATAFRAME = pd.DataFrame(D)

print(DATAFRAME)
```

جدول نهایی (DATAFRAME) شبیه جدول نهایی در *شکل* ***i*** می‌شود.

نکته:

برای وکتورایز کردن داده‌های تست باید از دایره واژگان موجود که از داده‌های آموزش به دست آمده استفاده کنیم و واژه جدیدی را به واژگان خود اضافه نکنیم.

پس دو تابع (معادل توابع fit_transform و transform در کتابخانه اصلی) تعریف می‌کنیم یکی برای آموزش:

```
def vec_fit_transform(X_train)
```

یکی هم برای تست:

```
def vec_transform(X_test, IDF, DF):
```

همان طور که می‌بینیم از مقادیر IDF بدست آماده در مرحله قبل باید استفاده کنیم.

پاس دادن متغیر DF برای داشتن دایره واژگان استفاده می‌شود و مقادیر DF بر اساس شرایط موجود دوباره محاسبه می‌شوند.

```
vectorized_X_train, IDF, DF, vocab_size = vec_fit_transform(X_train)
```

```
vectorized_X_test = vec_transform(X_test, IDF, DF)
wiki_vectorized_X_test = vec_transform(dataset_wikipedia_test.ID, IDF, DF)
```

حالا داده‌ها آماده پردازش هستند!

مدل‌های آموزش

مدل‌های یادگیری عمیق با استفاده از کتابخانه Keras

یادگیری با مدل‌های یادگیری عمیق LSTM و GRU نتیجه مناسبی نداشتند پس به سراغ روش‌های دیگر می‌رویم.

مدل‌های یادگیری با استفاده از کتابخانه Sklearn

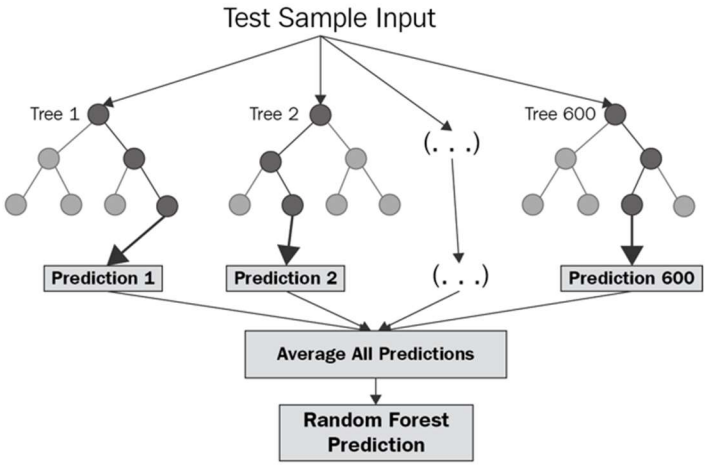
به بررسی بهترین مدل‌های موجود می‌پردازیم:

سمت راست: ۱۰ درصد دیتاست خبرگزاری – سمت راست: دیتاست ویکی‌پدیا

- `RandomForestClassifier`

پارامترهای ورودی:

`n_estimators=200`



تعداد درخت‌هایی که تشکیل می‌شود برابر ۲۰۰ است.

random_state=0

درخت‌ها به صورت تصادفی تشکیل نشوند.

نتایج

```
[[451   2   2   0  45   0]
 [  0 494   0   1   5   0]
 [  0   2 489   0   9   0]
 [  0   0   0 478  22   0]
 [  0   3   0   0 495   1]
 [  0   8   0   0  56 436]]
              precision    recall  f1-score   support

      Arabic       1.00      0.90      0.95       500
     English       0.97      0.99      0.98       500
      French       1.00      0.98      0.99       500
      German       1.00      0.96      0.98       500
      Pashto       0.78      0.99      0.88       499
     Persian       1.00      0.87      0.93       500

    accuracy                           0.95      2999
   macro avg       0.96      0.95      0.95      2999
weighted avg       0.96      0.95      0.95      2999

0.9479826608869624
```

```
[[113   0   0   0   1   0]
 [  0  96   0   0   2   0]
 [  0   1 117   0   0   0]
 [  0   0   0  94   0   0]
 [  0   0   0   0  93   0]
 [  0   0   0   0   0  75]]
              precision    recall  f1-score   support

