اللغة العربية والذكاء الاصطناعي

نماذج لغة عربية تم حلها باستخدام التعلم الآلي والتعلم العميق

ترجمة واعداد: د. علاء طعيمة

بـــسم تعالى

اللغة العربية والذكاء الاصطناعي

مشاريع لغة عربية تم حلها باستخدام التعلم الآلي والتعلم العميق

ترجمة واعداد:

د. علاء طعيمة

مقدمة المترجم

تعد اللغة العربية من أقدم لغات العالم وأغناها؛ وركناً من أركان التنوع الثقافي للبشرية ،وهي من أكثر اللغات الست انتشارًا واستخدامًا على مستوى العالم ، وهي من بين اللغات الأربع الأكثر استخدامًا في الإنترنت، وهي اللغة الرسمية لكل الدول العربية، يتحدث بها أكثر من 500 مليون نسمة بالوطن العربي ، إلى جانب الدول المسلمة غير العربية، وهي ضرورية لأكثر من مليار مسلم لتأدية الصلاة وقراءة القرآن ، وهي تمتاز بخصائص ومزايا نادرًا ما تجتمعفي غيرها من اللغات، فلغة الضاد هي لغة الصوت والصورة، والمفردات والتراكيب، والحكم والأمثال.

ومن المؤكّد أنّ الذكاء الاصطناعي (التعلم الآلي والتعلم العميق) يسمح بالقيام بخطوات جبّارةفي مجال خدمة اللغة العربية، وجعلهافي مستوى اللغات العالمية، من حيث مواكبة التقنية، ومن حيث التنافسية معها، حيث يكون المجال متاحا للوصول إلى شرائح واسعة من الناطقين بغير اللغة العربية التّواقين لتعلّم لغة الضاد.

تم تطبيق الذكاء الاصطناعي على العديد من مشاريع اللغة العربية، بمافي ذلك توليد القصائد العربية وتحليل المشاعر العربية والتلخيص التلقائي للنص العربي وغيرها من المشاريع. يحل الذكاء الاصطناعي العديد من مشكلات اللغة العربية وتخلق حلولاً جديدة.في هذ الكتاب، سوف نستكشف دور الذكاء الاصطناعيفي حل مشكلات اللغة العربية من خلال العديد من المشاريع التي تم حلها وشرحها باستخدام نماذج التعلم الآلي والتعلم العميق.

لقد حاولت قدر المستطاع ان اترجم المقالات والمشاريع الأكثر طرحاًفي مجال اللغة العربية والذكاء الاصطناعي مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq.

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجال اللغة العربية والذكاء الاصطناعي ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية / العراق

المحتويات

1) التعلم الآلي والمعالجة اللغوية الطبيعية للغة العربية Machine Learning and NLP For Arabic

يدور هذا المشروع من Omdena حول بناء مكتبات وأدوات المعالجة اللغوية الطبيعية NLP مفتوحة المصدر للغة العربية، حيث أن اللغة العربية هي اللغة الخامسة الأكثر استخدامًافي العالم وهناك العديد من التحديات التي يمكن العثور عليهافي اللغة العربية مثل القواعد المعقدة ووجود لهجات متعددة.

لذا فإن الهدف من مشروعنا هو بناء هذه المكتبة مفتوحة المصدر التي تساعد الشعب العربيفي تطبيقات المعالجة اللغوية الطبيعية. سنتحدثفي هذه المقالة عن تصنيف اقسام الكلام speech tagging للغة العربية.

ما هو تصنيف اقسام الكلام

إن تصنيف اقسام الكلام Part-of-speech (POS) يعني ببساطة تصنيف الكلمات باستخدام جزء الكلام المناسب لها بحيث يشرح كيفية استخدام الكلمةفي الجملة.

word	POS
جاء	verb
فى	Preposition
التعاون	Noun

تعتمد النماذج الأساسيةفي المعالجة اللغوية الطبيعية على حقيبة الكلمات Bag of Words التي لا تعد الحل الأفضل لأنها تفشلفي التقاط أي علاقات نحوية بين الكلمات.

لذلك يمكننا من تحسين نموذج حقيبة الكلمات هذا باستخدام تقنية تصنيف اقسام الكلام POS.

بعض تطبيقات وضع العلامات على POS

- التعرف على الكيان المسمى Named entity recognition (NER).
- يعد POS أمرًا ضروريًا لبناء أدوات lemmatizer التي تقلل الكلمات إلى شكلها الجذري.
- تحليل المشاعر Sentiment analysis.

يعتمد تصنيف الكلمة على تصنيف اقسام الكلام على سياقهافي الجملة، لذا فإن هذه المهمة ليست واضحة لأن الكلمة قد تحتوي على علامة POS (POS tag) مختلفة بناءً على سياقهافي الجملة.

لذلك،في هذه المقالة، سننظرفي استخدام أساليب التعلم العميق ـ الشبكات العصبية المتكررة Recurrent Neural Networks لتصنيف اقسام الكلام POS tagging.

جمع البيانات والمعالجة المسبقة

في البداية استخدمنا مجموعة البيانات العربية مفتوحة المصدر UD_Arabic-PADT لأنها مجموعة بيانات مرجعية ومعروفة لتصنيف pos ولكن بعد ذلك قررنا إنشاء مجموعة بيانات أخرى من أجل الحصول على مجموعة بيانات أكبر وأكثر تنوعًا

لذلك قام الفريق الفرعي لجمع البيانات باستخلاص البيانات من الإنترنت واستخدم بعض المكتبات لإنشاء بيانات مشروحة تتكون من جمل حيث يتم تعيين كل كلمة لعلامة pos، لذا كانت مجموعة البيانات النهائية التي استخدمناها هي أن البيانات تتكون من 36000 جملة.

لقد قمنا ببعض خطوات المعالجة المسبقة من أجل جعل البيانات بالشكل الصحيح لتكون مدخلاً لنموذج التعلم الآلي ويتم التدريب عليها.

```
def process_csv(csv):
df = pd.read_csv(csv)
train_text ,train_tags = [] ,[]
for i in tqdm(df['sentence_id'].unique()):
train_text.append(df[df['sentence_id'] == i]['word'].tolist())
train_tags.append(df[df['sentence_id'] == i]['tag'].tolist())
return train_text,train_tags
```

كانت الخطوة التاليةفي المعالجة المسبقة لهذه البيانات هي إزالة التشكيل Diacritization والإطالة longation حتى نتمكن من الحصول على جمل واضحة وموحدة.

```
def clean_str(text):

#remove tashkeel
p_tashkeel = re.compile(r'[\u0617-\u061A\u064B-\u0652]')
text = re.sub(p_tashkeel,"", text)
#remove longation
p_longation = re.compile(r'(.)\1+')
subst = r"\1\1"
text = re.sub(p_longation, subst, text)
text = text.replace('وو', 'و')
text = text.replace('يي', 'ي')
text = text.replace('اا', 'ا')
text = text.replace('أ', 'ا')
text = text.replace('إ', 'ا')
text = text.replace('آ', 'ا')
text = text.replace('ى', 'ي')
return text.split()
for i in range(len(train_text)):
train_text[i] = clean_str(' '.join(train_text[i]))
```

التوكينازيشن

نموذج التعلم الآلي لا يفهم الكلمات لذا نحتاج إلى ترميز بيانات الإدخال والإخراج. لذلك نعطي معرفًا فريدًا لكل كلمةفي بيانات الإدخال. ومن ناحية أخرى، نعطي معرفًا فريدًا لكل علامةفي بيانات الإخراج.

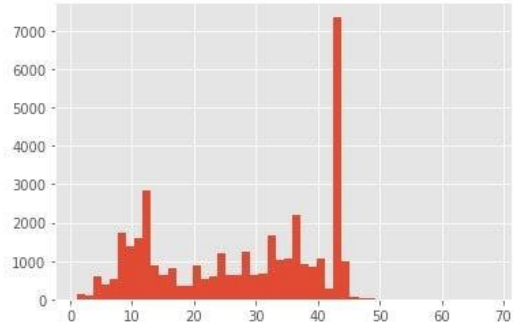
استخدمنا دالة tokenizer من مكتبة Keras لترميز تسلسل النص إلى تسلسل صحيح يمكن استخدامه كمدخل لنموذج التعلم الآلي للتدريب.

```
word_tokenizer = Tokenizer(oov_token = oov_tok)
word_tokenizer.fit_on_texts(train_text)
VOCABULARY_SIZE = len(word_tokenizer.word_index) + 1
X_encoded_train = word_tokenizer.texts_to_sequences(train_text)

tag_tokenizer = Tokenizer()
tag_tokenizer.fit_on_texts(train_tags)
Y_encoded_train = tag_tokenizer.texts_to_sequences(train_tags)
```

حشو التسلسل

جملنا الآن ليس لها نفس الطول لذلك قمنا بتمثيل الجمل على شكل رسم بياني لنرى أطوال جميع الجمل ونجد الحد الأقصى للطول حتى نتمكن من جعله طول كل تسلسل عن طريق حشو padding التسلسلات الأصغر بالأصفار.

```
[ ]  1 plt.style.use("ggplot")
     2 plt.hist([len(s) for s in train_text], bins=50)
     3 plt.show()
```




```
[ ]  1 print('Max sentence length:',len(max(train_text, key=len)))
```

```
Max sentence length: 68
```

استخدمنا دالة pad sequences من keras وحددنا طول التسلسل ليكون 50 و padding_type = "post" مما يعني أن الأصفار المحشوة padded zeros ستكونفي نهاية كل تسلسل.

أصبحت بياناتنا الآن بالشكل الصحيح وجاهزة للاستخدامفي نموذج التعلم الآلي، لذا قمنافي النهاية بتقسيمها إلى 20% للتحقق من الصحة validation وحوالي 15% للاختبار testing.

```
X_train = pad_sequences(X_encoded_train,
maxlen=MAX_SEQUENCE_LENGTH, padding=padding_type,
truncating=trunc_type)
Y_train = pad_sequences(Y_encoded_train,
maxlen=MAX_SEQUENCE_LENGTH, padding=padding_type,
truncating=trunc_type)
```

تقسيم البيانات إلى مجموعات بيانات التدريب Training والتحقق من الصحة Validation والاختبار Testing.

```
X_train, X_valid , Y_train, Y_valid = train_test_split(X_train,
Y_train, test_size = 0.20, random_state = 41)

X_train, X_test, Y_train, Y_test = train_test_split(X_train,
Y_train, test_size=0.15, random_state=41)
```

تضمين الكلمات

تعد عمليات تضمين الكلمات Word embeddings أحد أنواع تمثيل الكلمات word representation الذي يسمح للكلمات ذات المعنى المماثل بأن يكون لها تمثيل مماثل. لذلك استخدمنا مصفوفة تضمين من Aravec لإضافتها إلى نموذجنا حيث تم تدريبها على مجموعة كبيرة بحيث يمكن لكلماتنا الموجودةفي المصفوفة التي تم تنزيلها أن تأخذ التمثيلات المقابلة التي قد تساعد النموذجفي معرفة دلالات كل كلمة.

```
!wget https://bakrianoo.ewr1.vultrobjects.com/aravec/full_grams_cbo
w_300_twitter.zip
!unzip full_grams_cbow_300_twitter.zip

import genism

embedding_model =
gensim.models.Word2Vec.load('full_grams_cbow_300_twitter.mdl')
embeddings = {}
for word,vector in
zip(embedding_model.wv.vocab,embedding_model.wv.vectors):
coefs = np.array(vector, dtype='float32')
embeddings[word] = coefs
embeddings_weights = np.zeros((VOCABULARY_SIZE, embedding_dim))
for word, i in word_tokenizer.word_index.items():
embedding_vector = embeddings.get(word)
if embedding_vector is not None:
embeddings_weights[i] = embedding_vector
```

بناء النماذج

بناء النموذج هو المفتاح لتدريب أي نموذج للتعلم الآلي. لذلك اخترنا بناء العديد من نماذج الشبكات العصبية.

الشبكة العصبية المتكررة(Recurrent Neural Network (RNN، الوحدة المتكررة المسورة Gated recurrent unit (GRU)، الذاكرة الطويلة قصيرة المدى Long short-term memory (LSTM)، و LSTM ثنائية الاتجاه Bidirectional LSTM (BILSTM) هي نماذج الشبكة العصبية التي تم اختيارها للبناء.

بالنسبة لجميع النماذج، استخدمنا دالة الخسارة (الخطأ) loss function مع categorical_crossentropy والتي يتم تعريفها على أنها إنتروبيا متقاطعة فئوية بين موتر الإخراج output tensor وموتر الهدف target tensor وبالنسبة للمُحسِّن optimizer، استخدمنا adam عبارة عن خوارزمية تحسين بديلة للتدرج الاشتقاقي العشوائي stochastic gradient descent لتدريب نماذج التعلم العميق.

تم إنشاء جميع النماذج باستخدام مكتبة Keras لنماذج الشبكة العصبية Neural Network models والتوكينازيشن Tokenization. بالإضافة إلى المكتبات التي استخدمناها من قبل لتغذية النماذج بالبيانات مثل Pandas لإعداد البيانات ومعالجتها مسبقًا، وGensim لتضمين الكلمات.

```
model = Sequential()
model.add(InputLayer((MAX_SEQUENCE_LENGTH)))
model.add(Embedding(input_dim = VOCABULARY_SIZE,
output_dim = embedding_dim,
input_length = MAX_SEQUENCE_LENGTH,
weights = [embeddings_weights],
trainable = True
))

model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Bidirectional(LSTM(256, return_sequences=True)))


model.add(TimeDistributed(Dense(NUM_CLASSES,
activation='softmax')))
model.compile(loss='categorical_crossentropy',
optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 50, 300)           21471300
_________________________________________________________________
bidirectional_4 (Bidirection (None, 50, 512)           1140736
_________________________________________________________________
bidirectional_5 (Bidirection (None, 50, 512)           1574912
_________________________________________________________________
bidirectional_6 (Bidirection (None, 50, 512)           1574912
_________________________________________________________________
bidirectional_7 (Bidirection (None, 50, 512)           1574912
_________________________________________________________________
time_distributed_1 (TimeDist (None, 50, 35)            17955
=================================================================
Total params: 27,354,727
Trainable params: 27,354,727
Non-trainable params: 0
_________________________________________________________________
```

الآن أصبح النموذج جاهزًا للتدريب، قمنا بتدريب النموذج لمدة 50 فترة epochs وكان حجم الدفعة batch size هو 128. كما استخدمنا رد اتصال ReducLROnPlateau من keras بحيث ينخفض معدل التعلم learning rate عندما لا تتحسن الدقة بعد 6 فترات ونحفظ أفضل نماذج الأوزانفي النهاية تعتمد على أعلى دقةفي التحقق.

بعد ذلك، قمنا ببناء الة المتجهات الداعمة Support vector machine (SVM).

معلمة التنظيم Regularization parameter هي أن قوة التنظيم تتناسب عكسيا مع C المختارة لتكون C = 10.0.

```
kernel{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’},
default=’rbf’ Specifies the kernel type to be used in the
algorithm. chosen kernel=’rbf’.
```

في البداية استغرق SVM ساعة واحدة ولكن وجدنا أن هناك مكتبة تسمى مكتبة Thundersvm والتي توفر دعم GPU لـ SVM والذي سيكون أسرع من SVM. بهذه الطريقة استغرق Thundersvm 5 دقائق بدلاً من ساعة واحدة.

```
! git clone https://github.com/Xtra-Computing/thundersvm.git
! cd thundersvm && mkdir build && cd build && cmake .. && make -j
! python /content/thundersvm/python/setup.py install
from importlib.machinery import SourceFileLoader
thundersvm = SourceFileLoader(“thundersvm”,
“/content/thundersvm/python/thundersvm/thundersvm.py”).load_module(
)
from thundersvm import SVC
```

```
clf = SVC(C=10)
clf.fit(x_train, y_train)
```

SVM

آلة المتجهات الداعمة (SVM) وهي عبارة عن مجموعة من أساليب التعلم الخاضعة للإشراف supervised learning المستخدمة للتصنيف classification والانحدار regression والكشف عن القيم المتطرفة outlier detection.

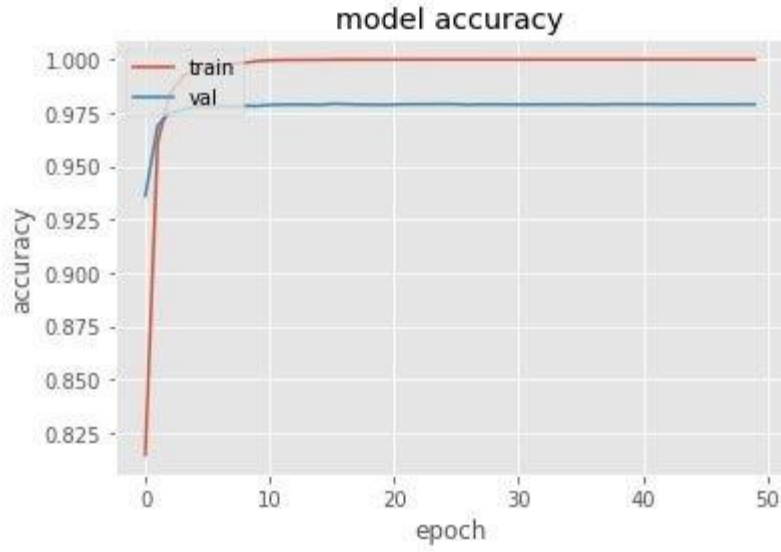
```
sklearn.svm.SVC(*, C=10.0, kernel='rbf', degree=3, gamma='scale',
coef0=0.0, shrinking=True, probability=False, tol=0.001,
cache_size=200, class_weight=None, verbose=False, max_iter=- 1,
decision_function_shape='ovr', break_ties=False, random_state=None)
```

بالإضافة إلى ذلك، جربنا BERT أيضًا، لكن الدقة لم تكن جيدة تمامًا، واستغرق التشغيل الكثير من الوقت. لذلك قررنا استخدام BILSTM لأنه يتمتع بأفضل دقة للنموذج.

تقييم النموذج

يعد رسم منحنيات النموذج مفيداً لأنه قد يوضح ما إذا كانت هناك مشكلةفي التدريب أو إذا كان هناك الضبط الزائد overfitting كما أنه يساعدنا على رؤية الفرقفي أداء النموذج بين التدريب training وبيانات التحقق غير المرئية unseen validation data.

```
plt.plot(result.history['accuracy'])
plt.plot(result.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','val'],loc='upper left')
plt.show()
```



بعد الانتهاء من التدريب نحتاج إلى تقييم النموذج ونقوم بحساب دقته.

```
loss, accuracy = model.evaluate(X_test, Y_test, verbose = 1)
print("Loss: {0},\nAccuracy: {1}".format(loss, accuracy))
```

بعد أن قمنا بحساب درجة F1 لـ BILSTM وSVM.

```
from sklearn.metrics import f1_score, precision_score,
recall_score, confusion_matrix
y_pred1 = model.predict(X_test)
y_pred = np.argmax(y_pred1, axis=-1)
y_pred = y_pred.reshape((y_pred.shape[0]*y_pred.shape[1],))
Y_test = np.argmax(Y_test, axis=-1)
Y_test = Y_test.reshape((Y_test.shape[0]*Y_test.shape[1],))
print(f1_score(Y_test, y_pred , average="macro"))
```

التنبؤ بالنموذج

يمكننا الآن استخدام النموذج المدرب للتنبؤ بجملة جديدة ولكن يجب علينا أولاً تمرير هذه الجملة إلى خطوات المعالجة المسبقة مثل إزالة التشكيل dictritization وتحويل الجملة إلى تسلسل من الكلمات وأخيراً حشو هذا التسلسل ليكون بنفس الشكل الذي يتوقعه النموذج.

```
def classify(sentence):
sentence = clean_str(sentence)
seq = [word_tokenizer.texts_to_sequences(sentence)]
pad_seq = pad_sequences(seq, maxlen=MAX_SEQUENCE_LENGTH,
padding=padding_type, truncating=trunc_type)
pad_seq = np.squeeze(pad_seq,axis=-1)
pred = np.squeeze(model.predict(pad_seq).argmax(-1))
output = [tag_tokenizer.index_word[tag] for tag in pred if tag !=
0]
return output

sentence = "جون يحب البيت الأزرق في نهاية الشارع"
output = classify(sentence)
word_tag = [(sentence.split()[i],output[i]) for i in
range(len(sentence.split()))]
print(word_tag)
```

المخرجات:

```
[('جون', 'proper noun'), ('يحب', 'verb'), ('البيت', 'noun'),
('الأزرق', 'noun'), ('في', 'preposition'), ('نهاية', 'noun'),
('الشارع', 'noun')]
```

Dataset Name	Classifier Name	Accuracy
Data Collation Team	RNN	97.15
Data Collation Team	LSTM	96.94
Data Collation Team	*GRU*	*97.04*
Data Collation Team	BiLSTM	***97.25***
Data Collation Team	*SVM1 - C = 1.0 embeddings = 100*	94.03
Data Collation Team	*SVM2 - C = 10.0 embeddings = 100*	94.36
Data Collation Team	*SVM3 - C = 100.0 embeddings = 100*	94.22
Data Collation Team	*SVM1 - C = 1.0 embeddings = 300*	93.61
Data Collation Team	*SVM2 - C = 10.0 embeddings = 300*	**93.90**
Data Collation Team	*SVM3 - C = 100.0 embeddings = 300*	**93.54**

Dataset Name	Classifier Name	Accuracy
UD_Arabic-PADT	RNN	95.7
UD_Arabic-PADT	LSTM	96.7
UD_Arabic-PADT	*GRU*	*97*
UD_Arabic-PADT	BiLSTM	97.2
UD_Arabic-PADT	BiLSTM (Modified)	**99.3**
Data Collation Team	BILSTM 1 -Embedding own-Maximum length - 39	97.25
Data Collation Team	BILSTM 2 -Embedding own-Maximum length - 500	**99.75**
Data Collation Team	BILSTM 3 - Embedding Pretrained - Maximum length - 39	96.92
Data Collation Team	BILSTM 4 - Embedding Pretrained - Maximum length - 500	**99.738**

النتائج النهائية

Dataset Name	Classifier Name	Accuracy	F1 Score
Final Data Collation Team	BILSTM 4 - Embedding Pretrained - Maximum length - 500	98	90.17

Dataset Name	Classifier Name	Accuracy	F1 Score
Final Data Collation Team	SVM	93.76	76.64

الاستنتاج

باختصار، أفضل نموذج حقق أفضل دقة هو BILSTM. كانت الدقة ودرجة F1 لـ BILSTM هي97.94% و90.19%.في هذه المقالة، عرضنا التقدم المحرزفي إعداد مجموعات البيانات ومعالجتها مسبقًا من خلال التوكينازيشن tokenization، وتضمين المصفوفة Embedding Matrix، وتقدم تسلسلات الحشو Pad sequences progress. النماذج التي قمنا بتنفيذها هي RNN وGRU وLSTM وBILSTM وSVM. وأخيرًا نقوم بتقييم كل نموذج ثم مقارنة نتائج النماذج المختلفة واختيار النموذج الأفضل بأفضل دقة.

المصدر:

https://medium.com/omdena/machine-learning-and-nlp-for-arabic-part-of-speech-tagging-d8388c1c2e84

2) المعالجة اللغوية الطبيعية للغة العربية: التحديات وحلولها Arabic NLP: Challenges and Their Solutions

في هذه المقالة، أقدم نظرة عامة موجزة ودقيقة عن تحديات العمل مع النص العربيفي مشاريع المعالجة اللغوية الطبيعية NLP... والأدوات المتاحة للتغلب عليها. أعتمد بشكل كبير على حزمة بايثون لأدوات الجمل التي تم تطويرهافي مختبر CAMeL بجامعة نيويورك أبو ظبي وهذه الندوة عبر الإنترنت الممتازة التي يقدمها مديرها الدكتور نزار حبش.

تحية كبيرة لهم لقيامهم بعمل رائدفي هذا المجال وجعل أدواتهمفي متناول الجمهور!.

التحديات

يمثل العمل مع النص العربيفي مشاريع المعالجة اللغوية الطبيعية (NLP) خمس تحديات فريدة (على الأقل):

1. يمكن أن يختلف شكل الأحرف وتهجئة الكلمات وفقًا لسياقها (مصطلح فاخر: الغموض الإملائي Orthographic Ambiguity)
2. يمكن أن يحتوي الفعل نفسه على آلاف (حرفيًا) من الأشكال المختلفة (مصطلح فاخر: الثراء الصرفي Morphological Richness)
3. هناك لهجات dialects كثيرة للغة العربية وهناك اختلافات كبيرة بينها (اختلاف اللهجات Dialectal Variation)
4. بما أن اللغة العربية هي لغة صوتية (ما تكتبه هو ما تقوله)، فمن الممكن أن تكون هناك طرق مختلفة لكتابة نفس الكلمة عند الكتابة باللهجة العربية، والتي لا يوجد لها معيار متفق عليه (عدم الاتساق الإملائي Orthographic Inconsistency).

تساهم هذه الخصائص الأربع المختلفة للغة العربيةفي شح البيانات. بسبب الأشكال العديدة المختلفة للكلمات، والتهجئة المختلفة المحتملة، والعدد الكبير من اللهجات، يمكن أن ينتهي الأمر بالفرد بمفردات المعالجة اللغوية الطبيعية NLP التي تصل إلى ملايين الكلمات. هذه ليست مبالغة: يمكن أن يحتوي الفعل الواحد باللغة العربية على ما يصل إلى 5400 شكل (!) مقارنة بحد أقصى 6 أشكال في اللغة الإنجليزية وشكل واحد فقطفي اللغة الصينية.

الحلول

لكن لا تخف! عندما تكون عالقًافي صحراء اللغات منخفضة الموارد ويكون مسارك محصورًا ببعض الأكواد السائلة لمواصلة رحلتك... هناك يتم إنشاء **CAMeL سحري وسهل الاستخدام لمساعدتك:

```
pip install camel-tools
```

سأشارك أدناه الخطوات التي اتخذتها للمعالجة المسبقة للنص العربي الخاص بي والتعامل مع التحديات الفريدة التي تفرضها اللغة العربية باستخدام حزمة camel-tools.

ملاحظة: أفترض أنك قد قمت بالفعل بجميع عمليات المعالجة المسبقة الأساسية والعالمية للمعالجة اللغوية الطبيعية مثل إزالة الأحرف المتكررة وكلمات التوقف والرموز التعبيرية وعلامات التصنيف والأرقام وأي من مهام تنظيف البيانات الأخرى التي تعد أفضل الممارسات والتي تعد جزءًا من أي مشروع المعالجة اللغوية الطبيعية. يحتوي Github repo المشار إليه أدناه على قائمة شاملة بكلمات التوقف العربية التي تفسر الغموض الإملائي المذكور سابقًا.

ملاحظة هامة: تأكد من إضافة كلمة "يا" إلى الملف لأنها غير مضمنة حاليًا وهي كلمة توقف شائعة جدًا.

لاحظ أن NLTK يحتوي أيضًا على مكتبة من الكلمات التوقف باللغة العربية ولكن هذه المجموعة تفتقد العديد من هذه التناقضات الإملائية الشائعة.

https://github.com/mohataher/arabic-stop-words?source=post_page-----d99e8a87893d

حسنًا، الآن دعونا نبدأ!

الخطوة 1: إزالة التشكيل

الخطوة الأولى هي تقليل بعض التباين الخطيرفي البيانات عن طريق إزالة علامات التشكيل diacritics من النص. علامات التشكيل Diacritics هي الرموز symbols (التي يمكن مقارنتهافي بعض الحالات بأحرف العلة vowelsفي اللغة الإنجليزية) الموجودة أعلى أو أسفل أحرف النص العربي ـ العلامات الزرقاءفي الصورة أدناه.

قد تحتوي الكلمة نفسها على علامات تشكيل مختلفة اعتمادًا على سياقها، وبالتالي فمن الشائع إزالة هذه الرموز لتقليل تناثر البيانات data sparsity.

```
# import the dediacritization tool
from camel_tools.utils.dediac import dediac_ar

# apply to your text column
```

```
df.tweet_text = df.tweet_text.apply(dediac_ar)
```

من الناحية الفنية، يؤدي هذافي الواقع إلى خلق مشكلة جديدة لأن نفس جذر الكلمة word-root يمكن أن يكون له معاني مختلفة تمامًا اعتمادًا على علامات التشكيل التي تكتبها...ولكننا سنتعامل مع ذلكفي الخطوة 4.

الخطوة 2: الحد من الغموض الإملائي

ولمراعاة عدد من التناقضات الإملائية spelling inconsistencies الشائعة عبر اللهجات dialects (وبسبب الفجوة بين اللغة العربية المنطوقة والمكتوبة بشكل عام)، فإن الخطوة التالية هي تقليل الغموض الإملائي orthographic ambiguity. على وجه التحديد، تقوم camel-tools بذلك عن طريق إزالة رموز معينة من حروف معينة (النقاط من التاء المربوطة teh-marbuta والهمزة hamza من الألف alef).

```
from camel_tools.utils.normalize import normalize_alef_maksura_ar
from camel_tools.utils.normalize import normalize_alef_ar
from camel_tools.utils.normalize import normalize_teh_marbuta_ar

def ortho_normalize(text):
    text = normalize_alef_maksura_ar(text)
    text = normalize_alef_ar(text)
    text = normalize_teh_marbuta_ar(text)
    return text

df.tweet_text = df.tweet_text.apply(ortho_normalize)
```

الخطوة 3: رمز بسيط للكلمات

الخطوة التالية هي مجرد أداة رمزية بسيطة للكلمات simple word tokenizer. نحن بحاجة إلى ذلك حتى نتمكن من إدخال النصفي دوال خطوتنا التالية.

```
from camel_tools.tokenizers.word import simple_word_tokenize

df.tweet_text = df.tweet_text.apply(simple_word_tokenize)
```

الخطوة 4: توضيح الصرف

الآن هذا هو المكان الذي تصبح فيه الأمور مثيرة للاهتمام. هل تتذكر كيف قلتفي نهاية الخطوة 1 أن إزالة علامات التشكيل تؤديفي الواقع إلى مشكلة جديدة؟ لدينا الآن الأحرف الجذرية فقط، ولكن من الناحية الفنية، لا توجد طريقة لمعرفة أي من الكلمات المختلفة يمكن أن تكون هذه الكلمة. على سبيل المثال، يمكن أن تعني الكلمة أدناه بشكل بديل: "وبعقدنا/قلادتنا/ذهاننا" ' and with our contract / necklace / psychoses' أو "وهو يشددنا" 'and he stresses us out' ـ اعتمادًا على علامات التشكيل المستخدمة والسياق واللهجة المكتوبة بها.

وبعقدنا

إذًا... أي شكل من أشكال الكلمة يجب أن نختار؟

تأتي حزمة camel-tools مع "المحلل الصرفي" 'morphological analyzer' أنيق، والذي – باختصار – يقارن أي كلمة تقدمها لها بقاعدة بيانات صرفية(تأتي مع قاعدة بيانات \صرفية واحدة) ويخرج تحليلا كاملا للأشكال والمعاني المحتملة الكلمة، بمافي ذلك الليما lemma، واقسام الكلام part of speech، والترجمة الإنجليزية إذا كانت متوفرة، وما إلى ذلك.

فيما يلي نجري تحليلًا صرفياً لكلمة "وبعقدنا".

```
from camel_tools.morphology.database import MorphologyDB
from camel_tools.morphology.analyzer import Analyzer

db = MorphologyDB.builtin_db()
analyzer = Analyzer(db)

analyses = analyzer.analyze('وبعقدنا')

for analysis in analyses:
    print(analysis, '\n')
```

لقد قمت بتضمين جزء من الناتج أدناه؛ هذه ليست سوى أعلى 3 تحليلات من بين أكثر من 20 تحليلًا يتم عرضها. لاحظ أنه أيضًا "يزيل الغموض disambiguates" عن الكلمة من خلال استعادة علامات التشكيل diacritics.

```
{'diac': 'وَبِعُقَدُنا', 'lex': 'عُقْدَة_1', 'bw':
'وَ/PART+بِ/PREP+عُقَد/NOUN+ُ/CASE_DEF_NOM+نا/POSS_PRON_1P', 'gloss':
'[part.]_+_by;with+complexes+our', 'pos': 'noun', 'prc3': '0',
'prc2': 'wa_part', 'prc1': 'bi_prep', 'prc0': '0', 'per': 'na',
'asp': 'na', 'vox': 'na', 'mod': 'na', 'stt': 'c', 'cas': 'n',
'enc0': '1p_poss', 'rat': 'i', 'source': 'lex', 'form_gen': 'm',
'form_num': 's', 'pattern': 'وَبُ123نا', 'root': 'ع.ق.د', 'catib6':
'PRT+PRT+NOM+NOM', 'ud': 'PART+ADP+NOUN+PRON', 'd1seg': 'وَبِعُقَدُنا',
'd1tok': 'وَ+_بِعُقَدُنا', 'atbseg': 'وَ+_بِ+_عُقَدُ+_نا', 'd3seg':
'وَ+_بِ+_عُقَدُ+_نا', 'd2seg': 'وَ+_بِ+_عُقَدُنا', 'd2tok': 'وَ+_بِ+_عُقَدُنا',
'atbtok': 'وَ+_بِ+_عُقَدُ+_نا', 'd3tok': 'وَ+_بِ+_عُقَدُ+_نا', 'bwtok':
'وَ+_بِ+_عُقَدُ+_نا', 'pos_lex_logprob': -4.923429, 'caphi':
'w_a_b_i_3_u_q_a_d_u_n_aa', 'pos_logprob': -0.4344233, 'gen': 'f',
'lex_logprob': -4.923429, 'num': 'p', 'stem': 'عُقَد', 'stemgloss':
'complexes', 'stemcat': 'N'}

{'diac': 'وَبِعُقَدنا', 'lex': 'عُقْدَة_1', 'bw':
'وَ/PART+بِ/PREP+عُقَد/NOUN+نا/POSS_PRON_1P', 'gloss':
'[part.]_+_by;with+complexes+our', 'pos': 'noun', 'prc3': '0',
'prc2': 'wa_part', 'prc1': 'bi_prep', 'prc0': '0', 'per': 'na',
'asp': 'na', 'vox': 'na', 'mod': 'na', 'stt': 'c', 'cas': 'u',
'enc0': '1p_poss', 'rat': 'i', 'source': 'lex', 'form_gen': 'm',
'form_num': 's', 'pattern': 'وَبِ123نا', 'root': 'ع.ق.د', 'catib6':
'PRT+PRT+NOM+NOM', 'ud': 'PART+ADP+NOUN+PRON', 'd1seg': 'وَبِعُقَدنا',
'd1tok': 'وَ+_بِعُقَدنا', 'atbseg': 'وَ+_بِ+_عُقَد+_نا', 'd3seg':
'وَ+_بِ+_عُقَد+_نا', 'd2seg': 'وَ+_بِ+_عُقَدنا', 'd2tok': 'وَ+_بِ+_عُقَدنا',
'atbtok': 'وَ+_بِ+_عُقَد+_نا', 'd3tok': 'وَ+_بِ+_عُقَد+_نا', 'bwtok':
'وَ+_بِ+_عُقَد+_نا', 'pos_lex_logprob': -4.923429, 'caphi':
'w_a_b_i_3_u_q_a_d_n_aa', 'pos_logprob': -0.4344233, 'gen': 'f',
```

```
'lex_logprob': -4.923429, 'num': 'p', 'stem': 'عُقَد', 'stemgloss':
'complexes', 'stemcat': 'N'}

{'diac': 'وَبِعُقَدِنا', 'lex': 'عُقْدَة_1', 'bw':
'وَ/PART+بِ/PREP+عُقَد/NOUN+/CASE_DEF_GEN+نا/POSS_PRON_1P', 'gloss':
'[part.]_+_by;with+complexes+our', 'pos': 'noun', 'prc3': '0',
'prc2': 'wa_part', 'prc1': 'bi_prep', 'prc0': '0', 'per': 'na',
'asp': 'na', 'vox': 'na', 'mod': 'na', 'stt': 'c', 'cas': 'g',
'enc0': '1p_poss', 'rat': 'i', 'source': 'lex', 'form_gen': 'm',
'form_num': 's', 'pattern': 'وَبِ123نا', 'root': 'ع.ق.د', 'catib6':
'PRT+PRT+NOM+NOM', 'ud': 'PART+ADP+NOUN+PRON', 'd1seg': 'وَبِعُقَدِنا',
'd1tok': 'وَ+_بِعُقَدِنا', 'atbseg': 'وَ+_بِ+_عُقَدِ_+نا', 'd3seg':
'وَ+_بِ+_عُقَدِنا', 'd2seg': 'وَ+_بِ+_عُقَدِنا', 'd2tok': 'وَ+_بِ+_عُقَدِنا',
'atbtok': 'وَ+_بِ+_عُقَدِ_+نا', 'd3tok': 'وَ+_بِ+_عُقَدِ_+نا', 'bwtok':
'وَ+_بِ+_عُقَدِ_+_نا', 'pos_lex_logprob': -4.923429, 'caphi':
'w_a_b_i_3_u_q_a_d_i_n_aa', 'pos_logprob': -0.4344233, 'gen': 'f',
'lex_logprob': -4.923429, 'num': 'p', 'stem': 'عُقَد', 'stemgloss':
'complexes', 'stemcat': 'N'}
```

الآن،في حالتي، أعمل مع أكثر من 6 ملايين تغريدة، لذا فإن إجراء تحليل كلمة بكلمة لن يكون مفيدًا أو فعالًا للغاية. بدلاً من ذلك، يمكننا استخدام أداة Morphological Disambiguator (انتبه يا أرنولد! watch out, Arnold!) للقيام بذلك نيابةً عنا. سيأخذ هذا قائمة من التوكنز tokens كمدخلات (وبالتالي كلمة tokenizer البسيطةفي الخطوة 3) ويخرج جميع الأشكال الواضحة للتوكنز. كل تحليل عبارة عن قاموس ويمكننا الوصول إلى النماذج والمكونات التي نريدها باستخدام مفاتيح القاموس dictionary keys. يتم سرد التحليلات من الأكثر احتمالا إلى الأقل احتمالا، لذا فإن الممارسة الشائعة هي تبسيط تحديد التحليل الأول باعتباره مخرجاتك.

```
from camel_tools.disambig.mle import MLEDisambiguator

# instantiate the Maximum Likelihood Disambiguator
mle = MLEDisambiguator.pretrained()

# The disambiguator expects pre-tokenized text
sentence = simple_word_tokenize('نجح بايدن في الانتخابات')

disambig = mle.disambiguate(sentence)

diacritized = [d.analyses[0].analysis['diac'] for d in disambig]
pos_tags = [d.analyses[0].analysis['pos'] for d in disambig]
lemmas = [d.analyses[0].analysis['lex'] for d in disambig]

# Print the combined feature values extracted above
for triplet in zip(diacritized, pos_tags, lemmas):
    print(triplet)
```

الذي يُخرج النموذج المُشكل، وعلامة اقسام الكلام Part-Of-Speech tag ، والليما lemma لكل كلمةفي الجملة:

```
('نَجَحَ', 'verb', 'نَجَحَ-a_1')
('بايدن', 'noun_prop', 'بايدن_0')
('فِي', 'prep', 'فِي_1')
('الانْتِخابات', 'noun', 'اِنْتِخاب_1')
```

هذا يعني أنه يمكننا، على سبيل المثال، الحصول على جميع المصطلحات الخاصة بالنص العربي باستخدام دالة مثل تلك الموجودة أدناه:

```
def get_lemmas(tokenized_text):
    disambig = mle.disambiguate(tokenized_text)
    lemmas = [d.analyses[0].analysis['lex'] for d in disambig]
    return lemmas
```

بالنسبة لمشروعي الذي يجري نمذجة الموضوع Topic Modelling، هذه هي الطريقة التي اخترتها لإجراء التوكينازيشن tokenization الخاص بي. قد تتطلب المشاريع الأخرى أساليب مختلفة باستخدام المحلل الصرفي Morphological Tokenizer، والذي سيقوم بترميز السلاسل بشكل مختلف اعتمادًا على المخطط الذي تحدده.

```
from camel_tools.tokenizers.morphological import
MorphologicalTokenizer

# atbseg scheme
tokenizer = MorphologicalTokenizer(mle, scheme='atbseg')
tokens = tokenizer.tokenize(df.tweet_text.iloc[0])
print(tokens)

# atbtok scheme
tokenizer = MorphologicalTokenizer(mle, scheme='atbtok')
tokens = tokenizer.tokenize(df.tweet_text.iloc[0])
print(tokens)

# bwtok scheme
tokenizer = MorphologicalTokenizer(mle, scheme='bwtok')
tokens = tokenizer.tokenize(df.tweet_text.iloc[0])
print(tokens)

# ...and so on...
```

لذلك... ما عليك سوى العثور على نهج tokenization / lemmatization الذي يناسب مشروعك بشكل أفضل، وتنفيذه، وبعد ذلك تكون قد انتهيت من المعالجة المسبقة للنص العربي الخاص بك!

المزيد من متعة Camel

توفر حزمة camel-tools المزيد من الميزات للمعالجة المسبقة (مثل النقل الحرفي transliteration، وتسوية يونيكود unicode normalization، وما إلى ذلك) التي قد تكون مفيدة لمشروعك. تحقق من الوثائق الكاملة لمزيد من المعلومات.

أريد أيضًا أن أشير إلى أنني قمت بنصيبي العادل من البحثفي محاولة للعثور على أفضل أداة للمعالجة المسبقة للنص العربي من أجل المعالجة اللغوية الطبيعية NLP. وعلى الرغم من وجود عدد من

الخيارات الجيدة الأخرى (مثل Farasa وMADAMIRA وStanford CoreNLP) فقد وجدت أن camel-tools هي الأكثر تنوعًا وشمولاً وسهولةفي الاستخدام. إلى جانب دوال المعالجة المسبقة الأساسية للتعامل مع التحديات المذكورة أعلاه، فإنه يأتي أيضًا مع بعض الميزات الإضافية الرائعة: يمكنه إجراء تحليل المشاعر sentiment analysis– وهذه حقًا واحدة من أروع الميزات – يمكنه تحديد اللهجة dialect التي تم بها النص مكتوبة (من أصل 25 لهجة كحد أقصى). سأستخدم بالتأكيد معرف اللهجة Dialect Identifier لإضافة "اللهجة" كميزةفي مسار المعالجة اللغوية الطبيعية الخاص بي.

المصدر:

https://towardsdatascience.com/arabic-nlp-unique-challenges-and-their-solutions-d99e8a87893d

3) مجموعات بيانات عربية مفيدة لمهندسي التعلم الآلي العاملين في المعالجة اللغوية الطبيعية Useful Arabic Datasets for Machine Learning Engineers working in NLP

مجموعات بيانات النص العربي

1) MASADER

Masader هي واجهة جديدة (كتالوج بيانات data catalogue) لاستكشاف أكثر من 500 مجموعة بيانات للمعالجة اللغوية الطبيعية للغة العربية ARABIC NLP، مصممة للاستخدام العامفي مجموعات بيانات المعالجة اللغوية الطبيعية للغة العربية والكلام.

2) Arabic Reviews Dataset

إذا كنت تتطلع إلى تدريب نموذج تحليل المشاعر sentiment analysis model، فيمكن أن تكون مجموعة البيانات هذه مفيدة لتطبيقك. تحتوي مجموعة البيانات هذه على حوالي 99,999 تقييمًا للفنادق والكتب والأفلام والمنتجات (متوازنة balanced)في 3 فئات: المراجعة المختلطة Mixed Review (5/3)، والسلبية Negative (1 أو 5/2) والإيجابية Positive (4 أو 5/5).

3) HARD-Arabic-Dataset

تتكون مجموعة بيانات HARD من 93700 تقييمًا للفنادق باللغة العربية من موقع Booking.com خلال شهري يونيو/يوليو 2016. تمت كتابة التقييمات المجمعة باللغة العربية الفصحى الحديثة بالإضافة إلى اللهجة العربية وتم تصنيفها على أنها تقييمات إيجابية positive ومحايدة neutral وسلبية negative.

4) ASTD

تتكون مجموعة بيانات ASTD (Arabic Sentiment Tweets Dataset) من أكثر من 10 آلاف تغريدة عن المشاعر العربية مصنفة إلى أربع فئات:

- موضوعي Objective.
- إيجابية ذاتية Subjective positive.
- سلبية ذاتية Subjective negative.
- مختلطة ذاتية Subjective mixed.

ArSAS (5

ArSAS (Arabic Speech-Act and Sentiment Corpus of Tweets) عبارة عن مجموعة من التغريدات العربية المشروحة لمهام التعرف على أفعال الكلام speech-act recognition وتحليل المشاعر sentiment analysis. أفعال الكلام هي أفعال لفظية تؤدي إلى إنجاز شيء ما: مثل التحية، والإهانة، والمدح، والتوسل، وتقديم المعلومات، وإنجاز العمل. تم جمع هذه التغريدات العربية المكونة من 21 ألف تغريدة والتي تغطي موضوعات متعددة، وإعدادها وتعليقها لستة فئات مختلفة من مسميات أفعال الكلام، مثل التعبير والتأكيد والسؤال. بالإضافة إلى ذلك، تم أيضًا إضافة تعليقات توضيحية لنفس المجموعة من التغريدات بأربع فئات من المشاعر.

Arabic Sentiment Twitter Corpus (6

تم جمع مجموعة البيانات هذه لتوفير مجموعة من المشاعر العربية لمجتمع البحث للتحقيقفي مناهج التعلم العميق لتحليل المشاعر العربية.

مجموعة البيانات هذه التي جمعتفي أبريل 2019. تحتوي على 58 ألف تغريدة باللغة العربية (47 ألف تدريب، 11 ألف اختبار) تغريدات مشروحة بتسميات إيجابية وسلبية. تمت موازنة مجموعة البيانات وتم جمعها باستخدام معجم الرموز التعبيرية الإيجابية والسلبية.

Jamalon Arabic Books Dataset (7

مجموعة بيانات لمكتبة إلكترونية تحتوي على أكثر من 8000 كتاب باللغتين العربية والإنجليزية يمكن استخدامها لتدريب نماذج المعالجة اللغوية الطبيعية للغة العربية. تحتوي مجموعة البيانات على البيانات التالية:

- العنوان Title.
- المؤلف Author.
- الوصف Description.
- الصفحات Pages.
- سنة النشر Publication year.
- الناشر Publisher.
- الغطاء Cover.
- الفئة Category.
- التصنيف الفرعي Subcategory.
- السعر Price.