      Arabic       1.00      0.99      1.00       114
     English       0.99      0.98      0.98        98
      French       1.00      0.99      1.00       118
      German       1.00      1.00      1.00        94
      Pashto       0.97      1.00      0.98        93
     Persian       1.00      1.00      1.00        75

    accuracy                           0.99       592
   macro avg       0.99      0.99      0.99       592
weighted avg       0.99      0.99      0.99       592

0.9932432432432432
```

- LogisticRegression

پارامترهای ورودی

max_iter=1000

ماکسیمم تعداد پیمایش برابر ۱۰۰۰ باشد.

نتایج

```
[[494   2   1   0   1   2]
 [  3 497   0   0   0   0]
 [  6   2 492   0   0   0]
 [  3   1   0 496   0   0]
 [ 62   9   1   0 421   6]
 [ 11   7   0   0   1 481]]
              precision    recall  f1-score   support

      Arabic       0.85      0.99      0.92       500
     English       0.96      0.99      0.98       500
      French       1.00      0.98      0.99       500
      German       1.00      0.99      1.00       500
      Pashto       1.00      0.84      0.91       499
     Persian       0.98      0.96      0.97       500

    accuracy                           0.96      2999
   macro avg       0.96      0.96      0.96      2999
weighted avg       0.96      0.96      0.96      2999

0.9606535511837279
```

```
[[114   0   0   0   0   0]
 [  0  98   0   0   0   0]
 [  0   1 117   0   0   0]
 [  0   0   0  94   0   0]
 [  4   0   0   0  89   0]
 [  0   0   0   0   0  75]]
              precision    recall  f1-score   support

      Arabic       0.97      1.00      0.98       114
     English       0.99      1.00      0.99        98
      French       1.00      0.99      1.00       118
      German       1.00      1.00      1.00        94
      Pashto       1.00      0.96      0.98        93
     Persian       1.00      1.00      1.00        75

    accuracy                           0.99       592
   macro avg       0.99      0.99      0.99       592
weighted avg       0.99      0.99      0.99       592

0.9915540540540541
```

- SGDClassifier

پارامترهای ورودی

max_iter=10000

ماکسیمم تعداد پیمایش برابر ۱۰۰۰۰ باشد.

alpha=0.00001

نرخ یادگیری برابر ۰/۰۰۰۰۱ باشد.

loss="modified_huber"

میزان خطا توسط رابطه زیر محاسبه شود.

$$L(y, f(x)) = \begin{cases} \max(0, 1 - y\,f(x))^2 & \text{for } y\,f(x) \geq -1, \\ -4y\,f(x) & \text{otherwise.} \end{cases}$$

نتایج

```
[[485   2   1   0   1  11]
 [  0 496   0   1   0   3]
 [  0   2 492   0   0   6]
 [  0   1   0 496   0   3]
 [  2   6   1   0 416  74]
 [  1   7   0   0   1 491]]
              precision    recall  f1-score   support

      Arabic       0.99      0.97      0.98       500
     English       0.96      0.99      0.98       500
      French       1.00      0.98      0.99       500
      German       1.00      0.99      0.99       500
      Pashto       1.00      0.83      0.91       499
     Persian       0.84      0.98      0.90       500

    accuracy                           0.96      2999
   macro avg       0.96      0.96      0.96      2999
weighted avg       0.96      0.96      0.96      2999

0.9589863287762588
```

```
[[113   0   0   0   0   1]
 [  0  98   0   0   0   0]
 [  0   1 117   0   0   0]
 [  0   0   0  94   0   0]
 [  0   0   0   0  89   4]
 [  0   0   0   0   0  75]]
              precision    recall  f1-score   support

      Arabic       1.00      0.99      1.00       114
     English       0.99      1.00      0.99        98
      French       1.00      0.99      1.00       118
      German       1.00      1.00      1.00        94
      Pashto       1.00      0.96      0.98        93
     Persian       0.94      1.00      0.97        75

    accuracy                           0.99       592
   macro avg       0.99      0.99      0.99       592
weighted avg       0.99      0.99      0.99       592