8] Arabizi

تتكون مجموعة بيانات Arabizi من نوعين من الموارد:

أ) مجموعة من النصوص الإنجليزية والعربي المختلطة تهدف إلى تدريب واختبار نظام للكشف التلقائي عن تبديل الرموزفي النصوص الإنجليزية والعربي المختلطة، حيث يتم تصنيف بيانات التدريب والاختبار يدويًا على أنها English ("e") و"a") Arabizi، أو other ("o").

ب) مجموعة مكونة من 3,452 رمز Arabizi مترجمة يدويًا إلى اللغة العربية، ومجموعة مكونة من 127 تغريدة Arabizi تحتوي على 1,385 كلمة مترجمة يدويًا أيضًا إلى اللغة العربية. تهدف مجموعة البيانات هذه إلى تدريب واختبار نظام يقوم بالترجمة الصوتية من العربية إلى العربية.

9] Arabic Poetry Dataset (6th — 21st century)

الشعر العربي هو أقدم وأبرز أشكال الأدب العربي اليوم. ربما يكون الشعر العربي القديم هو المصدر الأساسي لوصف الحياة الاجتماعية والسياسية والفكريةفي العالم العربي. لقد مر الشعر الحديث بتغيرات وتحولات كبيرة سواءفي الشكل أوفي المواضيع.

تحتوي مجموعة البيانات على أكثر من 58 ألف قصيدة تمتد من القرن السادس حتى يومنا هذا. إلى جانب كل قصيدة، تم أيضًا استخراج البيانات الوصفية للقصيدة مثل اسم الشاعر والقصيدة وفئتها. تم استخراج البيانات من adab.com.

10] Arabic Learner Corpus (ALC)

مجموعة بيانات أخرى للمعالجة اللغوية الطبيعية للغة العربية، تضم مجموعة متعلم اللغة العربية Arabic Learner Corpus (ALC) أكثر من 900 طالبفي المملكة العربية السعودية يساهمونفي مجموعة البيانات.

11] Arabic BERT Dataset

تتكون هذه المجموعة من ملفات مجزأة من ويكيبيديا العربية وأكثر من 1000 كتاب، تم تنظيفها وتنسيقها لتناسب تطبيق Nvidia PyTorch لـ BERT.

12] OntoNotes

مجموعة البيانات هذه عبارة عن مجموعة من النصوص باللغة العربية ولغات أخرى مثل الإنجليزية والصينية، من مجموعة متنوعة من المصادر مثل الأخبار العربية وخدمة العملاء والمحادثات الهاتفية وما إلى ذلك.

13) Arabic Question Answering Datasets

يحتوي مستودع Github التالي على مجموعات بيانات متعددة مثل Arab SquAD وARCD وغيرها. عند استخدام مجموعات البيانات المختلفة تأكد من الاستشهاد بها بشكل صحيح.

مجموعات البيانات المكتوبة بخط اليد للتعرف البصري على الحروف العربية

1) Arabic Handwritten Digits

هذه مجموعة مكتوبة بخط اليد من قبل 700 كاتب وتحتوي على 60,000 صورة تدريبية و10,000 صورة اختبارية للمعالجة اللغوية الطبيعية للغة العربية Arabic NLP، حيث كتب كل كاتب كل رقم عشر مرات.

2) Arabic Handwritten Characters Dataset

تحتوي مجموعة البيانات هذه على 16800 حرفًا مكتوبة بخط اليد من قبل 60 شخصًا تتراوح أعمارهم بين 19 إلى 40 عامًافي المعالجة اللغوية الطبيعية للغة العربية. 10٪ فقط من الكتاب كانوا أعسر. الصور لديها دقة 300 نقطةفي البوصة.

3) Yarmouk Arabic OCR

تضم مجموعة البيانات العربية هذه 8994 صورة من 4587 مقالة مكتوبة باللغة العربية على ويكيبيديا.

المصدر:

https://medium.com/@amnahhmohammed/useful-arabic-datasets-for-machine-learning-engineers-working-in-nlp-d06ba6c5e96d

4) تحليل المشاعر العربية باستخدام التعلم الآلي Arabic Sentiment Analysis using Machine Learning

يعد تحديد وتصنيف الآراء المعبر عنهافي جزء من النص (المعروف أيضًا باسم تحليل المشاعر sentiment analysis) أحد أكثر المهام التي يتم إجراؤهافي المعالجة اللغوية الطبيعية. اللغة العربية، على الرغم من كونها واحدة من أكثر اللغات انتشارافي العالم، لا تحظى باهتمام كبير فيما يتعلق بتحليل المشاعر. ولذلك فإن هذه المقالة مخصصة لتطبيق تحليل المشاعر العربية Arabic Sentiment Analysis (ASA) باستخدام بايثون.

مجموعة البيانات

تتكون مجموعة البيانات المستخدمةفي هذه المقالة من 1800 تغريدة مصنفة على أنها إيجابية positive وسلبية negative. ويمكن العثور عليها هنا.

استيراد المكتبة واستكشاف البيانات

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import string
import re
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score,
classification_report

data = pd.read_excel(r"C:\Users\ibrom\Desktop\NOTEBOOK\NLP
PRACICE\AJGT.xlsx")
print(data.head())
print(data.sample(5))
```

	ID	Feed	Sentiment
0	1	...اريد فيها جامعات اكثر من عمان ... وفيها قد عم	Positive
1	2	... الحلو انكم بتحكوا على اساس انو الاردن ما فيه	Negative
2	3	كله رافع بجد ربنا يكرمك	Positive
3	4	لسانك قذر يا همامه	Negative
4	5	...انا داشره وغير متزوجه ولدي علاقات مشبوه واحشش	Negative

	ID	Feed	Sentiment
833	834	سبحان الله خالق السماوات الارض	Positive
813	814	ركز في الاجابه وقول يارب وخد قرار	Positive
935	936	صباح ملؤه الحب العرفان	Positive
38	39	ادعي من الله ان يقوني علي طاعته	Positive
1336	1337	ما في احلى من انه الواحد يوكل ويدام	Positive

	ID	Feed	Sentiment
833	834	سبحان الله خالق السماوات الارض	Positive
813	814	ركز في الاجابه وقول يارب وخد قرار	Positive
935	936	صباح ملؤه الحب العرفان	Positive
38	39	ادعي من الله ان يقوني علي طاعته	Positive
1336	1337	ما في احلى من انه الواحد يوكل ويدام	Positive

	ID	Feed	Sentiment
833	834	سبحان الله خالق السماوات الارض	Positive
813	814	ركز في الاجابه وقول يارب وخد قرار	Positive
935	936	صباح ملؤه الحب العرفان	Positive
38	39	ادعي من الله ان يقوني علي طاعته	Positive
1336	1337	ما في احلى من انه الواحد يوكل ويدام	Positive


```
Positive    900
Negative    900
Name: Sentiment, dtype: int64
```

لدينا فئات متوازنة balanced classes للغاية هنا.

المعالجة المسبقة للنص

باعتباري شخصًا معتادًا على العمل مع النصوص الإنجليزية، وجدت صعوبةفي المقام الأولفي ترجمة خطوات المعالجة المسبقة المستخدمة بشكل روتيني للنصوص الإنجليزية إلى اللغة العربية. لحسن الحظ، وجدت لاحقًا مستودع Github يحتوي على كود لتنظيف النصوص باللغة العربية. تتضمن الخطوات بشكل أساسي إزالة علامات الترقيم punctuation وعلامات التشكيل العربية Arabic diacritics (حروف العلة القصيرة short vowels والحركات harakahs الأخرى) والاستطالة elongation وكلمات التوقف stopwords (المتوفرةفي مجموعة NLTK (NLTK corpus)).

```
'''
The first step is to subject the data to preprocessing.
This involves removing both arabic and english punctuation
Normalizing different letter variants with one common letter
'''
# first we define a list of arabic and english punctiations that we
want to get rid of in our text

punctuations = '''`÷×؛<>_()*&^%][ـ-"…”!|+¦~{}',.؟":/،ـ''' +
string.punctuation

# Arabic stop words with nltk
stop_words = stopwords.words()

arabic_diacritics = re.compile("""
                                 ّ| # Shadda
                                 َ| # Fatha
                                 ً| # Tanwin Fath
                                 ُ| # Damma
                                 ٌ| # Tanwin Damm
                                 ِ| # Kasra
                                 ٍ| # Tanwin Kasr
                                 ْ| # Sukun
                                  ـ# Tatwil/Kashida
                           """, re.VERBOSE)

def preprocess(text):

    '''
    text is an arabic string input

    the preprocessed text is returned
    '''

    #remove punctuations
    translator = str.maketrans('', '', punctuations)
    text = text.translate(translator)

    # remove Tashkeel
    text = re.sub(arabic_diacritics, '', text)

    #remove longation
    text = re.sub("ا" ,"[إأآا]", text)
    text = re.sub("ي" ,"ى", text)
    text = re.sub("ء" ,"ؤ", text)
    text = re.sub("ء" ,"ئ", text)
    text = re.sub("ه" ,"ة", text)
    text = re.sub("ك" ,"گ", text)

    text = ' '.join(word for word in text.split() if word not in
stop_words)

    return text

  data['Feed'] = data['Feed'].apply(preprocess)
```

```
print(data.head(5))
```

	Feed	Sentiment
0	...اريد جامعات اكثر عمان وفيها عمان ونص لعيبه الم	1
1	الحلو انكم بتحكوا علي اساس انو الاردن فساد سرقات	0
2	كله راءع بجد ربنا يكرمك	1
3	لسانك قذر قمامه	0
4	...انا داشره وغير متزوجه ولدي علاقات مشبوه واحشش	0

تحليل المشاعر بتقنيات مختلفة

الهدف من هذه المقالة هو توضيح كيفية استخدام تقنيات استخلاص المعلومات information extraction المختلفةفي تحليل المشاعر SA. ولكن من أجل التبسيط، سأوضح فقط تحويل الكلمات (أي tf-idf) هنا. كما هو الحال مع أي مهمة تعليمية خاضعة للإشراف، يتم تقسيم البيانات أولاً إلى ميزات (Feed) وتصنيف (Sentiment). بعد ذلك، يتم تقسيم البيانات إلى مجموعات تدريب train واختبار test، ويتم تنفيذ مصنفات مختلفة بدءًا من الانحدار اللوجستي Logistic Regression.

الانحدار اللوجستي

الانحدار اللوجستي هو خوارزمية تصنيف شائعة جدًا. إنها سهلة التنفيذ ويمكن أن تكون بمثابة خوارزمية أساسية لمهام التصنيف classification. من أجل جعل الكود أقصر، يتم استخدام كلاس Pipeline في Scilkit-Learn والتي تجمع بين التوجيه vectorization والتحويل transformation والبحث الشبكي gridsearch والتصنيف. يمكنك قراءة المزيد عن البحث الشبكيفي الوثائق الرسمية هنا.

```
# splitting the data into target and feature
feature = data.Feed
target = data.Sentiment
# splitting into train and tests
X_train, X_test, Y_train, Y_test = train_test_split(feature,
target, test_size =.2, random_state=100)

# make pipeline
pipe = make_pipeline(TfidfVectorizer(),
                     LogisticRegression())
# make param grid
param_grid = {'logisticregression__C': [0.01, 0.1, 1, 10, 100]}

# create and fit the model
model = GridSearchCV(pipe, param_grid, cv=5)
model.fit(X_train,Y_train)

# make prediction and print accuracy
prediction = model.predict(X_test)
print(f"Accuracy score is {accuracy_score(Y_test,
prediction):.2f}")
```

```
print(classification_report(Y_test, prediction))
```

```
Accuracy score is 0.84
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       176
           1       0.83      0.86      0.84       184

    accuracy                           0.84       360
   macro avg       0.84      0.84      0.84       360
weighted avg       0.84      0.84      0.84       360
```

تم تحقيق دقة 84٪

مصنف الغابات العشوائية

```
pipe = make_pipeline(TfidfVectorizer(),
                     RandomForestClassifier())

param_grid = {'randomforestclassifier__n_estimators':[10, 100,
1000],
              'randomforestclassifier__max_features':['sqrt',
'log2']}

rf_model = GridSearchCV(pipe, param_grid, cv=5)
rf_model.fit(X_train,Y_train)

prediction = rf_model.predict(X_test)
print(f"Accuracy score is {accuracy_score(Y_test,
prediction):.2f}")
```

```
Accuracy score is 0.85
              precision    recall  f1-score   support

           0       0.82      0.89      0.85       176
           1       0.88      0.81      0.84       184

    accuracy                           0.85       360
   macro avg       0.85      0.85      0.85       360
weighted avg       0.85      0.85      0.85       360
```

مُصنف Naive Bayes (متعدد الحدود)

```
pipe = make_pipeline(TfidfVectorizer(),
                     MultinomialNB())
pipe.fit(X_train,Y_train)
prediction = pipe.predict(X_test)
print(f"Accuracy score is {accuracy_score(Y_test,
prediction):.2f}")
print(classification_report(Y_test, prediction))
```

```
Accuracy score is 0.84
              precision    recall  f1-score   support

           0       0.89      0.76      0.82       176
           1       0.80      0.91      0.85       184

    accuracy                           0.84       360
   macro avg       0.85      0.84      0.84       360
weighted avg       0.85      0.84      0.84       360
```

آلة المتجهات الداعمة

```
pipe = make_pipeline(TfidfVectorizer(),
                     SVC())
param_grid = {'svc__kernel': ['rbf', 'linear', 'poly'],
              'svc__gamma': [0.1, 1, 10, 100],
              'svc__C': [0.1, 1, 10, 100]}

svc_model = GridSearchCV(pipe, param_grid, cv=3)
svc_model.fit(X_train, Y_train)

prediction = svc_model.predict(X_test)
print(f"Accuracy score is {accuracy_score(Y_test,
prediction):.2f}")
print(classification_report(Y_test, prediction))
```

```
Accuracy score is 0.85
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       176
           1       0.87      0.83      0.85       184

    accuracy                           0.85       360
   macro avg       0.85      0.85      0.85       360
weighted avg       0.85      0.85      0.85       360
```

الاستنتاج

توضح هذه المقالة الخطوات المتبعةفي تحليل المشاعر العربية. الفرق الرئيسي بين المعالجة اللغوية الطبيعية (NLP) العربية والإنجليزية هو خطوة المعالجة المسبقة. أعطت جميع المصنفات المجهزة درجات دقة مذهلة تتراوح من 84 إلى 85%.في حين أعطى Naive Bayes والانحدار اللوجستي والغابة العشوائية دقة تبلغ 84%، فقد تم تحقيق تحسن بنسبة 1% باستخدام آلة المتجهات الداعمة الخطية. يمكن تحسين النماذج بشكل أكبر من خلال تطبيق تقنيات مثل تضمين الكلمات word embedding والشبكات العصبية المتكررة recurrent neural networks والتي سأحاول تنفيذها في مقالة لاحقة.

المصدر:

https://towardsdatascience.com/arabic-sentiment-analysis-5e21b77fb5ea

5) تحليل المشاعر لبيانات النص العربي (التغريدات)
Sentiment Analysis of Arabic Text Data (Tweets)

المقدمة

قامت الشركة بجمع مجموعة البيانات هذه لتوفير مجموعة من المشاعر العربية Arabic sentiment corpus للبحث الذي تجريه الشركة للتحقيقفي أساليب التعلم العميق deep learning لتحليل المشاعر العربية Arabic sentiment analysis.

لذا، من هذا المشروع، أحصل على فرصة للعملفي مجال تحليل المشاعر. كما أنه سيكون بالتأكيد مفيدًا لشركتي الناشئة. لأنه أثناء التعامل مع تقييمات العملاء، نريد تفسير ما يميل المستخدم إلى تصويره حتى نتمكن من تقديم أفضل النتائج الموصى بها.

وبصرف النظر عن هذا، كان تحليل المشاعر مجالًا مثيرًا للدراسة. لا يزال هذا موضوعًا قيد التطوير، وله وظائف معقدة جدًا بحيث لا يمكن للآلات فهمها مثل السخرية sarcasm والمشاعر السلبية negative emotions والمبالغة hyperbole وما إلى ذلك.

ولأنني جزء من هذه الصناعة، فأنا أعرف الإمكانات الكامنةفي تحليل المشاعر. ويضيف الكثير من القيمة لهذه الصناعة. يبني تحليل المشاعر نتائجه على عوامل إنسانية بطبيعتها، ومن المحتم أن تصبح أحد المحركات الرئيسية للعديد من قرارات الأعمالفي المستقبل.

فهم الأعمال والبيانات

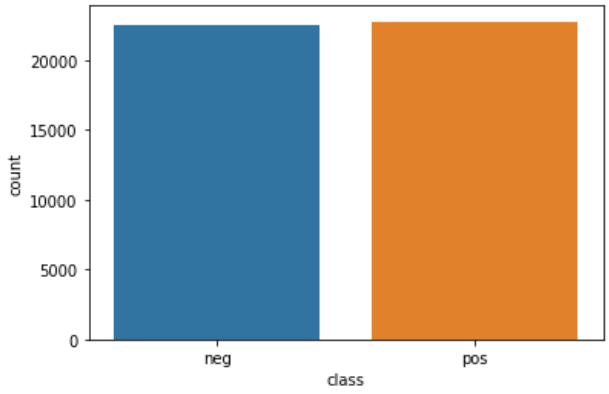
قامت الشركة بجمع مجموعة البيانات هذه لتوفير مجموعة من المشاعر العربية للبحث الذي تجريه الشركة للتحقيقفي أساليب التعلم العميق لتحليل المشاعر العربية.

تقوم مجموعة بيانات التغريدات العربية Arabic Tweets Dataset لدينا بتقسيم التغريدات إلى فئتين إيجابية Positive وسلبية negative.

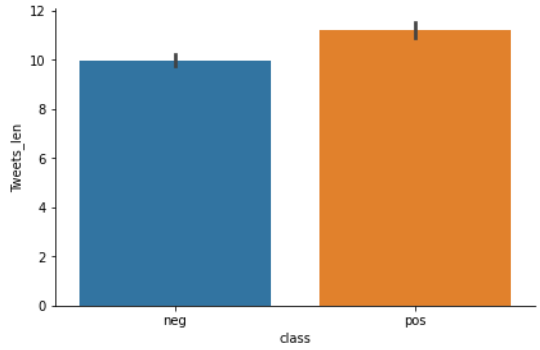
أول شيء يجب عليك فعله هو تحديد السلوك الذي تنتمي إليه التغريدة.

لذلك، بناءً على محتوى النص يخبرنا من نص التغريدات إذا كان إيجابيًا أم سلبيًا، إذا كانت الرموز التعبيرية والنص بها مشاعر إيجابية، فسيتم تصنيفها على أنها إيجابية، وإلا فهي سلبية.

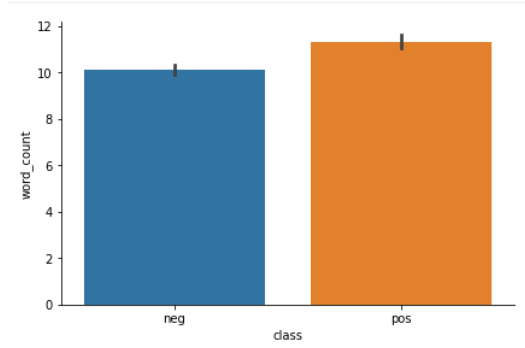
وبناء على ذلك ستقوم الشركة بتقسيمها إلى إيجابية أو سلبية كما قلنا سابقا بناء على النص والعواطف المضمنة فيه.



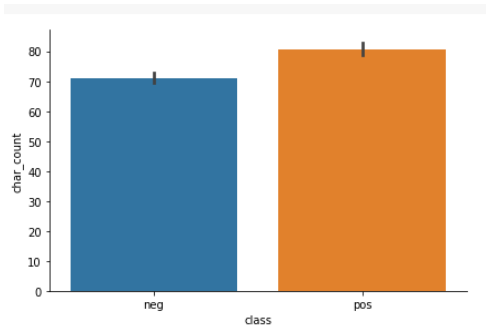
1. **طول التغريدات Length of tweets:** طول الكلماتفي كل تغريدة.



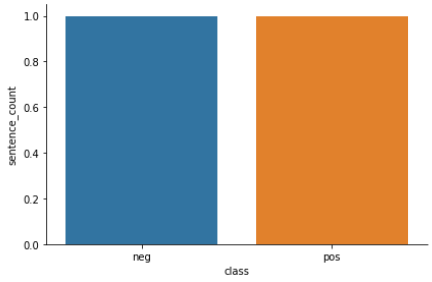
2. **عدد الكلمات Number of words count**: عدد الأحرففي كل تغريدة.



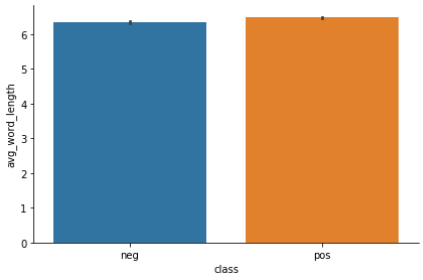
3. **عدد الأحرف Number of char**: عدد الجملفي كل تغريدة.



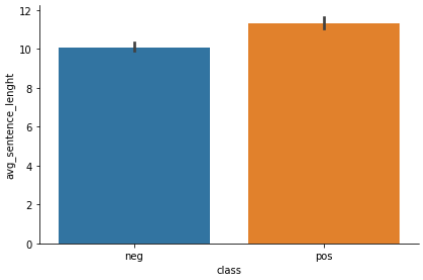
4. عدد الجمل **Number of sentences**: متوسط طول الكلماتفي التغريدات.



5. متوسط طول الكلمات **Average words length**: متوسط طول الجملفي التغريدات.

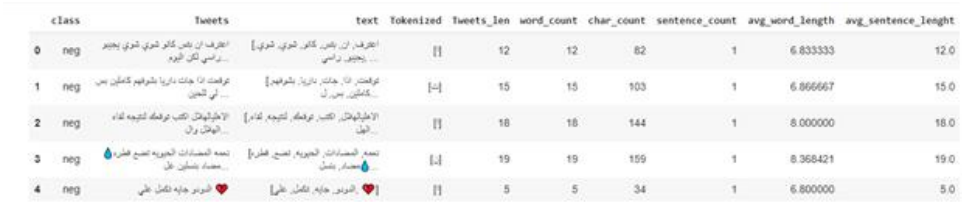


6. متوسط طول الجملة **Average sentence length**: متوسط طول الجملفي التغريدات.



هندسة الميزات

لكيلا ندفع أي خوارزمية أخرى إلى الحد الأقصىفي نموذج البيانات الحالي، فلنحاول إضافة بعض الميزات التي قد تساعدفي تصنيف التغريدات.



	class	Tweets	text	Tokenized	Tweets_len	word_count	char_count	sentence_count	avg_word_length	avg_sentence_lenght
0	neg	اظرف ان بقى كاكو شوي شوي يجيبر ... راسي لكن اليوم	اظرف، ان، بقى، كاكو، شوي، شوي، ... يجيبر، راسي	[]	12	12	82	1	6.833333	12.0
1	neg	عرفت اذا جات داريا بشوفهم كاملين بس ... لي تامين	عرفت، اذا، جات، داريا، بشوفهم، ... كاملين، بس، ل	[--]	15	15	103	1	6.866667	15.0
2	neg	الاخليهاهان اكتب عرفتك لتيجه لقاء ... الهان وان	الاخليهاهان، اكتب، عرفتك، لتيجه، لقاء ... الهان	[]	18	18	144	1	8.000000	18.0
3	neg	نعمه العصارات الجويه تصبح فطره ... معصار بتسلين عل	نعمه، العصارات، الجويه، تصبح، فطره ... معصار، بتسل	[,]	19	19	159	1	8.368421	19.0
4	neg	الدوبر جايه اكامل علي	الدوبر، جايه، اكامل، علي	[]	5	5	34	1	6.800000	5.0

اسماء الميزات

1. طول التغريدات Length of tweets.
2. عدد الكلمات Number of words count.
3. عدد الأحرف Number of char.
4. عدد الجمل Number of sentences.
5. متوسط طول الكلمات Average words length.
6. متوسط طول الجملة Average sentence length.

التوضيح

1. طول الكلماتفي كل تغريدة.
2. عدد أعداد كل كلمةفي التغريدات.
3. عدد الأحرففي كل تغريدة.
4. عدد الجملفي كل تغريدة.
5. متوسط طول الكلماتفي التغريدات.
6. متوسط طول الجملفي التغريدات.

مكتبات بايثون المستخدمة

تمت معالجة البيانات مسبقًا باستخدام مكتبات pandas وgensim وnumpy وتم إنشاء عملية التعلم/التحقق من الصحة باستخدام scikit-learn. تم إنشاء المخططات باستخدام Seaborn.

```
import pandas as pd
import numpy as np
from collections import Counter
import nltk
import pandas as pd
import re as regex
import numpy as np
import plotly
from plotly import graph_objs
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, RandomizedSearchCV
from time import time
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
import plotly
import cufflinks as cf
import re
nltk.download('punkt')
```

مجموعات البيانات والمدخلات

تتكون بيانات الإدخال من ملفين CSV:

1. Train.csv (45000 تغريدة).
2. test.csv (45000 تغريدة).

واحد للتدريب والآخر للاختبار. كان تنسيق البيانات كما يلي (تحتوي بيانات الاختبار على عمود الفئة Class column):

تحتوي مجموعة البيانات على الميزات التالية:

بيانات التدريب

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45275 entries, 0 to 45274
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   class   45275 non-null  object
 1   Tweets  45275 non-null  object
dtypes: object(2)
memory usage: 707.5+ KB
```

	class	Tweets
0	neg	اعترف ان بعض كانو شوي شوي يجيبو راسي لكن اليوم...
1	neg	توقعت اذا جات داريا بشوفهم كاملين بس لي للحين ...
2	neg	الاهلي_الهلال اكتب توقعك لنتيجة لقاء الهلال و#...
3	neg	نعمة المضادات الحيوية . تضع قطرة💧مضاد بنسلين ع...
4	neg	💔 الدودو جابه تكمل علي

هنا، "class" هي الفئة المستهدفة، وبالنظر إلى عمود "Tweets"، تحدد "class" ما إذا كانت تغريدة المستخدم المحددة إيجابية أم سلبية.

بيانات الاختبار

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11520 entries, 0 to 11519
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   class   11520 non-null  object
 1   Tweets  11520 non-null  object
dtypes: object(2)
memory usage: 180.1+ KB
```

نظرًا لأن ملف test.csv كان مليئًا بالإدخالات الفارغة empty entries، فستتم إزالتها.

	class	Tweets
0	neg	حتى الايتونز خربتوه مو صاحين انتو؟؟ 😭
1	neg	واحد تبع النظام السوري يقول أن المخابرات السور...
2	neg	الى متى التعامل السئ للخادمات وعدم احترامهم وك...
3	neg	رايح جاي ي طحلبي 🐸 #الهلال_الاهلي
4	neg	تتمغط ومعها سداع 😠

المعالجة المسبقة

المعالجة المسبقة للبيانات Data preprocessing هي تقنية للتنقيبفي البيانات data mining تتضمن تحويل البيانات الأولية إلى تنسيق مفهوم. غالبًا ما تكون بيانات العالم الحقيقي غير مكتملة وغير متسقة و/أو تفتقر إلى سلوكيات أو اتجاهات معينة، ومن المحتمل أن تحتوي على العديد من الأخطاء. تعد المعالجة المسبقة للبيانات طريقة مجربة لحل مثل هذه المشكلات. تقوم المعالجة المسبقة للبيانات بإعداد البيانات الأولية لمزيد من المعالجة.

الهدف من المعالجة المسبقة التالية هو إنشاء تمثيل للبيانات. سيتم تنفيذ الخطوات على النحو التالي:

1. التنظيف Cleaning
   - إزالة عناوين URL (Remove URLs).
   - إزالة علامات الترقيم (Remove punctuatuions).
   - إزالة الاستطالة (Remove longation).
   - إزالة أسماء المستخدمين Remove usernames (الإشارات mentions).

- إزالة الأحرف الخاصة Remove special characters.
- إزالة الأرقام Remove numbers.

2. معالجة النصوص Text processing
   - التوكينايزر Tokenize.
   - التجذيع Stemming.

تنظيف البيانات

يعد تنظيف البيانات Data cleaning أحد الأجزاء المهمة لإعداد البيانات لتمثيل حقيبة الكلمات Bag-of-word representation. بعد التنظيف:

```
] fulldf['Tweets'].head()

0    اعترف ان بنس كانو شوي شوي يجيبو راسي لكن اليوم...
1    توقعت اذا جات داريا بتوقفهم كاملين بس لي للحين ...
2    الاهلياالهلال اكتب توقعك لنتيجه لقاء الهلال وال...
3    نعمه المضادات الحيويه تضع قطره💧مضاد بنسلين عل...
4                                  الدودو جايه تكمل علي 💔
Name: Tweets, dtype: object
```

التوكينازيشن والتجذيع

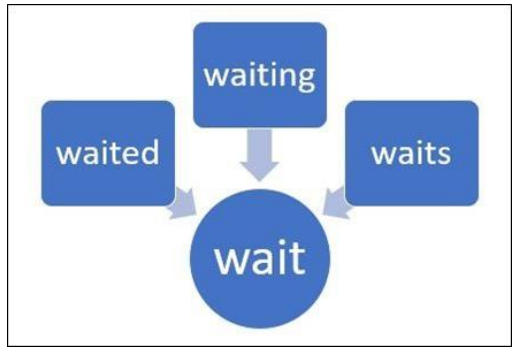
يتكون الترميز Tokenization من تقسيم النص إلى كلمات، وكلماتفي سياق يحتوي على مسافات وعلامات ترقيم وحالات وعلامات تشكيل وعلامات تشكيل وما إلى ذلك ــ إلى كلمات موحدة. هذه الخطوة حاسمة بالنسبة لدقة التدفق بأكملها

يتكون التجذيع Stemming من تحضير تعبيرات الكلمات للعثور على أصولها. تعتمد العملية على قاموس لاحقة مما يجعل من الممكن استخراج الجذور stems بعد تحليل شكل (صرف) morphology الكلمة.

وفقًا للأشكال التصريفية المحددة واللغة المحددة، فإنه يحسب المصادر الأكثر صلة من القواعد النحوية والسياقية للغة.

يقدم التجذيع فائدتين رئيسيتين:

- نظرًا لأنه يركز على أصول الكلمات، فإن العملية متسامحة تمامًا مع الأخطاء الإملائية.
- إنها تحتاج فقط إلى الكلمات لتتبع دون الحاجة إلى سياق استخدامها. الشكل أدناه هو المثال على التجذيع stemming:



لمعالجة النصوص، يتم استخدام مكتبة nltk. أولاً، يتم ترميز التغريدات باستخدام nltk.word_tokenize وبعد ذلك، تتم عملية التجذيع باستخدام Porter Stemmer حيث أن التغريدات باللغة العربية بنسبة 100%.

	class	Tweets	text	Tokenized
0	neg	...اعترف ان بنص كانو شوي شوي يجيبو راسي لكن اليوم	... ,اعترف, ان, بنص, كانو, شوي, شوي, يجيبو, راسي]	[ا]
1	neg	... توقعت اذا جات داريا بشوفهم كاملين بس لي للحين	...توقعت, اذا, جات, داريا, بشوفهم, كاملين, بس, ل]	[ت]
2	neg	...الاهليالهلال اكتب توقعك لتتيجه لقاء الهلال وال	...الاهليالهلال, اكتب, توقعك, لتتيجه, لقاء, الهل]	[ا]
3	neg	...نعمه المضادات الحيويه تضع قطره💧مضاد بتسلين عل	...نعمه, المضادات, الحيويه, تضع, قطره💧مضاد, بتسل]	[ن]
4	neg	💔 الدودو جايه تكمل علي	[💔 ,الدودو, جايه, تكمل, علي]	[ا]

قائمة الكلمات

نموذج حقيبة الكلمات bag-of-words، أو BoW للاختصار، هو وسيلة لاستخراج الميزات من النص لاستخدامهافي النمذجة، كما هو الحال مع خوارزميات التعلم الآلي.

هذا النهج بسيط للغاية ومرن ويمكن استخدامه بعدة طرق لاستخراج الميزات من المستندات.

يتم إنشاء قائمة الكلمات wordlist (القاموس dictionary) من خلال عدد بسيط من مرات ظهور كل كلمة فريدة عبر مجموعة بيانات التدريب بأكملها.

النتائج

التجربة 1: الانحدار اللوجستي

من الجميل أن نرى نوع النتائج التي قد نحصل عليها من هذا النموذج البسيط. يبدو أن مُصنف الانحدار اللوجستي Logistic Regression عبارة عن خوارزمية رائعة لبدء التجارب.

ستعتمد التجربة على تدريب: اختبار 7:3: اختبار التقسيم الطبقي stratified split.

```
Accuracy score is 0.75
              precision    recall  f1-score   support

         neg       0.71      0.83      0.77      4480
         pos       0.80      0.67      0.73      4575

    accuracy                           0.75      9055
   macro avg       0.76      0.75      0.75      9055
weighted avg       0.76      0.75      0.75      9055
```

التجربة 2: مصنف نايف بايز

كمحاولة ثانية للتصنيف، سيتم استخدام مصنف Naïve Bayes.

لا يبدو أفضل، وليس أفضل من مصنف الانحدار اللوجستي.

```
Accuracy score is 0.74
              precision    recall  f1-score   support

         neg       0.77      0.69      0.72      4480
         pos       0.72      0.80      0.76      4575

    accuracy                           0.74      9055
   macro avg       0.74      0.74      0.74      9055
weighted avg       0.74      0.74      0.74      9055
```

يمكننا أن نلاحظ انخفاض مستوى الاسترجاع recall level لمصنف الانحدار اللوجستي للفئة السلبية negative class، والذي قد يكون ناجما عن انحراف البيانات data skewness.

من التجارب المذكورة أعلاه يمكننا أن نستنتج أن مصنف الانحدار اللوجستي يعطي نتيجة أفضل من النماذج الأخرى ومن ثم سيتم تصنيف بيانات الاختبار Test Data على الانحدار اللوجستي.

تصنيف بيانات الاختبار

بعد العثور على أفضل مصنف، قم بتحميل بيانات الاختبار وتوقع المشاعر تجاهها. سيتم تصدير البيانات إلى ملف CSV بتنسيق يحتوي على عمودين: الفئة Class، والتغريدات Tweets. هناك 45000 عينة اختبار مع توزيع معروف لتسميات المشاعر sentiment labels للمقارنة بين نتائجك والنتائج المقدمة.

الاستنتاج

إن زيادة مواقع المدونات الصغيرة مثل تويتر توفر فرصة لا مثيل لها لتشكيل واستخدام الأساليب والتقنيات التي تبحث عن المشاعر وتستخرجها. يحدد العمل المقدمفي هذه الورقة منهجًا لتحليل المشاعر على بيانات تويتر العربية. وللكشف عن المشاعر، استخرجنا البيانات ذات الصلة من التغريدات، وأضفنا الميزات.

ثم تم حساب الشعور العام بالتغريدة باستخدام النموذج المعروضفي هذا التقرير. هذا العمل استكشافي بطبيعته والنموذج الأولي الذي تم تقييمه هو نموذج أولي.

أظهرت النماذج أن التنبؤ بمشاعر النص يعد مهمة غير سهلة للتعلم الآلي. هناك حاجة إلى الكثير من المعالجة المسبقة فقط لتتمكن من تشغيل الخوارزمية. المشكلة الرئيسية لتحليل المشاعر هي صياغة التمثيل الآلي للنص. تم إنشاء الكثير من الميزات الإضافية بناءً على المنطق السليم (طول الكلمات، عدد الأحرف، عدد الجمل، إلخ). أعتقد أنه يمكن تطوير تحسن طفيففي دقة التصنيف لمجموعة بيانات التدريب المحددة، ولكن بما أنها تضمنت بيانات شديدة الانحراف (عدد صغير من الحالات السلبية)، فمن المحتمل أن يكون الفرقفي حدود نسب مئوية قليلة. الشيء الذي يمكن أن يعزز نتائج التصنيف هو إضافة الكثير من الأمثلة الإضافية (زيادة مجموعة بيانات التدريب)، لأنه من الواضح أن 45275 مثالًا لا تتضمن كل تسلسل الكلمات المستخدمة، علاوة على ذلك ــ من المؤكد أن الكثير من المعلومات التي تعبر عن المشاعر مفقودة.

المصدر:

https://medium.com/analytics-vidhya/sentiment-analysis-of-arabic-text-data-tweets-4e96c8da892b

6) توليد القصائد العربية باستخدام التعلم العميق Generate Arabic Poems using Deep Learning

في هذه المقالة سيتم تقديم نموذج PyTorch RNN لتوليد القصائد العربية Arabic poems.

لنقم أولاً بتحميل مكتباتنا المطلوبة لتحميل البيانات وإنشاء النماذج.

```
import numpy as np
import torch
from torch import nn
import torch.nn.functional as F
```

تحميل البيانات

بعد ذلك، سنقوم بتحميل الملف النصي للأشعار العربية Arabic poems text file وتحويله إلى معالجة مسبقة لتستخدمه شبكتنا.

```
# open text file and read in data as `text`
with open('all_data.txt', 'r',encoding="utf-8") as f:
    text = f.read()
```

دعونا نتحقق من أول 100 حرف، ونتأكد من أن كل شيء صحيح.

```
text[:100]
```

```
'تميس فلا \n\nوجادت بالزيارة والوصال\nبدت تختال في حُلل الجمالِ\nبم\n\nوإن ترنو تداعب بالنصال\nيعادلها قضيب'
```

يبدو وكأنه معمارية قصيدة عربية نموذجية.

التوكينازيشن

في الخلايا أدناه، أقوم بإنشاء قاموسين لتحويل الأحرف من وإلى الأعداد الصحيحة.

```
encode the text and map each character to an integer and vice versa

# we create two dictionaries:
# 1. int2char, which maps integers to characters
# 2. char2int, which maps characters to unique integers
chars = tuple(set(text))
int2char = dict(enumerate(chars))
char2int = {ch: ii for ii, ch in int2char.items()}

# encode the text
encoded = np.array([char2int[ch] for ch in text])
```

الآن، يمكننا أن نرى الجزء أعلاه من القصيدة مشفرة بالأرقام.

```
encoded[:100]
```

```
array([13, 54, 25, 67, 25, 27, 25, 20,  6, 67, 30, 12, 67, 74,  2,
6,  6,
       67, 20,  6, 28,  0, 20,  6, 37, 39, 39, 38, 28, 20, 54, 25,
67, 13,
       20,  6, 72, 12, 20, 42, 66, 67, 38, 20,  6, 38,  8, 20,  6,
39, 39,
       25,  0, 12, 22, 67, 30,  6, 20, 67, 12, 47, 20, 54,  6, 76,
20, 67,
       36, 31, 12, 13, 39, 39, 38, 53, 46, 67, 25, 42, 46, 38, 67,
25, 54,
       20, 47, 13, 67, 13, 20,  6, 46,  8, 20,  6, 39, 39, 13,  0])
```

المعالجة المسبقة للبيانات

في char-RNN، يتوقع LSTM إدخالاً مشفرًا بترميز واحد ساخن one-hot encoded.

```
def one_hot_encode(arr, n_labels):

    # Initialize the the encoded array
    one_hot = np.zeros((arr.size, n_labels), dtype=np.float32)

    # Fill the appropriate elements with ones
    one_hot[np.arange(one_hot.shape[0]), arr.flatten()] = 1.

    # Finally reshape it to get back to the original array
    one_hot = one_hot.reshape((*arr.shape, n_labels))

    return one_hot
```

عمل دفعات تدريبية صغيرة

للتدريب على هذه البيانات، نريد أيضًا إنشاء دفعات صغيرة mini-batches للتدريب.

في هذا المثال، سنأخذ الأحرف المشفرة (التي تم تمريرها كمعامل arr) ونقسمها إلى تسلسلات متعددة، يتم تحديدها بواسطة Batch_size. سيكون طول كل تسلسل من تسلسلاتنا هو seq_length.

إنشاء دفعات

1. أول شيء يتعين علينا القيام به هو التخلص من بعض النص بحيث يكون لدينا فقط دفعات صغيرة كاملة تمامًا.
2. بعد ذلك، نحتاج إلى تقسيم arr إلى دفعات.
3. والآن بعد أن أصبح لدينا هذه المصفوفة، يمكننا تكرارها للحصول على دفعاتنا الصغيرة.

تدريب واختبار الدفعات

4. أنشئ دفعتين x,y حيث x هي دفعة الإدخال input batch وy هي دفعة التدريب training batch التي هي بالضبط x ولكن تم إزاحتها بحرف واحد.

```
def get_batches(arr, batch_size, seq_length):
```

```
    batch_size_total = batch_size * seq_length
    # total number of batches we can make
    n_batches = len(arr)//batch_size_total

    # Keep only enough characters to make full batches
    arr = arr[:n_batches * batch_size_total]
    # Reshape into batch_size rows
    arr = arr.reshape((batch_size, -1))

    # iterate through the array, one sequence at a time
    for n in range(0, arr.shape[1], seq_length):
        # The features
        x = arr[:, n:n+seq_length]
        # The targets, shifted by one
        y = np.zeros_like(x)
        try:
            y[:, :-1], y[:, -1] = x[:, 1:], arr[:, n+seq_length]
        except IndexError:
            y[:, :-1], y[:, -1] = x[:, 1:], arr[:, 0]
        yield x, y
```

رسم الإخراج

دعونا نحاول الحصول على دفعات من 100 حرف من البيانات المشفرة ونرى ما يحدث.

```
batches = get_batches(encoded, 8, 50)
x, y = next(batches)
# printing out the first 10 items in a sequence
print('x\n', x[:10, :10])
print('\ny\n', y[:10, :10])
```

```
x
 [[13 54 25 67 25 27 25 20  6 67]
 [46 16 67  0 20 67 74 20  6 25]
 [ 8 16 74 16 38 46 20 67 30 16]
 [42 36  2 67 20  6 47  2 12 38]
 [30 16 41 16 46  4 16 41 38 20]
 [41 42 31 37 67 51 16 46 72 20]
 [76 20 46  2 76 20 39 39 74 25]
 [38 76 28 42 25 76 67  6 20 67]]

y
 [[54 25 67 25 27 25 20  6 67 30]
 [16 67  0 20 67 74 20  6 25 37]
 [16 74 16 38 46 20 67 30 16 41]
 [36  2 67 20  6 47  2 12 38 46]
 [16 41 16 46  4 16 41 38 20 67]
 [42 31 37 67 51 16 46 72 20 19]
 [20 46  2 76 20 39 39 74 25 69]
 [76 28 42 25 76 67  6 20 67 25]]
```

وكما نرى أن y هو نفس x ولكن يتم إزاحته بحرف واحد فقط.

تعريف الشبكة باستخدام PyTorch

أدناه هو المكان الذي أقوم فيه بتعريف الشبكة.

بعد ذلك، سأستخدم PyTorch لتحديد بنية الشبكة. أبدأ بتحديد الطبقات والعمليات التي نريدها. ثم تحديد طريقة للتمرير إلى الأمام forward pass.

```
# check if GPU is available
train_on_gpu = torch.cuda.is_available()
if(train_on_gpu):
    print('Training on GPU!')
else:
    print('No GPU available, training on CPU; consider making
n_epochs very small.')
```

لا تتوفر وحدة معالجة الرسومات (GPU)، والتدريب على وحدة المعالجة المركزية (CPU)؛ فكرفي جعل عدد الفترات n_epochs صغيرًا جدًا.