0.9898648648648649
```

- LinearSVC

```
[[487   2   1   0   8   2]
 [  0 497   1   0   2   0]
 [  0   2 493   0   5   0]
 [  0   1   0 496   3   0]
 [  2   8   1   0 485   3]
 [  0   7   0   0  13 480]]
              precision    recall  f1-score   support

      Arabic       1.00      0.97      0.98       500
     English       0.96      0.99      0.98       500
      French       0.99      0.99      0.99       500
      German       1.00      0.99      1.00       500
      Pashto       0.94      0.97      0.96       499
     Persian       0.99      0.96      0.97       500

    accuracy                           0.98      2999
   macro avg       0.98      0.98      0.98      2999
weighted avg       0.98      0.98      0.98      2999

0.9796598866288763
```

```
[[113   0   0   0   1   0]
 [  0  98   0   0   0   0]
 [  0   1 117   0   0   0]
 [  0   0   0  94   0   0]
 [  0   0   0   0  93   0]
 [  0   0   0   0   0  75]]
              precision    recall  f1-score   support

      Arabic       1.00      0.99      1.00       114
     English       0.99      1.00      0.99        98
      French       1.00      0.99      1.00       118
      German       1.00      1.00      1.00        94
      Pashto       0.99      1.00      0.99        93
     Persian       1.00      1.00      1.00        75

    accuracy                           1.00       592
   macro avg       1.00      1.00      1.00       592
weighted avg       1.00      1.00      1.00       592

0.9966216216216216
```

همان‌طور که مشاهده می‌شود مدل LinearSVC بهترین عملکرد را دارد.

مدل یادگیری (پیاده‌سازی دستی)

منبع:

[Implement-SGD-from-scratch./Implement SGD for Linear Regression for Boston Housing Dataset.ipynb at master · saugatapaul1010/Implement-SGD-from-scratch. (github.com)]

در تلاشی دیگر با اعمال تغییراتی بر روی کد بالا مدل SGDClassifier از پایه پیاده‌سازی شده که از نظر زمانی و نتیجه پایین‌تر از SGDClassifier خود کتابخانه scikit-learn می‌باشد.

برای صرفه‌جویی در زمان وزن‌های بدست آمده را در فایل ذخیره می‌کنیم.

همچنین ولیوها به دست‌آمده ممکن است از بازه مدنظر ما ۰ – ۵ کمی فاصله داشته باشد.

تمامی این مقادیر با استفاده از توزیع نرمال به این محدود مپ می‌کنیم.

نتایج نهایی به شرح زیر می‌باشند:

```
[[495   0   1   1   0   3]
 [  3   4 210 277   0   6]
 [  6   0   8   1   0 485]
 [  3   0 459   1   0  37]
 [100   0   5   1 390   3]
 [148   0   1   0   0 351]]
              precision    recall  f1-score   support

      Arabic       0.66      0.99      0.79       500
     English       1.00      0.01      0.02       500
      French       0.01      0.02      0.01       500
      German       0.00      0.00      0.00       500
      Pashto       1.00      0.78      0.88       499
     Persian       0.40      0.70      0.51       500

    accuracy                           0.42      2999
   macro avg       0.51      0.42      0.37      2999
weighted avg       0.51      0.42      0.37      2999

0.41647215738579524
```

```
[[188   0   0   0   0   0]
 [  0  28  23 162   0   2]
 [  0   0 135   1   0  73]
 [  0   0 153  38   0   1]
 [  7   0   0   0 191   0]
 [  3   0   0   0   0 178]]
              precision    recall  f1-score   support

      Arabic       0.95      1.00      0.97       188
     English       1.00      0.13      0.23       215
      French       0.43      0.65      0.52       209
      German       0.19      0.20      0.19       192
      Pashto       1.00      0.96      0.98       198
     Persian       0.70      0.98      0.82       181

    accuracy                           0.64      1183
   macro avg       0.71      0.65      0.62      1183
weighted avg       0.71      0.64      0.61      1183

0.6407438715131023
```

لینک colab

https://colab.research.google.com/drive/162MlavzWuuloq6ZiS3Ow2dJJ4FQefU9X?usp=sharing