```
class CharRNN(nn.Module):

    def __init__(self, tokens, n_hidden=256, n_layers=2,
                               drop_prob=0.5, lr=0.001):
        super().__init__()
        self.drop_prob = drop_prob
        self.n_layers = n_layers
        self.n_hidden = n_hidden
        self.lr = lr

        # creating character dictionaries
        self.chars = tokens
        self.int2char = dict(enumerate(self.chars))
        self.char2int = {ch: ii for ii, ch in
self.int2char.items()}

        ## define the LSTM
        self.lstm = nn.LSTM(len(self.chars), n_hidden, n_layers,
                            dropout=drop_prob, batch_first=True)

        ## define a dropout layer
        self.dropout = nn.Dropout(drop_prob)

        ## define the final, fully-connected output layer
        self.fc = nn.Linear(n_hidden, len(self.chars))


    def forward(self, x, hidden):
        ''' Forward pass through the network.
            These inputs are x, and the hidden/cell state `hidden`.
'''
```

```
        ## Get the outputs and the new hidden state from the lstm
        r_output, hidden = self.lstm(x, hidden)

        ## pass through a dropout layer
        out = self.dropout(r_output)

        # Stack up LSTM outputs using view
        # you may need to use contiguous to reshape the output
        out = out.contiguous().view(-1, self.n_hidden)

        ## put x through the fully-connected layer
        out = self.fc(out)

        # return the final output and the hidden state
        return out, hidden


    def init_hidden(self, batch_size):
        ''' Initializes hidden state '''
        # Create two new tensors with sizes n_layers x batch_size x
n_hidden,
        # initialized to zero, for hidden state and cell state of
LSTM
        weight = next(self.parameters()).data

        if (train_on_gpu):
            hidden = (weight.new(self.n_layers, batch_size,
self.n_hidden).zero_().cuda(),
                      weight.new(self.n_layers, batch_size,
self.n_hidden).zero_().cuda())
        else:
            hidden = (weight.new(self.n_layers, batch_size,
self.n_hidden).zero_(),
                      weight.new(self.n_layers, batch_size,
self.n_hidden).zero_())

        return hidden
```

```
def train(net, data, epochs=5, batch_size=10, seq_length=50,
lr=0.001, clip=5, val_frac=0.1, print_every=10):
    ''' Training a network

        Arguments
        ---------

        net: CharRNN network
        data: text data to train the network
        epochs: Number of epochs to train
        batch_size: Number of mini-sequences per mini-batch, aka
batch size
        seq_length: Number of character steps per mini-batch
        lr: learning rate
```

```
        clip: gradient clipping
        val_frac: Fraction of data to hold out for validation
        print_every: Number of steps for printing training and
validation loss

    '''
    # keep track of training and validation loss
    train_loss = 0.0
    valid_loss = 0.0
    valid_loss_min = np.Inf # track change in validation loss


    net.train()

    opt = torch.optim.Adam(net.parameters(), lr=lr)
    criterion = nn.CrossEntropyLoss()

    # create training and validation data
    val_idx = int(len(data)*(1-val_frac))
    data, val_data = data[:val_idx], data[val_idx:]

    if(train_on_gpu):
        net.cuda()

    counter = 0
    n_chars = len(net.chars)
    for e in range(epochs):
        # initialize hidden state
        h = net.init_hidden(batch_size)

        for x, y in get_batches(data, batch_size, seq_length):
            counter += 1

            # One-hot encode our data and make them Torch tensors
            x = one_hot_encode(x, n_chars)
            inputs, targets = torch.from_numpy(x),
torch.from_numpy(y)

            if(train_on_gpu):
                inputs, targets = inputs.cuda(), targets.cuda()

            # Creating new variables for the hidden state,
otherwise
            # we'd backprop through the entire training history
            h = tuple([each.data for each in h])

            # zero accumulated gradients
            net.zero_grad()

            # get the output from the model
            output, h = net(inputs, h)

            # calculate the loss and perform backprop
            loss = train_loss = criterion(output,
targets.view(batch_size*seq_length).long())
```

```
            loss.backward()
            # `clip_grad_norm` helps prevent the exploding gradient
problem in RNNs / LSTMs.
            nn.utils.clip_grad_norm_(net.parameters(), clip)
            opt.step()

            # loss stats
            if counter % print_every == 0:
                # Get validation loss
                val_h = net.init_hidden(batch_size)
                val_losses = []
                net.eval()
                for x, y in get_batches(val_data, batch_size,
seq_length):
                    # One-hot encode our data and make them Torch
tensors
                    x = one_hot_encode(x, n_chars)
                    x, y = torch.from_numpy(x), torch.from_numpy(y)

                    # Creating new variables for the hidden state,
otherwise
                    # we'd backprop through the entire training
history
                    val_h = tuple([each.data for each in val_h])

                    inputs, targets = x, y
                    if(train_on_gpu):
                        inputs, targets = inputs.cuda(),
targets.cuda()

                    output, val_h = net(inputs, val_h)
                    val_loss = valid_loss =  criterion(output,
targets.view(batch_size*seq_length).long())

                    val_losses.append(val_loss.item())

                net.train() # reset to train mode after iterationg
through validation data

                print("Epoch: {}/{}...".format(e+1, epochs),
                      "Step: {}...".format(counter),
                      "Loss: {:.4f}...".format(loss.item()),
                      "Val Loss:
{:.4f}".format(np.mean(val_losses)))

    # save model if validation loss has decreased
    if valid_loss <= valid_loss_min:
        print('Validation loss decreased ({:.6f} --> {:.6f}).
Saving model ...'.format(
        valid_loss_min,
        valid_loss))
        model_name = 'best_loss_so_far.net'

        checkpoint = {'n_hidden': net.n_hidden,
                      'n_layers': net.n_layers,
```

```
                        'state_dict': net.state_dict(),
                        'tokens': net.chars}

        with open(model_name, 'wb') as f:
            torch.save(model_name, f)
```

التدريب

يجب أن تحدد الخليتان التاليتان المعمارية وتدربان النموذج الذي حددناه أعلاه، وعدد طبقات LSTM n_layers متروك لك، لقد قمت بتدريبها باستخدام 5 طبقات على google colab مع 50 فترة epochs واستغرق الأمر 4 ساعات.

```
# define and print the net
n_hidden=512
n_layers=5

net = CharRNN(chars, n_hidden, n_layers)
print(net)
```

```
CharRNN(
  (lstm): LSTM(82, 512, num_layers=5, batch_first=True,
dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=512, out_features=82, bias=True)
)
```

```
batch_size = 128
seq_length = 100
n_epochs = 50 # start smaller if you are just testing initial
behavior

# train the model
train(net, encoded, epochs=n_epochs, batch_size=batch_size,
seq_length=seq_length, lr=0.001, print_every=10)
```

تحميل العينة

```
# Here we have loaded in a model that trained over 50 epochs
`rnn_50_epoch.net`
with open('rnn_50_epoch.net', 'rb') as f:
    if train_on_gpu:
            checkpoint = torch.load(f)
    else:
        checkpoint = torch.load(f,map_location=torch.device('cpu'))


loaded = CharRNN(checkpoint['tokens'],
n_hidden=checkpoint['n_hidden'], n_layers=checkpoint['n_layers'])
loaded.load_state_dict(checkpoint['state_dict'])
```

```
<All keys matched successfully>
```

تم أخذ الدالتين التاليتين من UDACITY – DeepLearning باستخدام PyTorch لأخذ عينات من الإدخال فقط وجعله بالتنسيق الصحيح ثم أخذ هذا الإدخال وإنشاء أحرف الإخراج بالحجم المحدد.

```
def predict(net, char, h=None, top_k=None):
        ''' Given a character, predict the next character.
            Returns the predicted character and the hidden state.
        '''

        # tensor inputs
        x = np.array([[net.char2int[char]]])
        x = one_hot_encode(x, len(net.chars))
        inputs = torch.from_numpy(x)

        if(train_on_gpu):
            inputs = inputs.cuda()

        # detach hidden state from history
        h = tuple([each.data for each in h])
        # get the output of the model
        out, h = net(inputs, h)

        # get the character probabilities
        p = F.softmax(out, dim=1).data
        if(train_on_gpu):
            p = p.cpu() # move to cpu

        # get top characters
        if top_k is None:
            top_ch = np.arange(len(net.chars))
        else:
            p, top_ch = p.topk(top_k)
            top_ch = top_ch.numpy().squeeze()

        # select the likely next character with some element of
randomness
        p = p.numpy().squeeze()
        char = np.random.choice(top_ch, p=p/p.sum())

        # return the encoded value of the predicted char and the
hidden state
        return net.int2char[char], h
```

```
def sample(net, size, prime='The', top_k=None):

    if(train_on_gpu):
        net.cuda()
    else:
        net.cpu()

    net.eval() # eval mode
```

```
    # First off, run through the prime characters
    chars = [ch for ch in prime]
    h = net.init_hidden(1)
    for ch in prime:
        char, h = predict(net, ch, h, top_k=top_k)

    chars.append(char)

    # Now pass in the previous character and get a new one
    for ii in range(size):
        char, h = predict(net, chars[-1], h, top_k=top_k)
        chars.append(char)

    return ''.join(chars)
```

```
print(sample(loaded, 300, prime='انا', top_k=5))
```

```
انا الماءِ ما أرى الدُنيا عَن أبـاكِ
وَكُنتُ مِنَ المَنازلِ واحِداتِ
وَإن أنصَرتِ في العَلياءِ فَتري
أَلَم يَبقَ المُلوكُ لِأُمَّةٍ وَعَبادِ
وَالنيلُ في حَولِ الحُسانِ وَكَيفَ يَن
مَولى النُفوسُ بِنارِ عَينِ المُلكِ
وَالمُلكُ ماءٌ يُرتَقى النُفُسَ المُعَدَّد
أَم لَها ما لا يَقيمُ النَجمُ فينا
```

المصدر:

https://github.com/AhmedAbdel-Aal/Arabic-poem-Generator/blob/master/Notebooks/Arabic%20Poem%20Generation.ipynb

7) توليد القصائد العربية بأسلوب نزار قباني باستخدام التعلم العميق Arabic Poems Generation in the Style of Nizar Qabbani using Deep Learning

إذا اختفى الشعر غداً، فلن تنهار سوق الأوراق المالية، وستبقى الجسورفي مكانها، وستظل أجهزة الكمبيوتر تعمل.

ومع ذلك، فإن للشعر قيمة فريدة لأنه يتحدث عن شيء بداخلنا لا يمكن قياسه.في هذه المقالة، سنحاول توليد الشعر باستخدام شبكة عصبية مع تحذير إضافي: سيكون باللغة العربية.

باختصار، يشمل هذا المنشور النقاط التالية:

- انشاء مجموعة بيانات مخصصة.
- معالجة البيانات مسبقا.
- ضبط المعلمة الفائقة لـ RNN.
- إخراج الشعر باللغة العربية (والترجمة الإنجليزية).

لا تترددفي تخطي الأجزاء الفنية والانتقال مباشرة إلى الإخراج. تم تضمين رابط لمستودع GitHubفي الأسفل.

شاعر من دمشق

كان نزار قباني شاعراً سورياً اشتهر بقصائده التي تناولت الحب والقومية والإثارة الجنسية والدين. بالإضافة إلى ذلك، كان كاتبًا غزير الإنتاج، مما يعني أن عمله يوفر كمية كبيرة من البيانات لشبكتنا العصبية للتعلم منها.

وفيما يلي عينة من عمله:

Who are you
woman entering my life like a dagger
mild as the eyes of a rabbit
soft as the skin of a plum
pure as strings of jasmine
innocent as children's bibs
and devouring like words?

كخطوة أولى، نحتاج إلى إنشاء مجموعة corpus نصية تحتوي على معظم أعماله المعروفة، إن لم يكن كلها. ولحسن الحظ، يمكننا العثور على مواقع إلكترونية مخصصة فقط للحفاظ على أعمال قباني.

باستخدام حزم مثل BeautifulSoup، يمكن للمرء استخراج scrape البيانات وإنشاء مجموعة تحتوي على جميع الأعمال المتاحة التي يمكننا العثور عليها. مع جمع كل القصائد، تكون كمية البيانات أقل بقليل من 1 ميجابايت، أي حوالي مليون حرف، وتحتوي على حوالي 32000 كلمة فريدة. لمزيد من المعلومات حول مقدار البيانات المطلوبة، راجع منشور Andrej Karpathy في المراجع أدناه.

على الرغم من ظهورها ككمية هائلة من النص، إلا أنها في الواقع تعتبر مجموعة بيانات صغيرة جداً، مما قد يمثل على الأرجح قيداً لأغراضنا.

خصوصية اللغة العربية

على عكس الحروف اللاتينية، تتم قراءة اللغة العربية من اليمين إلى اليسار. وبالإضافة إلى ذلك، لا يوجد شيء مثل الأحرف الكبيرة أو الصغيرة. علاوة على ذلك، فإن مفهوم حروف العلة والحروف الساكنة يختلف عن نطق اللغة الإنجليزية.

هناك المزيد من الجوانب حول كيفية اختلاف هذه اللغة وغيرها عن اللغة الإنجليزية. كانت هناك أمثلة ناجحة لتأليف قصائد بلغات أخرى غير الإنجليزية، مثل الصينية (انظر المراجع في الأسفل).

تحضير البيانات

تتضمن هذه الخطوة إنشاء جدول بحث يقوم بإرجاع قاموسين:

- عدد صحيح الى مفردات integer to vocab.
- المفردات إلى عدد صحيح integer to vocab.

بعد ذلك، قمنا بتقسيم النص إلى مصفوفة كلمات باستخدام المسافات كمحددات delimiters. ومع ذلك، يمكن لعلامات الترقيم مثل النقاط وعلامات التعجب إنشاء معرفات متعددة لنفس الكلمة. على سبيل المثال، "bye" و"bye!" من شأنه أن يولد معرفين مختلفين للكلمات.

نقوم بتنفيذ دالة لإرجاع القاموس الذي سيتم استخدامه لترميز tokenize رموز مثل "!" في "||Exclamation_Mark||"، تبدو قائمتنا كما يلي:

- فترة Period (.)
- فاصلة Comma (،)
- ارجاع Return (\n)
- ارجاع الحامل Carriage Return (\r)

سيتم استخدام هذا القاموس لترميز الرموز وإضافة الفاصل (المسافة) حوله. يؤدي هذا إلى فصل كل رمز على أنه كلمة خاصة به، مما يسهل على الشبكة العصبية التنبؤ بالكلمة التالية.

ضبط المعلمات الفائقة

بشكل عام، قد نحصل على نتائج أفضل بأبعاد مخفية أكبر وأبعاد n_layer، لكن النماذج الأكبر حجمًا تستغرق وقتًا أطول للتدريب. فيما يلي قائمة بالمعلمات التي سيتم ضبطها:

- sequence_length: طول التسلسل.
- Batch_size: حجم الدفعة.
- num_epochs: عدد الفترات التي سيتم التدريب عليها.
- Learning_rate: معدل التعلم لمحسن آدم.
- vocab_size: عدد التوكنزفي مفرداتنا.
- put_size: الحجم المطلوب للإخراج.
- embedding_dim: بُعد التضمين؛ أصغر من vocab_size.
- Hidden_dim: البعد المخفي لشبكة RNN الخاصة بنا.
- n_layers: عدد الطبقات/الخلايافي RNN الخاص بنا.

تقليد الشاعر

بعد تعيين المعلمات والنموذج المدرب، يمكننا الانتقال إلى الجزء الممتع: إنشاء القصائد! Generating poems!

للقيام بذلك يجب علينا تحديد prime_word وهي الكلمة التي يجب أن نبدأ بها.في الأقسام الفرعية أدناه نقدم بعض النتائج.

"أنا" (I/me):

```
أنا كهذا الماء ؟.
وظلت شيخ حارتنا..
وتشاهد أسماكٌ وبحار.
وتشاهد فهم أنفسنا
الخائفون من الخشب
ويضاجعون تطرف يعنيها.
تنقرض الأمة من الأصل ذكورا..
تهاجر أسماكٌ وبحار..
وعدت..
بإلغاء
```

ترجمة تقريبية:

Am I like this water?
And I stayed the Sheikh of our neighborhood ..
And you see fish and seas.
And you watch the understanding of ourselves

Afraid of wood
And they sleep with extremism.
The nation is extinct from male origin.
Fish and seas migrate ..
I promised ..
To cancel

(O) "يا"

يا قطتي الخليفة
وباركنا في جبيني
خليفةُ في خاصرة التعذيب الأوسط

هل سأقرأ حيِ ، ينسى القط؟
والزنبق الأسود ، والشذا ومستواه
فقصها..
وبقينا نسأل أنفسنا
وفي غرف الإنعاش ،
وألوان للعشاء
قطعاً في

ترجمة تقريبية:

O my successor cat
Bless us on my forehead
A caliph in the side of middle torture

Will I read alive, the cat forgets?
Black lily, scent and its level
So cut it ..
We kept asking ourselves
And in the recovery rooms,
And colors for dinner
Absolutely in

(We) "نحن"

نحن عشته
لا تحسبي أن أحبك في البيادر
في أخبار التاريخ,.
تنقرض الأمة يعنيها.
تنقرض الأمة من عارٍ فيها- الحداد..
عيونها على ذراعيها..
ومذيع الدولة في أجساد الأميره ؟

يا رب أين ملتفُ نسبي

ترجمة تقريبية:

We experienced it
Do not think that I love you in the Gardens
In history news ,.
The nation becomes extinct.
The nation becomes extinct from its disgrace — mourning ..
Her eyes are on her arms ..
And the state broadcaster in the princess's bodies?

O Lord, where is the relative winding?

(Woman) "امرأة"

```
امرأة كلها...
يا كل عام في الطبيعة..
ومذيع الدولة في جرحنا
نتفاءل جميله..
ووجدنا جسداً مغتصباً..
ومذيع الدولة ؟؟
من هؤلاء هؤلاء الهدبا
من هؤلاء سقيت أعماقي وإرهاقي برأس أدبي؟
```

ترجمة تقريبية:

A whole woman ...
Oh every year in nature ..
The state broadcaster is in our wound
Beautiful optimism ..
We found a raped body.
And the state broadcaster ??
Of these are cilia
Who are these people watered deep down and exhausted with a literary head?

الكود

مولد القصيدة

في هذا النوتبوك نعتمد على استخراج الويب لإنشاء شبكة RNN يمكنها توليد قصائد عربية بأسلوب نزار قباني.

```
import glob
import re
import numpy as np
from collections import Counter
import torch
from torch.utils.data import TensorDataset, DataLoader
import torch.nn as nn
import torch.nn.functional as F
import helper
```

```
#get list of text files in data
poem_txt_list = glob.glob('data/*.txt')
```

```
with open('raw_corpus.txt', 'w') as outfile:
    for fname in poem_txt_list:
        with open(fname) as infile:
            outfile.write(infile.read())
```

```
data_dir = 'raw_corpus.txt'
text = helper.load_data(data_dir)
```

```
view_line_range = (0, 10)

print('Dataset Stats')
print('Roughly the number of unique words: {}'.format(len({word:
None for word in text.split()})))

lines = text.split('\n')
print('Number of lines: {}'.format(len(lines)))
word_count_line = [len(line.split()) for line in lines]
print('Average number of words in each line:
{}'.format(np.average(word_count_line)))

print()
print('The lines {} to {}:'.format(*view_line_range))
print('\n'.join(text.split('\n')[view_line_range[0]:view_line_range
[1]]))
```

```
Dataset Stats
Roughly the number of unique words: 31980
Number of lines: 24304
Average number of words in each line: 3.9885615536537196

The lines 0 to 10:
 تحركي خطوةً.. يا نصف عاشقةٍ
 فلا أريد أنا أنصاف عشاقِ
 إن الزلازل طول الليل تضربني
 وأنت واضعةٌ ساقاً على ساق
 وأنت آخر من تعنيه مشكلتي
 ومن يشاركني حزني وإرهاقي
 تبللي مرةً بالماء .. أو بدمي
 وجربي الموت يوماً فوق أحداقي
 أنا غريبٌ.. ومنفيٌ.. ومستلب
 وثلج نهديك غطى كل أعماقي
```

المعالجة المسبقة

أول شيء يجب فعله لأي مجموعة بيانات هو المعالجة المسبقة pre-processing. نقوم بتنفيذ دوال المعالجة المسبقة التالية أدناه:

- جدول البحث Lookup Table.
- ترميز علامات الترقيم Tokenize Punctuation.

جدول البحث

لإنشاء تضمين كلمة word embedding، نحتاج أولاً إلى تحويل الكلمات إلى معرفات.في هذه الدالة ids، نقوم بإنشاء قاموسين:

- القاموس للانتقال من الكلمات إلى المعرف، سوف نسميه vocab_to_int
- القاموس للانتقال من المعرف إلى الكلمة، سوف نسميه int_to_vocab

نعيد هذه القواميس في الصف tuple التالي (vocab_to_int, int_to_vocab).

```
def create_lookup_tables(text):
    """
    Create lookup tables for vocabulary
    :param text: The text of tv scripts split into words
    :return: A tuple of dicts (vocab_to_int, int_to_vocab)
    """
    # TODO: Implement Function

    #create a counter for all words in text
    word_counts = Counter(text)

    #sort words from most to least frequent in the text
    sorted_vocab = sorted(word_counts, key=word_counts.get,
reverse=True)

    #create int to vocab dictionatires

    int_to_vocab = {ii: word for ii, word in
enumerate(sorted_vocab)}
    vocab_to_int = {word:ii for ii, word in int_to_vocab.items()}

    # return tuple
    return (vocab_to_int, int_to_vocab)
```

ترميز علامات الترقيم

سنقوم بتقسيم السكريبت إلى مصفوفة كلمات باستخدام المسافات كمحددات. ومع ذلك، يمكن لعلامات الترقيم مثل النقاط وعلامات التعجب إنشاء معرفات متعددة لنفس الكلمة. على سبيل المثال، "bye" و"bye!" من شأنه أن يولد معرفين مختلفين للكلمات.

نقوم بتنفيذ الدالة token_lookup لإرجاع الإملاء الذي سيتم استخدامه لترميز الرموز مثل "!"في "||علامة_التعجب||". نقوم بإنشاء قاموس للرموز التالية حيث الرمز هو المفتاح والقيمة هي التوكن:

- Period (.)
- Comma (,)

- Quotation Mark (**"**)
- Semicolon (**;**)
- Exclamation mark (**!**)
- Question mark (**?**)
- Left Parentheses (**(**)
- Right Parentheses (**)**)
- Dash (**-**)
- Return (**\n**)

سيتم استخدام هذا القاموس لترميز الرموز وإضافة الفاصل (المسافة) حوله. يؤدي هذا إلى فصل كل رمز على أنه كلمة خاصة به، مما يسهل على الشبكة العصبية التنبؤ بالكلمة التالية. تأكد من أننا لا نستخدم قيمة يمكن الخلط بينها وبين كلمة؛ على سبيل المثال، بدلاً من استخدام القيمة "dash"، جرّب استخدام قيمة مثل "||dash||".

لاحظ أننا قد نحتاج إلى النسخ من النص لبعض علامات الترقيم مثل علامة الاستفهام التي لها اتجاه معكوس في السكريبت العربي.

```
def token_lookup():
    """
    Generate a dict to turn punctuation into a token.
    :return: Tokenized dictionary where the key is the punctuation
and the value is the token
    """
    # TODO: Implement Function
    punct_dict = {'.': '||PERIOD||',
                  '*': '||COMMA||',
                  '\r': '||RECUR||',
                  '…': '||DOTDOTDOT||',
                  '!': '||EXCLAMATIONMARK||',
                  '؟ ': '||QUESTIONMARK||',
                  '(': '||LEFTPARANTH||',
                  ')': '||RIGHTPARANTH||',
                  '-': '||DASH||',
                  '\n': '||RETURN||'}

    return punct_dict
```

سيؤدي تشغيل خلية التعليمات البرمجية أدناه إلى معالجة جميع البيانات مسبقًا وحفظها في ملف. يرجى مراجعة الكود الخاص بـ preprocess_and_save_data في ملف helpers.py لمعرفة ما يفعله بالتفصيل.

```
helper.preprocess_and_save_data(data_dir, token_lookup,
create_lookup_tables)
int_text, vocab_to_int, int_to_vocab, token_dict =
helper.load_preprocess()
```

بناء الشبكة العصبية

في هذا القسم، نقوم ببناء المكونات اللازمة لبناء RNN من خلال تنفيذ وحدة RNN ودوال الانتشار الأمامي والخلفي.

تحقق من الوصول إلى GPU

```
# Check for a GPU
train_on_gpu = torch.cuda.is_available()
if not train_on_gpu:
    print('No GPU found. Please use a GPU to train your neural
network.')
```

```
No GPU found. Please use a GPU to train your neural network.
```

المدخلات

لنبدأ ببيانات الإدخال المعالجة مسبقًا. سنستخدم TensorDataset لتوفير تنسيق معروف لمجموعة البيانات الخاصة بنا؛ بالاشتراك مع DataLoader، فإنه سيتعامل مع دوال التجميع batching والخلط shuffling وغيرها من دوال تكرار مجموعة البيانات.

يمكننا إنشاء بيانات باستخدام TensorDataset عن طريق تمرير موترات الميزة والهدف. ثم قم بإنشاء DataLoader كالمعتاد.

التجميع

نقوم بتنفيذ دالة Batch_data لتجميع بيانات الكلماتفي أجزاء بحجم Batch_size باستخدام فئتي TensorDataset وDataLoader.

يمكننا تجميع الكلمات باستخدام DataLoader، ولكن سيكون الأمر متروكًا لنا لإنشاء feature_tensors وtarget_tensors بالحجم والمحتوى الصحيحين لطول تسلسل sequence_length معين.

على سبيل المثال، لنفترض أن لدينا هذه المدخلات:

```
words = [1, 2, 3, 4, 5, 6, 7]
sequence_length = 4
```

يجب أن يحتوي feature_tensor الأول على القيم:

```
[1, 2, 3, 4]
```

ويجب أن يكون target_tensor المقابل هو "الكلمة word"/قيمة الكلمة المميزة التالية tokenized word value فقط:

```
5
```

يجب أن يستمر هذا مع feature_tensor الثاني، حيث يكون target_tensor:

```
[2, 3, 4, 5]  # features
6             # target
```

```
def batch_data(words, sequence_length, batch_size):
    """
    Batch the neural network data using DataLoader
    :param words: The word ids of the TV scripts
    :param sequence_length: The sequence length of each batch
    :param batch_size: The size of each batch; the number of
sequences in a batch
    :return: DataLoader with batched data
    """

    #number of batches by integer definition
    n_batches = len(words)//batch_size

    #only full batches
    words = words[:n_batches*batch_size]
    y_len = len(words) - sequence_length

    x, y = [], []
    for idx in range(0, y_len):
        end = idx + sequence_length
        x_batch = words[idx:end]
        y_batch = words[end]

        x.append(x_batch)
        y.append(y_batch)

    #wrapping tensor
    data = TensorDataset(torch.from_numpy(np.asarray(x)),
torch.from_numpy(np.asarray(y)))
    # Combines a dataset and a sampler, and provides single- or
multi-process iterators over the dataset
    data_loader = DataLoader(data, batch_size=batch_size)

    # return a dataloader
    return data_loader
```

اختبار Dataloader

يتعين علينا تعديل هذا الكود لاختبار دالة التجميع batching، ولكن يجب أن تبدو متشابهة إلى حد ما.

أدناه، نقوم بإنشاء بعض بيانات نص الاختبار وتحديد أداة تحميل البيانات Dataloader باستخدام الدالة التي حددتها أعلاه. بعد ذلك، نحصل على بعض نماذج المدخلات Sample_x والأهداف Sample_y من أداة تحميل البيانات dataloader الخاصة بنا.

يجب أن يُرجع الكود الخاص بنا شيئًا مثل ما يلي (من المحتمل بترتيب مختلف، إذا قمنا بتعديل بياناتك):

```
torch.Size([10, 5])
tensor([[ 28,  29,  30,  31,  32],
        [ 21,  22,  23,  24,  25],
        [ 17,  18,  19,  20,  21],
        [ 34,  35,  36,  37,  38],
        [ 11,  12,  13,  14,  15],
        [ 23,  24,  25,  26,  27],
        [  6,   7,   8,   9,  10],
        [ 38,  39,  40,  41,  42],
        [ 25,  26,  27,  28,  29],
        [  7,   8,   9,  10,  11]])

torch.Size([10])
tensor([ 33,  26,  22,  39,  16,  28,  11,  43,  30,  12])
```

الأحجام

يجب أن يكون حجم Sample_x الخاص بنا (batch_size, serial_length) أو (5, 10)في هذه الحالة ويجب أن يكون لـ Sample_y بُعدًا واحدًا فقط: (batch_size (10.

القيم

يجب أن نلاحظ أيضًا أن الأهداف، Sample_y، هي القيمة التاليةفي بيانات test_text المطلوبة. لذلك، بالنسبة لتسلسل الإدخال [28، 29، 30، 31، 32] الذي ينتهي بالقيمة 32، يجب أن يكون الإخراج المقابل 33.

```
# test dataloader

test_text = range(50)
t_loader = batch_data(test_text, sequence_length=5, batch_size=10)

data_iter = iter(t_loader)
sample_x, sample_y = data_iter.next()

print(sample_x.shape)
print(sample_x)
print()
print(sample_y.shape)
print(sample_y)
```

```
torch.Size([10, 5])
tensor([[ 0,  1,  2,  3,  4],
        [ 1,  2,  3,  4,  5],
        [ 2,  3,  4,  5,  6],
        [ 3,  4,  5,  6,  7],
        [ 4,  5,  6,  7,  8],
        [ 5,  6,  7,  8,  9],
        [ 6,  7,  8,  9, 10],
        [ 7,  8,  9, 10, 11],
        [ 8,  9, 10, 11, 12],
        [ 9, 10, 11, 12, 13]])

torch.Size([10])
tensor([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

بناء الشبكة العصبية

نقوم بتنفيذ RNN باستخدام فئة الوحدة النمطية لـ PyTorch. قد نختار استخدام GRU أو LSTM. لإكمال RNN، علينا تنفيذ الدوال التالية للفئة:

- __init__ : دالة التهيئة.
- init_hidden: دالة التهيئة لحالة LSTM/GRU المخفية.
- forward: دالة الانتشار إلى الأمام.

يجب أن تقوم دالة التهيئة بإنشاء طبقات الشبكة العصبية وحفظهافي الفئة. ستستخدم دالة الانتشار الأمامي هذه الطبقات لتشغيل الانتشار الأمامي وإنشاء مخرجات وحالة مخفية.

يجب أن يكون مخرج هذا النموذج هو الدفعة الأخيرة من درجات الكلمات word scores بعد معالجة التسلسل الكامل. وهذا يعني أنه بالنسبة لكل تسلسل إدخال من الكلمات، نريد فقط إخراج درجات الكلمات لكلمة واحدة تالية على الأرجح.

ملاحظات

1. تأكد من تكديس مخرجات lstm لتمريرها إلى الطبقة المتصلة بالكامل، يمكننا القيام بذلك باستخدام lstm_output = lstm_output.contigious().view(-1, self.hidden_dim)
2. يمكننا الحصول على الدفعة الأخيرة من درجات الكلمات من خلال تشكيل shaping مخرجات الطبقة النهائية المتصلة بالكامل كما يلي:

```
# reshape into (batch_size, seq_length, output_size)
output = output.view(batch_size, -1, self.output_size)
# get last batch
out = output[:, -1]
```

```
class RNN(nn.Module):

    def __init__(self, vocab_size, output_size, embedding_dim,
hidden_dim, n_layers, dropout=0.5, lr=0.001):
        """
        Initialize the PyTorch RNN Module
        :param vocab_size: The number of input dimensions of the
neural network (the size of the vocabulary)
        :param output_size: The number of output dimensions of the
neural network
        :param embedding_dim: The size of embeddings, should you
choose to use them
        :param hidden_dim: The size of the hidden layer outputs
        :param dropout: dropout to add in between LSTM/GRU layers
```

```
        """
        super(RNN, self).__init__()
        #Implement function

        #set class variables
        self.n_layers = n_layers
        self.hidden_dim = hidden_dim
        self.output_size = output_size

        # embedding layer
        self.embedding = nn.Embedding(num_embeddings = vocab_size,
embedding_dim = embedding_dim)

        # define lstm
        self.lstm = nn.LSTM(input_size = embedding_dim,
                            hidden_size = hidden_dim,
                            num_layers = n_layers,
                            bias = True,
                            batch_first = True,
                            dropout = dropout)

        #define fc layer
        self.fc = nn.Linear(hidden_dim, output_size)


    def forward(self, nn_input, hidden):
        """
        Forward propagation of the neural network
        :param nn_input: The input to the neural network
        :param hidden: The hidden state
        :return: Two Tensors, the output of the neural network and
the latest hidden state
        """

        #get the batch size
        batch_size = nn_input.size(0)

        #get embedding
        embed = self.embedding(nn_input)

        #get lstm output
        out, hidden = self.lstm(embed, hidden)

        #stack the outputs of the lstm
        out = out.contiguous().view(-1, self.hidden_dim)

        out = self.fc(out)


        # reshape into (batch_size, seq_length, output_size)
        out = out.view(batch_size, -1, self.output_size)
        # get last batch
        out = out[:, -1]
```

```
        # return one batch of output word scores and the hidden
state
        return out, hidden


    def init_hidden(self, batch_size):
        '''
        Initialize the hidden state of an LSTM/GRU
        :param batch_size: The batch_size of the hidden state
        :return: hidden state of dims (n_layers, batch_size,
hidden_dim)
        '''
        # Implement function

        # initialize hidden state with zero weights, and move to
GPU if available
        weight = next(self.parameters()).data

        if (train_on_gpu):
            hidden = (weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().cuda(),
                      weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().cuda())
        else:
            hidden = (weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_(),
                      weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_())

        return hidden
```

تعريف الانتشار الأمامي والخلفي

استخدم فئة RNN التي قمنا بتنفيذها لتطبيق الانتشار الأمامي والخلفي. سيتم استدعاء هذه الدالة بشكل متكررفي حلقة التدريب كما يلي:

```
loss = forward_back_prop(decoder, decoder_optimizer, criterion,
inp, target)
```

ويجب أن يُرجع متوسط الخسارة (الخطأ) average loss على الدُفعة والحالة المخفية التي يتم إرجاعها عن طريق استدعاء RNN(inp, Hidden). تذكر أنه يمكننا الحصول على هذه الخطأ عن طريق حسابها، كالعادة، واستدعاء Loss.item() .

إذا كانت وحدة معالجة الرسومات (GPU) متاحة، فانقل بياناتك إلى جهاز وحدة معالجة الرسومات (GPU) هنا.

```
def forward_back_prop(rnn, optimizer, criterion, inp, target,
hidden, clip=5):
    """
    Forward and backward propagation on the neural network
```

```
    :param decoder: The PyTorch Module that holds the neural
network
    :param decoder_optimizer: The PyTorch optimizer for the neural
network
    :param criterion: The PyTorch loss function
    :param inp: A batch of input to the neural network
    :param target: The target output for the batch of input
    :param clip: Max norm of the gradients
    :return: The loss and the latest hidden state Tensor
    """

    # move data to GPU, if available
    if train_on_gpu:
        rnn.cuda()

    # perform backpropagation and optimization


    # Creating new variables for the hidden state, otherwise
    # we'd backprop through the entire training history
    h = tuple([each.data for each in hidden])

    # zero accumulated gradients
    rnn.zero_grad()

    # get the output from the model
    if train_on_gpu:
        inp = inp.cuda()
        target = target.cuda()


    output, h = rnn(inp, h)

    # calculate the loss and perform backprop
    loss = criterion(output, target)
    loss.backward()

    # `clip_grad_norm` helps prevent the exploding gradient problem
in RNNs / LSTMs.
    nn.utils.clip_grad_norm_(rnn.parameters(), clip)
    optimizer.step()


    # return the loss over a batch and the hidden state produced by
our model
    return loss.item(), h
```

تدريب الشبكات العصبية

مع اكتمال معمارية الشبكة وجاهزية البيانات لتغذيتهافي الشبكة العصبية، فقد حان الوقت لتدريبها.

حلقة التدريب

يتم تنفيذ حلقة التدريبفي دالة Train_decoder. ستقوم هذه الدالة بتدريب الشبكة على جميع الدُفعات لعدد الفترات المحددة. سيتم عرض تقدم النموذجفي كل عدد من الدُفعات. يتم تعيين هذا

الرقم باستخدام المعلمة show_every_n_batches. قمنا بتعيين هذه المعلمة مع المعلمات الأخرى في القسم التالي.

```
def train_rnn(rnn, batch_size, optimizer, criterion, n_epochs,
show_every_n_batches=100):
    batch_losses = []

    rnn.train()

    print("Training for %d epoch(s)..." % n_epochs)
    for epoch_i in range(1, n_epochs + 1):

        # initialize hidden state
        hidden = rnn.init_hidden(batch_size)

        for batch_i, (inputs, labels) in enumerate(train_loader,
1):

            # make sure you iterate over completely full batches,
only
            n_batches = len(train_loader.dataset)//batch_size
            if(batch_i > n_batches):
                break

            # forward, back prop
            loss, hidden = forward_back_prop(rnn, optimizer,
criterion, inputs, labels, hidden)
            # record loss
            batch_losses.append(loss)

            # printing loss stats
            if batch_i % show_every_n_batches == 0:
                print('Epoch: {:>4}/{:<4}  Loss: {}\n'.format(
                    epoch_i, n_epochs, np.average(batch_losses)))
                batch_losses = []

    # returns a trained rnn
    return rnn
```

المعلمات الفائقة

نقوم بإعداد وتدريب الشبكة العصبية باستخدام المعلمات التالية:

- اضبط sequence_length على طول التسلسل.
- اضبط Batch_size على حجم الدفعة.
- اضبط num_epochs على عدد الفترات التي سيتم التدريب عليها.
- اضبط learning_rate على معدل التعلم لمُحسِّن Adam.
- اضبط vocab_size على عدد التوكن الفريدةفي مفرداتنا.
- اضبط output_size على الحجم المطلوب للإخراج.

- اضبط embedding_dim على بُعد التضمين؛ أصغر من vocab_size.
- اضبط Hidden_dim على البعد المخفي لـ RNN الخاص بنا.
- اضبط n_layers على عدد الطبقات/الخلايافي RNN الخاص بنا.
- اضبط show_every_n_batches على عدد الدُفعات التي يجب أن تطبع فيها الشبكة العصبية التقدم.

إذا لم تحصل الشبكة على النتائج المرغوبة، فقم بتعديل هذه المعلمات و/أو الطبقات الموجودةفي فئة RNN.

```
# Data params
# Sequence Length = # of words in a sequence
sequence_length =  10
# Batch Size
batch_size = 128

# data loader - do not change
train_loader = batch_data(int_text, sequence_length, batch_size)
```

```
# Training parameters
# Number of Epochs
num_epochs = 15
# Learning Rate
learning_rate = 0.001

# Model parameters
# Vocab size
vocab_size = len(vocab_to_int)
# Output size
output_size = vocab_size
# Embedding Dimension
embedding_dim = 250
# Hidden Dimension
hidden_dim = 512
# Number of RNN Layers
n_layers = 2

# Show stats for every n number of batches
show_every_n_batches = 500
```

التدريب

في الخلية التالية، نقوم بتدريب الشبكة العصبية على البيانات المعالجة مسبقًا. إذا واجهنا صعوبةفي الحصول على خسارة جيدة، فقد نفكرفي تغيير المعلمات الفائقة. بشكل عام، قد نحصل على نتائج أفضل بأبعاد مخفية وأبعاد n_layer أكبر، لكن النماذج الأكبر حجمًا تستغرق وقتًا أطول للتدريب.

ملاحظة: الهدف لخسارة أقل من 3.5.

يمكننا أيضًا تجربة أطوال تسلسلية مختلفة، والتي تحدد حجم التبعيات طويلة المدى التي يمكن للنموذج تعلمها.

```
# create model and move to gpu if available
rnn = RNN(vocab_size, output_size, embedding_dim, hidden_dim,
n_layers, dropout=0.5)
if train_on_gpu:
    rnn.cuda()

# defining loss and optimization functions for training
optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

# training the model
trained_rnn = train_rnn(rnn, batch_size, optimizer, criterion,
num_epochs, show_every_n_batches)

# saving the trained model
helper.save_model('./save/trained_rnn', trained_rnn)
print('Model Trained and Saved')
```

```
Training for 15 epoch(s)...
Epoch:    1/15    Loss: 6.76675343799591

Epoch:    1/15    Loss: 6.370319797515869

Epoch:    2/15    Loss: 5.788004631557256

Epoch:    2/15    Loss: 5.498991756439209

Epoch:    3/15    Loss: 5.315837315389779

Epoch:    3/15    Loss: 5.086999155521393

Epoch:    4/15    Loss: 4.981034511894964

Epoch:    4/15    Loss: 4.744471074104309

Epoch:    5/15    Loss: 4.643881075653755

Epoch:    5/15    Loss: 4.411609432220459

Epoch:    6/15    Loss: 4.350242452576826

Epoch:    6/15    Loss: 4.088176232337951

Epoch:    7/15    Loss: 4.028933911911225

Epoch:    7/15    Loss: 3.7675121421813964

Epoch:    8/15    Loss: 3.6930512214971594
```

```
Epoch:    8/15    Loss: 3.4714884057044983
Epoch:    9/15    Loss: 3.3951571801802296
Epoch:    9/15    Loss: 3.1781059091091155
Epoch:   10/15    Loss: 3.096065091268507
Epoch:   10/15    Loss: 2.916186452627182
Epoch:   11/15    Loss: 2.8425318199600706
Epoch:   11/15    Loss: 2.6659098613262175
Epoch:   12/15    Loss: 2.577285630212931
Epoch:   12/15    Loss: 2.4270215339660646
Epoch:   13/15    Loss: 2.357348472344894
Epoch:   13/15    Loss: 2.2437951612472533
Epoch:   14/15    Loss: 2.166019320069535
Epoch:   14/15    Loss: 2.0584885563850404
Epoch:   15/15    Loss: 2.0044423047912288
Epoch:   15/15    Loss: 1.8923821833133698
Model Trained and Saved
```

نقطة الحفظ

بعد تشغيل خلية التدريب المذكورة أعلاه، سيتم حفظ نموذجنا بالاسم، Trainer_rnn، وإذا قمنا بحفظ تقدم النوتبوك الخاص بنا، فيمكننا التوقف هنا والعودة إلى هذا الكودفي وقت آخر. يمكننا استئناف تقدمنا عن طريق تشغيل الخلية التالية، والتي سيتم تحميلهافي قواميس Word:id الخاصة بنا وتحميلهافي النموذج المحفوظ لدينا بالاسم.

```
_, vocab_to_int, int_to_vocab, token_dict =
helper.load_preprocess()
trained_rnn = helper.load_model('trained_rnn')
```

توليد القصائد

بعد تدريب الشبكة وحفظها، يمكننا استخدامها لتوليد قصيدة قباني جديدةفي هذا القسم.

إنشاء نص

لإنشاء النص، تحتاج الشبكة إلى البدء بكلمة واحدة وتكرار توقعاتها حتى تصل إلى طول محدد. نستخدم دالة generate للقيام بذلك. يستغرق الأمر معرف كلمة للبدء به، وهو prime_id، ويولد

طولًا محددًا للنص، predict_len. لاحظ أيضًا أنه يستخدم أخذ عينات من Topk لإدخال بعض العشوائيةفي اختيار الكلمة التالية الأكثر احتمالاً،في ضوء مجموعة مخرجات من درجات الكلمات!

كيفية كتابة الحروف العربية

قد يكون لدى بعض المستخدمين لوحة مفاتيح عربية مدمجة على أجهزة الكمبيوتر الخاصة بهم. ومع ذلك، قد لا يكون هذا هو الحال، لذا لا تترددفي استخدام لوحات المفاتيح الذكية مثل yamli.com.

```
def generate(rnn, prime_id, int_to_vocab, token_dict, pad_value,
predict_len=100):
    """
    Generate text using the neural network
    :param decoder: The PyTorch Module that holds the trained
neural network
    :param prime_id: The word id to start the first prediction
    :param int_to_vocab: Dict of word id keys to word values
    :param token_dict: Dict of puncuation tokens keys to puncuation
values
    :param pad_value: The value used to pad a sequence
    :param predict_len: The length of text to generate
    :return: The generated text
    """
    rnn.eval()

    # create a sequence (batch_size=1) with the prime_id
    current_seq = np.full((1, sequence_length), pad_value)
    current_seq[-1][-1] = prime_id
    predicted = [int_to_vocab[prime_id]]

    for _ in range(predict_len):
        if train_on_gpu:
            current_seq = torch.LongTensor(current_seq).cuda()
        else:
            current_seq = torch.LongTensor(current_seq)

        # initialize the hidden state
        hidden = rnn.init_hidden(current_seq.size(0))

        # get the output of the rnn
        output, _ = rnn(current_seq, hidden)

        # get the next word probabilities
        p = F.softmax(output, dim=1).data
        if(train_on_gpu):
            p = p.cpu() # move to cpu

        # use top_k sampling to get the index of the next word
        top_k = 5
        p, top_i = p.topk(top_k)
        top_i = top_i.numpy().squeeze()

        # select the likely next word index with some element of
randomness
```

```
        p = p.numpy().squeeze()
        word_i = np.random.choice(top_i, p=p/p.sum())

        # retrieve that word from the dictionary
        word = int_to_vocab[word_i]
        predicted.append(word)

        # the generated word becomes the next "current sequence"
and the cycle can continue
        current_seq = np.roll(current_seq, -1, 1)
        current_seq[-1][-1] = word_i

    gen_sentences = ' '.join(predicted)

    # Replace punctuation tokens
    for key, token in token_dict.items():
        ending = ' ' if key in ['\n', '(', '"'] else ''
        gen_sentences = gen_sentences.replace(' ' + token.lower(),
key)
    gen_sentences = gen_sentences.replace('\n ', '\n')
    gen_sentences = gen_sentences.replace('( ', '(')

    # return all the sentences
    return gen_sentences
```

توليد قصيدة

حان الوقت لإنشاء النص. قمنا بتعيين gen_length على طول القصيدة التي نريد إنشاءها وقمنا بتعيين prime_word على أحد العناصر التالية لبدء التنبؤ:

- "أنا" (I/me)
- "يا" (O)
- "نحن" (We)
- "امرأة" (Woman)

يمكننا تعيين الكلمة الأساسية لأي كلمةفي قاموسنا، ولكن من الأفضل أن نبدأ بكلمة تبدأ عادة جملة باللغة العربية، أو حتى أفضل:

اقرأ القصائد وحاول أن تبدأ مثل المؤلف الأصلي:

```
# run the cell multiple times to get different results!
gen_length = 50 # modify the length to your preference
# name for starting the script
prime_word =  'أنـا'


pad_word = helper.SPECIAL_WORDS['PADDING']
generated_script = generate(trained_rnn,
                            vocab_to_int[prime_word],
                            int_to_vocab,
                            token_dict,
```

```
                                vocab_to_int[pad_word],
                                gen_length)
print(generated_script)
```

```
؟ أنا كهذا الماء.
وظلت شيخ حارتنا..
وتشاهد أسماكٌ وبحار.
وتشاهد فهم أنفسنا
الخائفون من الخشب
ويضاجعون تطرف يعنيها.
تنقرض الأمة من الأصل ذكورا..
تهاجر أسماكٌ وبحار..
12
وعدت..
بإلغاء
```

```
prime_list = ["أنا",  "يا",  "نحن", "امرأة"]

for word in prime_list:
    print("Generating poem for {}".format(word))
    generated_script = generate(trained_rnn,
                                vocab_to_int[word],
                                int_to_vocab,
                                token_dict,
                                vocab_to_int[pad_word],
                                gen_length)
    print(generated_script)
    print(20*'-')
```

```
Generating poem for أنا
أنا كهذا عصرنا- الحداد
من هؤلاء الطارئون والصندل

يا شام، حي.. ومنفيٌ..
وبنثرهم نبيذٍ.. أو تهربي..
وبدأنا أشجار حارتنا
قطعاً أنت.. وبين الفار..
ووجدنا أشباه مغتصباً..
وتشاهد
--------------------
Generating poem for يا
يا قطتي الخليفة والقصور..
ونحن, في غباء دفتري..
وهم أطلع من أرنبتي نعال وبين ورود.
وبدأنا ننسى أنفسنا جاءها معاويةٍ

لكنه غضبٌ طالت أظافره
ماذا سيحدث يتغزلون؟
كيف أحوالك ، مفرد ، والشذا خطاك؟

--------------------
```

```
Generating poem for نحن
..نحن رحيل الأميره
.وهم أطلع في الأصل إناثاً
وبدأنا ننسى متجولون ؟؟
4
البائعون ثقافةً مغشوشةً
..والراقدون بغرفة الإنعاش
..عيونها
..ضفيرةً
..وقصة رصاصهم
ماذا سيحدث للكواكب؟ وأنت خلتها كذبا
--------------------
Generating poem for امرأة
...امرأة كلها
9
..يا كل عام في الطبيعة
ومذيع الدولة في جرحنا
..نتفاءل جميله
..ووجدنا جسداً مغتصباً
ومذيع الدولة ؟؟
من هؤلاء هؤلاء الهدبا
من هؤلاء سقيت أعماقي وإرهاقي برأس أدبي؟
```

الاستنتاج

يمكننا أن نرى أن محاولاتنا للشعر ليست متماسكة وبالتأكيد ليست ببلاغة المؤلف الأصلي.في بعض الأحيان كانت الكتابة هزلية وكسرت كل قواعد النحو والمنطق.

قد يكون أحد الأسباب المحتملة لأوجه القصور لدينا هو عدم كفاية بيانات التدريب، حيث أننا نريد نصًا بقيمة 3 ميغابايت على الأقل. بالإضافة إلى ذلك، قد تكون هناك جوانب مميزة للغة نفسها تحتاج إلى أخذهافي الاعتبار. ومع ذلك، ضعفي اعتبارك أن RNN كان عليها أن تتعلم واحدة من أصعب اللغات من الصفر.

أتمنى أن تكون قد استمتعت بقراءة هذا المقال وتعرفت على ما هو ممكن فيما يتعلق بإنشاء النص. وأتمنى أيضًا أن يتمكن أعضاء مجتمع التعلم العميق الذين ليسوا متحدثين أصليين للغة الإنجليزية من تصور التطبيقات المفيدة المحتملةفي مجتمعاتهم الأصلية.

المصدر:

https://github.com/NadimKawwa/PoeticNeuralNetworks/blob/master/generate_poem.ipynb

https://towardsdatascience.com/poetic-neural-networks-487616512

8) نمذجة موضوعات من القرآن الكريم باستخدام التعلم الآلي والمعالجة اللغوية الطبيعية Modeling Topics from The Nobel Quran using Machine Learning & NLP

هل سبق لك أن أردت معرفة المزيد عن المعالجة اللغوية الطبيعية Natural Language Processing (NLP)، أو كنت مهتمًا بمعرفة كيفية تدريب خوارزمية نمذجة المواضيع Topic Modeling على مجموعة من الفقرات للتنبؤ بالموضوعات من الكلمات والعكس؟ إذا كانت إجابتك بنعم، فنأمل أن تساعدك هذه المقالةفي رحلة تعلم المعالجة اللغوية الطبيعية.

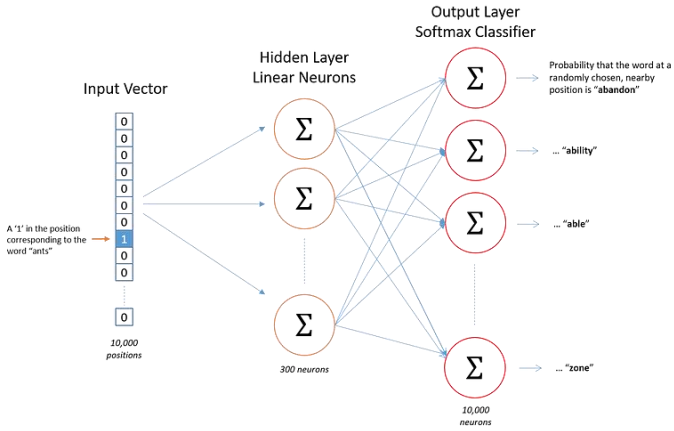
سأقومفي هذه المقالة بتوظيف إحدى الخوارزميات المعاصرة التي نشرها باحثو Google (ميكولوف، توماس، وآخرون). "التمثيلات الموزعة للكلمات والعبارات وتركيبها Distributed representations of words and phrases and their compositionality" (2013). يقدم البحث نموذج تضمين الكلمات باستخدام شبكة عصبية ضحلة shallow Neural Network ذات طبقة مخفية واحدة يمكن تدريبها على إعادة بناء السياق اللغوي للكلمات.

لوضع هذا التمرينفي سياقه، سنقوم ببناء خوارزمية تعلم غير خاضعة للأشراف unsupervised learning algorithm لنمذجة موضوعات من آيات متعددة من القرآن الكريم. ولتحقيق ذلك، سأستخدم word2vec من مكتبة Gensim لتكوين نموذج وتدريبه وبناءه عن طريق مسح القرآن الكريم بالكامل من البداية إلى النهاية، آية بآية، وكلمة بكلمة.

تستخدم الخوارزمية نافذة متحركة moving window مكونة من 7 كلمات لمسح الآيات وتحديد التواجد المشترك للكلمات بناءً على تكرار ظهورها كجيران، ولتحقيق ذلك، استخدمت معلمات فائقة hyperparameters مختلفة لنافذة تضمين الكلمات word embedding ومعدل التعلم learning rate.

دعونا نتعمقفي الكود. سأشرح العملية بشكل عكسي هذه المرة لمشاركة النتائج أولاً ولجعل متابعتها أكثر إثارة للاهتمام. لذا، لنبدأ بعرض بعض نتائج الخوارزمية؛ وبعد ذلك، سنستمرفي معرفة كيفية تدريب النموذج ونشره.

تم الحصول على النتائج التالية من تشغيل الخوارزمية للعثور على سياق يعتمد على تمرير كلمة واحدة كمدخل، وهي تقنية تسمى أيضًا طريقة تخطي جرام skip-gram.



في الأمثلة التالية، لكل كلمة إدخال، سنقوم بطباعة سحابة كلمات wordcloud تحتوي على أفضل 80 كلمة حدثتفي سياق مشابه.

مثال 1:تمرير `موسى``moses` للنموذج المتدرب

عندما تقرأ الخوارزمية "Moses" ككلمة مدخلة لنموذجنا، اكتسبت الخوارزمية مجموعة من الكلمات المجاورة (80) التي تم ذكرها بشكل متكررفي سياق مماثل. على سبيل المثال لا الحصر (Israel, Noah, Merry, Lot, John, Believers, Righteous, Tribes, etc). مرة أخرى، تحتوي سحابة الكلمات الموضحة أدناه على جميع الكلماتفي القرآن ذات الاحتمالية الأعلى للظهور حول كلمة "Moses".

تمرير `Moses` ككلمة إدخال

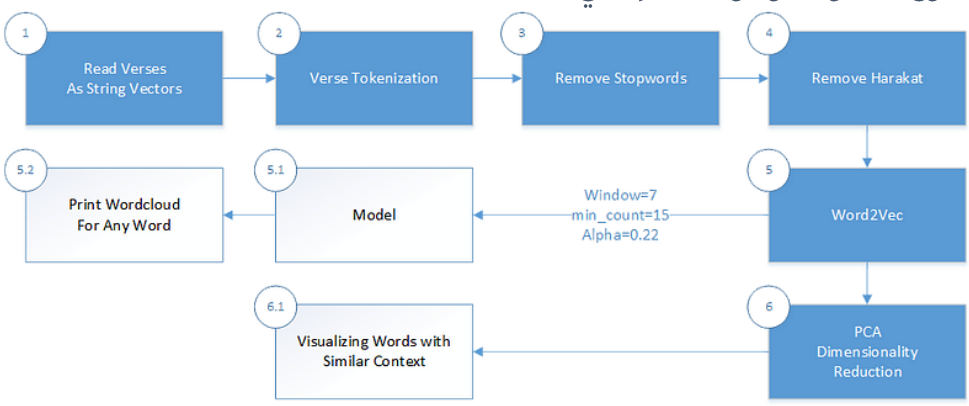
مثال 2: تمرير `جنة` `heaven` للنموذج المدرب

تمرير "heaven" ككلمة

المتطلبات الأساسية لبناء المشروع

- تحتاج إلى تثبيت بايثون.
- تحميل النسخة الرقمية csv للقرآن الكريم Holy Quran.
- تثبيت مكتبة (gensim) لتتمكن من استخدام خوارزمية word2vec.
- تثبيت مجموعة أدوات المعالجة اللغوية الطبيعية للغة بايثون (nltk) لتنزيل كلمات التوقف stop-words للغة العربية، واستخدم مصدر الكلمة العربية الخاص بها.
- تثبيت مكتبات bi-directional وarabic_reshaper لدعم تصور النص العربي.
- تثبيت (pandas) لتحميل البيانات وتحويلها باستخدام مكتبات Dataframe.
- تثبيت مكتبة (scikit) لتتمكن من تصور التضمين embedding باستخدام خوارزمية PCA.
- تثبيت الاضافة (wordcloud) لتصور أوجه التشابه بين الكلماتفي شكل سحابي.
- كل ما عليك فعله هو التحقق من مشروع بايثون مفتوح المصدر وتثبيت التبعيات من ملف require.txt

شرح مسار الكود والتعلم الآلي



1. قراءة الآيات من نسخة رقمية من القرآن الكريم كإطار بيانات.

```
# Download Arabic stop words Dataset from NLTK library
nltk.download('stopwords')# Extract Arabic stop words
arb_stopwords = set(nltk.corpus.stopwords.words("arabic"))#
Initialize Arabic stemmer
st = ISRIStemmer()# Load Quran from csv into a dataframe
df = pd.read_csv('data/arabic-original.csv', sep='|',
header='infer');
```

2. ترميز الآية

قم بتقسيم الكلمات من كل آية وقم بتعيين كل آية إلى متجه vector مع مجموعة من عناصر الكلمة المقابلة.

```
# Tokinize words from verses and vectorize them
df['verse'] = df['verse'].str.split()
```

3. إزالة كلمات التوقف لتحديد الكلمات ذات المعنى المهم فقط

```
# Remove Arabic stop words
df['verse'] = df['verse'].map(lambda x: [w for w in x if w not in
arb_stopwords])
```

4. إزالة الحركات (الواصلة)

ستساعدنا إزالة الحركات Harakatفي تقليل مجموعة الكلمات التي تنتمي إلى نفس الفعل.

```
# Remove harakat from the verses to simplify the corpus
df['verse'] = df['verse'].map(lambda x: re.sub('[ٰۭۜ۟۠ۡ]', '',
x))
```

5. بدأ التدريب وبناء نموذج Word2vec

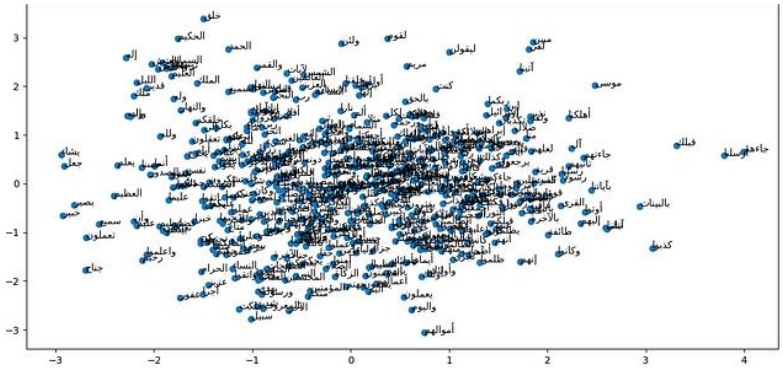
عرض متغير الحالة الأخيرة للآيات بعد تصفية كلماتنا وقبل أن نمررها إلى النموذج

في الشكل أعلاه، يمكنك رؤية الحالة الأخيرة للآيات المتغيرةفي القائمة بعد مرور المتجهات إلى مسار الترشيح filtration pipeline (ترميز الآية Verse Tokinization، وإزالة كلمات الإيقاف Remove Stop Words، وإزالة الحركات Remove Harakaat)، لذا فإن قائمة الآيات الآنفي شكلها المختصر الأخير، وجاهزة للحذف يتم تمريرها إلى نموذجنا للتدريب.

لا ترددفي تجربة المعلمات الفائقة hyperparameters، لأنها قد تؤثر على النموذج سلبًا أو إيجابًا. يتم استخدام أفضل المعلمات التي وجدتها شخصيًافي مقتطف الكود التالي. لفهم المعلمات أكثر يُرجى قراءة (التوثيق). لاحظ أنني قمت بتمرير min_count=15، الذي يملي على النموذج تجاهل جميع الكلمات التي يقل عدد تكرارها عن 15. لا ترددفي تغييره لزيادة أو تقليل حجم المفردات التي تم إنشاؤها.

```
# You can filter for one surah too if you want!
verses = df['verse'].values.tolist()# train the model
model = Word2Vec(verses, min_count=15, window=7, workers=8,
alpha=0.22)
```

مكتبة تصور النماذج المخصصة



تصور متجهات الكلمات بعد تطبيق PCA

يجب أن يحتوي النموذج الذي تم إنشاؤه على تمثيل متجه لجميع الكلمات التي نقوم بإدخالها. لكي نتمكن من تصور تمثيل المتجهات ومعرفة أي الكلمات أقرب من حيث التواجد والسياق إلى الكلمات الأخرى، نحتاج إلى تقليل الأبعاد التي لدينا من مصفوفة بالحجم (549x100) model[model.wv.vocab] . يتطلب تحقيق ذلك خوارزمية تقليل الأبعاد مثل PCA، ويوضح لك المقتطف التالي من الكود بالضبط كيف استخدمت PCA لتقليل تمثيل المتجه بحجم (549x100) إلى (549x2) حتى أتمكن من تصور المفردات بأكملهافي الرسم البياني ثنائي الابعاد الموضح أعلاه.

```
# fit a 2d PCA model to the vectors
X = model[model.wv.vocab]
pca = PCA(n_components=2)
result = pca.fit_transform(X)# create a scatter plot of the
projection
plt.scatter(result[:, 0], result[:, 1])
words = list(model.wv.vocab)# Pass list of words as an argument
for i, word in enumerate(words):
    reshaped_text = arabic_reshaper.reshape(word)
    artext = get_display(reshaped_text)
    plt.annotate(artext, xy=(result[i, 0], result[i, 1]))
plt.show()
```

الكود في Github

https://github.com/aelbuni/quran-nlp

المصدر:

https://aelbuni.medium.com/modeling-topics-in-quran-using-machine-learning-nlp-b88ca23fb44d

9) تصنيف المواضيع العربية في مجموعة بيانات أخبار هسبريس Arabic Topic Classification On The Hespress News Dataset

وفقًا لموقع "alexa.com"، يحتل موقع هسبريس Hespress المرتبة الرابعةفي المغرب، وهو أكبر موقع إخباريفي البلاد، ويقضي المواطن المغربي العادي حوالي 6 دقائق يوميًا على الموقع.

مجموعة بيانات Hespress عبارة عن مجموعة من 11 ألف مقالة إخبارية مصنفة حسب الموضوع و300 ألف تعليق مع نقاط من قبل المستخدمين المرتبطين بكل واحد منهم، فكرفي النتائج على أنها إعجابات بمنشور على Facebook. يمكن استخدام مجموعة البيانات هذه لتصنيف المقالات الإخبارية التي ستكون محور تركيزنافي هذه المقالة وللتحليل العاطفي للرأي العام المغربي. يمكنك تحميل مجموعة البيانات من خلال الرابط.

هذه المقالة موجهة للأشخاص الذين لديهم القليل من المعرفة حول التعلم الآلي، على سبيل المثال، ما هو الفرق بين التصنيف classification والانحدار regression، وما هو التحقق من الصحة المتقاطع cross validation. ومع ذلك، سأقدم شرحًا موجزًا للخطوات المتبعة للمشروع.

مقدمة المشكلة

لحسن الحظ، تحتوي مجموعة البيانات الخاصة بنا على كل من المقالات والتسميات الخاصة بها، لذلك نحن نتعامل مع مشكلة التعلم الخاضع للإشراف والتي ستجعل حياتنا أسهل بكثير، لأنه إذا لم يكن الأمر كذلك، فسيتعين علينا تصنيف كل مقالة يدويًا أو اتباع نهج غير خاضع للأشراف.

باختصار، هدفنا هو التنبؤ بموضوع المقال بالنظر إلى نصه.في المجموع لدينا 11 موضوعا:

- الأمازيغية (لغة مغربية) Tamazight (A Moroccan Language).
- رياضة (رياضة) Sport (Sport).
- المجتمع (المجتمع) Societe (Society).
- المناطق (المناطق) Regions (Regions).
- السياسة (السياسة) Politique (Politics).
- المدارات (أخبار العالم) Orbites (World news).
- الاعلام (أخبار من الصحف المحلية) Medias (News from local newspapers).
- مغاربة العالم (مغاربة العالم) Marocains Du Monde (Moroccans of the world).

- فيت دايفرز (متنوعة) Faits Divers (Miscellaneous).
- الاقتصاد (الاقتصاد) Economie (Economy).
- الفن والثقافة (الفن والثقافة) Art Et Culture (Art and culture).

تحليل البيانات الاستكشافية

سنستخدم seaborn لتصور البيانات وpandas لمعالجة البيانات.

لنبدأ بتحميل البيانات:

نظرًا لأنه يتم تخزين البياناتفي ملفات مختلفة، يحتوي كل ملف على بيانات لموضوع معين، سيتعين علينا تكرار المواضيع وتسلسل النتائج.

```
import pandas as pd
stories=pd.DataFrame()
topics["tamazight","sport","societe","regions","politique","orbites
","medias","marocains-du-monde","faits-divers","economie","art-et-
culture"]for topic in topics:

stories=pd.concat([stories,pd.read_csv("stories_"+topic+".csv")])st
ories.drop(columns=["Unnamed: 0"],axis=1,inplace=True)
```

لنأخذ بعد ذلك عينة من البيانات:

```
stories.sample(5)
```

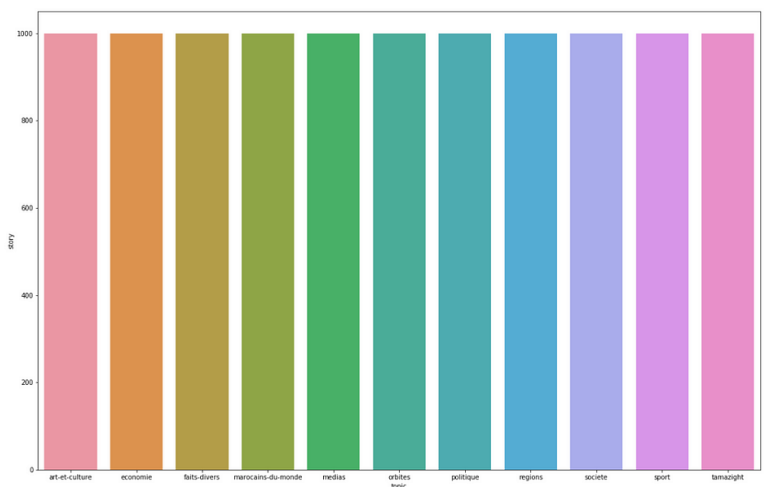
	id	title	date	author	story	topic
680	81e820dc055b11eb81e8646e69d991ea	كروردا يُلغي 100 جنوب إفريقي على أرض المغرب	الخميس 02 أبريل 2020 - 15:40	هسبريس - مصطفى شاكوتي	مازال مئات السياح الأجانب عالقين في المملكة ...ال	orbites
911	c53a3a1804e811ebb473646e69d991ea	وزارة التربية تعلن حصيلة تعميم التعليم الأولي	الاثنين 20 يوليوز 2020 - 10:40	هسبريس من الرباط	أعلنت وزارة التربية الوطنية عن حصيلة البرنامج ...	societe
193	0c2fc51804ef11eba4c9646e69d991ea	تتويج 5 فراء مغربية بجائزة فرانية كبرى في أبوظبي	الأحد 17 ماي 2020 - 14:11	هسبريس - و.ع	فاز 5 فراء مغربية من التكور والاذان بجائزة ...ال	marocains-du-monde
658	05a01a2e055211ebb27a646e69d991ea	في تلك الحارات" تعزز الرصيد القصصي لزروال	السبت 08 فبراير 2020 - 08:35	هسبريس من الرباط	تعزز الرصيد القصصي للقاص عبد الله زروال بمجموع...	art-et-culture
708	9e2c20dc055b11eb878c646e69d991ea	مؤلف يُقارب تيمة "حراك الريف" باللغة الفرنسية	الثلاثاء 24 مارس 2020 - 01:24	هسبريس من الدار البيضاء	حراك الريف.. في مساءلة حكامة الأزمة وأزمة" ...الخ	orbites

أعمدة عينة من مجموعة بيانات القصص

يمكننا أن نرى أن لدينا 5 أعمدة،في هذه المقالة نحن مهتمون فقط بالقصة وميزات الموضوع.

الآن دعونا نتحقق من عدد القصص لديناي كل موضوع، وهذا مهم للغاية للتصنيف لأنه إذا كان لدينا مجموعة بيانات غير متوازنة imbalanced dataset، أي (لدينا نقاط بيانات أكثر بكثيرفي موضوع ما من المواضيع الأخرى) فسيكون نموذجنا متحيزًا biased ولن يعمل أيضًا. إذا كانت لدينا هذه المشكلة، فإن أحد الحلول الشائعة هو تطبيق طريقة أخذ العينات under sampling أو الإفراطفي أخذ العينات oversampling، فلن نتناول التفاصيل لأنها ليستفي نطاق مقالتنا.

```
import seaborn as sns
storiesByTopic=stories.groupby(by="topic").count()["story"]
sns.barplot(x=storiesByTopic.index,y=storiesByTopic)
```



عدد القصص حسب الموضوع

يمكننا أن نرى أن لدينا ما يقرب من 1000 قصة لكل موضوع، ومجموعة البيانات لدينا متوازنة تمامًا.

تنظيف البيانات

نحن نتعامل مع بيانات النص العربي. ستتكون عملية تنظيف البيانات data cleaning لدينا من خطوتين:

إزالة كلمات التوقف Removing Stop Words: بعض الكلمات مثل "و"، "كيف" لها تكرار كبير للغايةفي جميع النصوص العربية ولا تقدم أي معنى يمكن أن يستخدمه نموذجنا للتنبؤ. ستؤدي إزالتها إلى تقليل الضوضاء noise والسماح لنموذجنا بالتركيز فقط على الكلمات ذات الصلة. للقيام بذلك، سنستخدم قائمة ونقوم بمراجعة جميع المقالات وإزالة جميع الكلمات التي تظهرفي القائمة.

قائمة كلمات التوقف التي استخدمتها متاحة على Github.

```
from nltk.tokenize import word_tokenizefile1 =
open('stopwordsarabic.txt', 'r', encoding='utf-8')
stopwords_arabic =
file1.read().splitlines()+["المغربي","المغربية","المغرب"]def
removeStopWords(text,stopwords):
    text_tokens = word_tokenize(text)
    return " ".join([word for word in text_tokens if not word in
stopwords])
```

إزالة علامات الترقيم Removing Punctuation: للسبب نفسه، سنقوم بإزالة علامات الترقيم، ولهذا استخدمت تعبير Regex.

```
from nltk.tokenize import RegexpTokenizer
def removePunctuation(text):
tokenizer = RegexpTokenizer(r'\w+')
return " ".join(tokenizer.tokenize(text))
```

رسم WordCloud

دعونا نستمتع ببعض المرح، سنقوم برسم Word Cloud من جميع القصص الموجودةفي DataSet لدينا باستخدام مكتبة بايثون " WordCloud"

قبل القيام بذلك، هناك بعض الخطوات الإضافية اللازمة للغة العربية، لمعرفة المزيد عنها قم بزيارة هذا الرابط.

```
import arabic_reshaper
from bidi.algorithm import get_display
import matplotlib.pyplot as plt
%matplotlib inlinedef
preprocessText(text,stopwords,wordcloud=False):
    noStop=removeStopWords(text,stopwords)
    noPunctuation=removePunctuation(noStop)
    if wordcloud:
        text=arabic_reshaper.reshape(noPunctuation)
        text=get_display(text)
        return text
    return
noPunctuationdrawWordcloud(stories.story,stopwords_arabic)
```

سحابة الكلمات لمقالات أخبار هسبريس

نظرًا لأن مجموعة البيانات هذه تحتوي على مقالات إخبارية حديثة، فإننا نرى كلمة "كورونا" (فيروس كورونا) كلمة متكررة. وهناك أيضا "الأمازيغية" وهي لغة رئيسيةفي المغرب، و"محمد" وهو الاسم الأكثر شعبيةفي المغرب وهو أيضا اسم ملك المغرب، و"الحكومة" وتعني الحكومة.

هندسة الميزات

نماذج التعلم الآلي هيفي جوهرها معادلات رياضية ولا يمكنها فهم النص، لذا قبل تشغيل نماذجنا، نحتاج إلى تحويل النص إلى أرقام، هناك طرق متعددة للقيام بذلك، دعنا نكتشف الطريقتين الأكثر شيوعًا.

عدد الكلمات

هذا بسيط للغاية، كل عمود يمثل كلمة من مجموعة القصص بأكملها، وكل صف يمثل قصة، وقيم الخلية هي التكرار الذي تظهر به الكلمةفي القصة!

TF–IDF

يرمز TF-IDF إلى "Term Frequency Inverse Document Frequency"، وهو يستخدم أسلوبًا أكثر تعقيدًا بعض الشيء والذي سيعاقب الكلمات الشائعة التي تحدثفي مستندات متعددة.

سوف نستخدم TF-IDF لأنهفي معظم الحالات يقدم أداءً أفضل!

```
from sklearn.feature_extraction.text import TfidfVectorizer#Clean
the stories
stories["storyClean"]=stories["story"].apply(lambda s:
preprocessText(s,stopwords_arabic))#Vectorize the storiesvectorizer
= TfidfVectorizer()
X = vectorizer.fit_transform(stories["storyClean"])
y=stories.topic
```

النمذجة

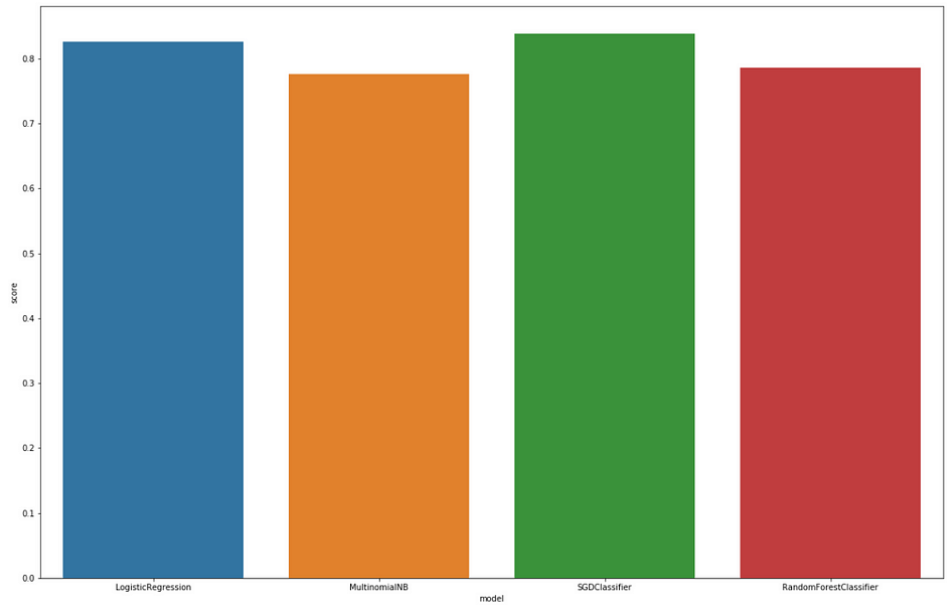
سنحاول النماذج التالية:

- الغابة العشوائية Random Forest.
- الانحدار اللوجستي Logistic Regression.
- مصنف التدرج الاشتقاقي العشوائي SGDClassifier
- نايف بايز متعدد الحدود Multinomial Naïve Bayes.

سنقوم بتشغيل البيانات من خلال كل نموذج ونستخدم الدقة accuracy وهي نسبة التنبؤات الصحيحة وإجمالي نقاط البيانات كمقياس لدينا، وللحصول على نتائج أكثر دقة استخدمنا التحقق المتقاطع cross validation مع 5 أضعاف folds لتسجيلنا ثم سنقوم برسم النتائج.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
import numpy as np
from sklearn.metrics import classification_report
def testModel(model,X,y):
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.2, random_state=42)
model.fit(X_train,y_train)
modelName = type(model).__name__
pred=model.predict(X_test)
print(modelName)
print(classification_report(y_test,model.predict(X_test)))
score=np.mean(cross_val_score(model, X, y, cv=5))

return model,{"model":modelName,"score":score}
```



دقة النماذج

أفضل نموذج لدينا هو SDGClassifier بدقة 87%.

تفسير النموذج

الآن بعد أن حصلنا على نموذج عملي، دعونا نحاول أن نفهم أكثر قليلاً ما يحدث، لذلك سنجيب على سؤالين:

- ما هي المواضيع التي يعاني منها نموذجنا؟
- ما هي الكلمات الأكثر تأثيرافي التنبؤ بالموضوعات المختلفة؟

بالنسبة للأسئلة الأولى يمكننا التحقق من تقرير التصنيف classification report لأفضل نموذج لدينا:

SGDClassifier

	precision	recall	f1-score	support
art-et-culture	0.90	0.93	0.92	213
economie	0.81	0.91	0.86	194
faits-divers	0.95	0.95	0.95	198
marocains-du-monde	0.85	0.91	0.88	178
medias	0.96	0.93	0.94	210
orbites	0.78	0.69	0.73	204
politique	0.80	0.84	0.82	197
regions	0.84	0.80	0.82	214
societe	0.73	0.70	0.71	184
sport	0.99	0.98	0.99	202
tamazight	0.97	0.98	0.97	206
accuracy			0.88	2200
macro avg	0.87	0.87	0.87	2200
weighted avg	0.87	0.88	0.87	2200

تقرير التصنيف SGDClassifier

نحن نتوقع "Sport" و"Art" و"Medias" و"Tamazight" بدقة عالية للغاية. نحن نعاني أكثر من غيرنا مع "orbites" ((world news)، و"societe" (Society)، وقد يكون هذا بسبب أن هذين الموضوعين أكثر عمومية واتساعًا.

للإجابة على السؤال الثاني، سنستخدم خاصية مفيدة للانحدار اللوجستي، فيمكننا استخدام الأوزان كمقياس لأهمية الكلمات في كل نموذج. مكتبة بايثون "ELI5" تجعل من السهل القيام بذلك:

y=art-et-culture top features		y=economie top features		y=faits-divers top features		y=marocains-du-monde top features		y=medias top features		y=orbites top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature
+4.092	الفنان	+4.244	العائلة	+3.792	عناصر	+5.475	الجالية	+6.476	المساء	+3.901	الله
+3.266	الفيلم	+3.098	المقاولات	+3.070	التهمة	+5.439	المغاربة	+4.881	الجريدة	+2.182	الدول
+3.128	فيلم	+3.082	الاقتصاد	+3.066	الدرك	+4.865	المغربية	+4.344	الصحافة	+2.151	العالم
+3.036	الثقافة	+2.969	الشركة	+2.596	المختصة	+4.200	مغاربة	+4.270	أخبار	+2.086	الملك
+2.875	الفنية	+2.922	شركة	+2.453	مصالح	+3.279	المالقين	+4.142	الصحافي	+2.084	المتحدة
+2.653	الكتاب	+2.655	الشركات	+2.434	العامة	+3.008	الهجرة	+3.744	الإعلام	+1.875	الأمريكية
+2.535	السينما	+2.371	القطاع	+2.405	المشتبه	+2.991	بالخارج	+3.226	المدير	+1.842	الإسلامي
+2.535	الفنانين	+2.173	التجارية	+2.357	الملكي	+2.893	القنصلية	+3.211	الفتاة	+1.619	الإنسان
+2.453	الأغنية	+2.133	البنك	+2.320	القضائية	+2.690	ألمانيا	+3.131	الأحداث	+1.618	السياسية
+2.451	الفن	+2.112	درهم	+2.153	توقيف	+2.626	المقيمين	+3.106	للصحافة	+1.538	السادس
... 45651 more positive ...		... 22905 more positive ...		... 11755 more positive ...		... 31792 more positive ...		... 39685 more positive ...		... 56660 more positive ...	
... 131031 more negative ...		... 153777 more negative ...		... 164927 more negative ...		... 144890 more negative ...		... 136997 more negative ...		... 120022 more negative ...	
-1.165	حالة	-1.013	التعليم	-1.171	كورونا	-0.864	الرباط	-0.589	لجريدة	-0.920	العثماني
-1.166	الصحة	-1.021	المغربي	-1.205	الحكومة	-0.980	العمومية	-0.625	<BIAS>	-0.999	الأمازيغي
-1.278	الحكومة	-1.071	الأمازيغية	-1.558	المغربي	-1.176	الصحية	-0.722	هسبريس	-1.007	بمدينة
-1.545	الأمازيغي	-1.085	الإنسان	-1.729	المغرب	-1.196	الأمازيغية	-0.756	لهسبريس	-1.213	البيضاء
-1.928	الأمازيغية	-1.099	عيد	-1.802	المغربية	-1.409	كورونا	-0.861	الأمازيغية	-1.771	الأمازيغية

y=politique top features		y=regions top features		y=societe top features		y=sport top features		y=tamazight top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature
+3.852	الحكومة	+2.817	الإقليمية	+3.866	الصحة	+6.216	القدم	+12.977	الأمازيغية
+3.584	الحزب	+2.686	بإقليم	+2.617	وزارة	+5.257	لكرة	+6.988	الأمازيغي
+3.399	النواب	+2.459	المحلية	+2.527	التعليم	+4.430	الفريق	+3.683	الأمازيغ
+3.201	الخارجية	+2.452	المديرية	+2.482	التربية	+3.854	الدوري	+2.165	بالأمازيغية
+2.900	العثماني	+2.442	حالات	+2.112	الإضراب	+3.773	اللاعب	+2.054	أمازيغية
+2.841	السياسي	+2.363	إقليم	+2.059	التلاميذ	+3.465	الرياضي	+1.881	اللغة
+2.734	حزب	+2.204	بفيروس	+2.001	الوزارة	+3.091	النادي	+1.830	للأمازيغية
+2.647	الأحزاب	+2.145	حالة	+1.827	المواطنين	+3.084	نادي	+1.805	سوس
+2.484	مجلس	+2.069	المستجد	+1.822	الصحية	+2.662	الموسم	+1.634	أمازيغ
+2.437	لحزب	+1.867	الدار	+1.815	الأصحي	+2.591	مباراة	+1.576	التنظيمي
... 23056 more positive ...		... 17657 more positive ...		... 26213 more positive ...		... 16367 more positive ...		... 38807 more positive ...	
... 153626 more negative ...		... 159025 more negative ...		... 150469 more negative ...		... 160315 more negative ...		... 137875 more negative ...	
-0.777	اللغة	-1.452	المغربي	-1.055	العالم	-0.919	الملك	-0.901	فيروس
-0.802	بالمغرب	-1.492	المغاربة	-1.082	المساء	-0.929	هسبريس	-0.907	الله
-0.805	الشركة	-1.618	الحكومة	-1.116	الشركة	-0.947	المغاربة	-0.910	الصحي
-0.869	الجمعية	-2.376	المغربية	-1.216	بإقليم	-1.122	أن	-1.047	الصحية
-2.033	الأمازيغية	-2.956	المغرب	-1.473	الأمازيغية	-1.329	المغرب	-1.839	كورونا

يمكننا أن نرى أن معظم الكلمات منطقية وتتوافق مع موضوع الموضوع، على سبيل المثال بالنسبة لـ "Art"، فإن الكلمات الرئيسية هي: "Artist"، "Film"، "Culture"، "Book".

الاستنتاج

لقد قمنافي هذه المقالة بجميع الخطوات اللازمة لتصميم نظام تصنيف النصوص للغة العربية من استكشاف البيانات إلى تفسير النماذج. ومع ذلك، لا يزال بإمكاننا تحسين دقتنا عن طريق ضبط المعلمات الفائقة.

المصدر:

https://medium.com/towards-data-science/arabic-topic-classification-on-the-hespress-news-dataset-7adceef12bed

10) تصنيف الكتب العربية باستخدام المعالجة اللغوية الطبيعية Arabic books classification using NLP

تُعد المعالجة اللغوية الطبيعية Natural Language Processing أحد أكثر مجالات البحث نشاطًا في علم البيانات data science اليوم. هذه منطقة تقع عند تقاطع التعلم الآلي Machine Learning واللغويات linguistics. والغرض منه هو استخراج المعلومات والمعنى من محتوى النص.

تجد المعالجة اللغوية الطبيعية (NLP) العديد من التطبيقاتفي الحياة اليومية:

- ترجمة النصوص text translation (التعلم العميق Deep Learning على سبيل المثال).
- مدقق املائي spell checker.
- الملخص التلقائي للمحتوى automatic summary of content.
- التركيب الصوتي vocal synthesis.
- تصنيف النص text classification.
- تحليل الرأي / الشعور opinion / feeling analysis.
- التنبؤ بالكلمة التالية next word prediction على الهاتف الذكي.
- استخراج الكيانات المسماة extracting named entities من النص.

تتكون المعالجة اللغوية الطبيعية عمومًا من مرحلتين إلى ثلاث مراحل رئيسية:

1. **المعالجة المسبقة Pre-processing:** وهي خطوة تهدف إلى توحيد النص لتسهيل استخدامه.
2. **تمثيل النص كمتجه Representation of the text as a vector:** يمكن تنفيذ هذه الخطوة باستخدام تقنيات حقيبة الكلمات Bag of Words أو Tf-IdF. يمكننا أيضًا تعلم تمثيل المتجهات (التضمين) من خلال التعلم العميق.
3. **التصنيف classification** وهو حال هذه المادة.

لذلك،في هذه المقالة، سنغطي مهام المعالجة اللغوية الطبيعية الأكثر شيوعًا والتي توجد لها أدوات خاصة باللغة العربية.

مجموعة البيانات

بالنظر إلى مجموعة بيانات Kaggle: https://www.kaggle.com/dareenalharthi/jamalon-arabic-books-dataset،

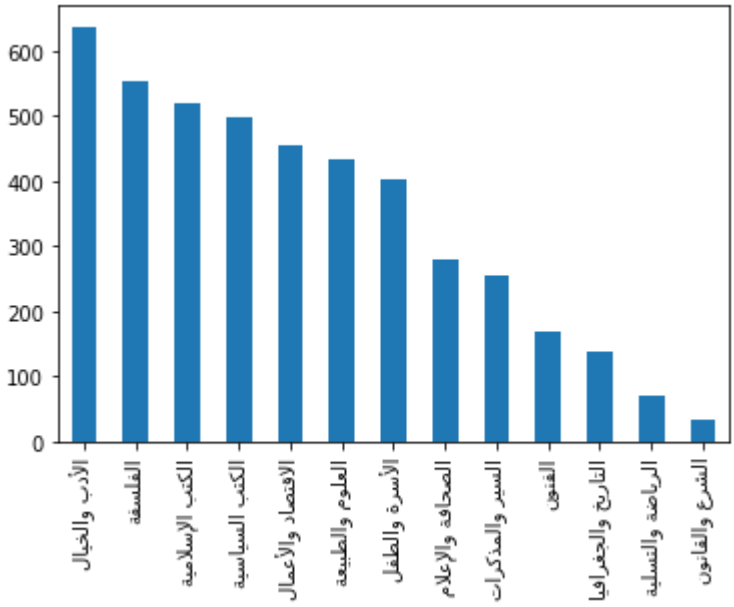
Jamalon(العربية: جملون) هو متجر لبيع الكتب بالتجزئة عبر الإنترنت مقرهفي عمّان، ويقوم بشحن الكتب إلى القراءفي جميع أنحاء الشرق الأوسط. تحتوي مجموعة البيانات الخاصة بنا على جميع الكتب التي شحنها جملون. نحن بحاجة إلى تصنيف أوصافهم للتنبؤ بالفئة المناسبة.

سنبدأ بالمعالجة المسبقة للبيانات مع بعض التصور لفهم هدف تحليلنا متبوعًا بخوارزمية التعلم الآلي لتصنيف النص.

	Title	Author	Description	Pages	Publication year	Publisher	Cover	Category	Subcategory	Price
0	في فقه الصراع على القدس وفلسطين	محمد عمارة	الإسلامية كانت القدس رمز الصراع وبوابة الانتص...	180	2006	دار الشروق – مصر	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	15.00
1	عذراء قريش	جرجي زيدان	روايات تاريخ الإسلام" سلسلة من الروايات التار...	176	0	دار البشير للطباعة والنشر والتوزيع	غلاف عادي	الأدب والخيال	الأدب الإسلامي	18.75
2	نفحات من الأدب الإسلامي	محمد الصابوني	هذه مذكرات في الأدب الإسلامي، وضعها المؤلف لطلابه...	168	1996	دار البشائر الإسلامية للطباعة والنشر والتوزيع	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	18.75
3	بسط الأعذار عن حب العذار	بدر الدين المنهاجي	كتاب في الادب وصفه المؤلف بعد أن أطلع على كتا...	464	2016	دار الكتب العلمية	غلاف كرتوني	الأدب والخيال	الأدب الإسلامي	45.00
4	قصة كاملة... لم يؤلفها بشر	علي الطنطاوي	قصة حقيقية شهد نهايتها العلامة الدمشقي علي ال...	34	2004	دار ابن حزم للطباعة والنشر والتوزيع	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	1.50
5	أسير المتمهدي	جرجي زيدان	روايات تاريخ الإسلام" سلسلة من الروايات التار...	144	0	دار البشير للطباعة والنشر والتوزيع	غلاف عادي	الأدب والخيال	الأدب الإسلامي	18.75
6	ما قرأت في الصحن الحيدري لشعراء المنبر الحسيني	محمد العوادي	مجموعة من القصائد لعدد من الشعراء الحسينيين ...من	512	2009	مؤسسة البلاغ للطباعة والنشر والتوزيع	غلاف كرتوني	الأدب والخيال	الأدب الإسلامي	45.00

نظرة عامة على المتغيرات الرئيسيةفي مجموعة البيانات هذه.

شكل البيانات هو (4443، 10) وسيكون أداتنا الرئيسية؛ لذلك، سيتم انشاء مجموعاتنا الفرعية من التدريب train والاختبار test منه.



واحدة من أكبر المشاكل التي كان علينا أن نواجهها هي أن مصدر الوصف غير مكتوب بشكل جيد. هناك الكثير من الأخطاء النحوية والمفردات، لذا ستكون خطوة المعالجة المسبقة هي الخطوة الأكثر أهمية لتقييم نموذجنا.

المعالجة المسبقة للبيانات

يعد هذا جزءًا مهمًا من تحليلنا، وقد يستغرق تنظيف البيانات ما يصل إلى 70 أو 80% من وقتنا. بالنسبة لمشروعنا، بدأنا بقراءة بعض الأوصاف ومن المدهش أننا اكتشفنا أنه سيكون لدينا الكثير من العمل للقيام به. بعض المشاكل التي سنواجهها هي:

- الأخطاء النحوية والمفردات Grammar and vocabulary errors.
- الكتب المكررة والأوصاف Duplicate books and descriptions.
- كتب بدون أوصاف Books without descriptions أو تصنيفات categories ونحو ذلك.

أول شيء يجب فعله هو رؤية معدل ليس رقماً NaNفي البيانات وتحديد إما الإزالة أو الاستبدال بالتقريب المناسب.في هذه الحالة من البيانات المفقودة هو نص ISO سنقوم فقط بإزالة الصفوف المعنية.

الخطوة التالية هي البدءفي تنظيف النصفي الأوصاف. إنه نص عربي، لذا سنقوم بإزالة علامات الترقيم وعلامات التشكيل العربية وتسوية بعض الحروف المشتركة وإزالة الأرقام.

هناك مشكلة أخرى وهي أن بعض الكلمات خاطئة نحويًا أو حتى بعضها غير موجود بسبب نسيان مسافة بين كلمتين أو 3 أو حتى كلمات خاطئة تم إدراجهافي مجموعة البيانات. الطريقة الأكثر فعالية التي وجدناها مفيدة هي تشغيل فصل تصحيح باستخدام بعض مجموعات البيانات التي تحتوي تقريبًا على الكلمات العربية الموجودة بالكامل.

تختلف الأوصاف الآن بشكل كبير عن الأوصاف المقدمةفي البداية ويمكننا الآن بدء بعض المعالجة المسبقة الحقيقية.

الترميز وإزالة كلمات التوقف

يسعى التوكينازيشن (الترميز) Tokenization إلى تحويل النص إلى سلسلة من التوكن الفردية individual tokens.في الفكرة، يمثل كل توكن كلمة، ويبدو أن تحديد الكلمات مهمة بسيطة نسبيًا. الآن يمكننا بسهولة إزالة كلمات التوقف stopwords التي تعتبر كلمات غير ضرورية والتي حتى لو تمت إزالتها فلن يتغير معنى الجملة. من أجل الحصول على أكبر عدد ممكن من الكلمات التوقف

العربية، قمنا بدمج ملفين نصيين يحتويان على غالبية الكلمات التوقف العربية وقمنا بإزالة الملفات المتكررة.

تصنيف اقسام الكلام

من أجل ربط كل كلمة بالمعلومات النحوية المقابلة لها، جربنا عددًا كبيرًا جدًا من تصنيف اقسام الكلام pos tagging للمجموعة العربية ولكن النتائج كانت غير متسقة للغاية.

أخيرًا، عملنا مع فئة class لتصنيف اقسام الكلام مع Stanford Tagger. الإدخال هو المسارات إلى نموذج تم تدريبه على بيانات التدريب والمسار إلى ملف جرة علامات Stanford. بلغت دقة العلامة 90٪ وكانت النتائج مهمة للغاية.

```
[('', 'روا/VBD'),
 ('', 'تاريخ/NN'),
 ('', 'إسلام/NN'),
 ('', 'سلسل/NN'),
 ('', 'روا/VBD'),
 ('', 'تاريخ/NN'),
 ('', 'تصور/NN'),
 ('', 'مراحل/NN'),
 ('', 'تاريخ/NN'),
 ('', 'إسلام/NN'),
 ('', 'ظهور/NN'),
 ('', 'إسلام/NN'),
 ('', 'روع/NN'),
 ('', 'عنصر/NN'),
 ('', 'تشويق/NN'),
 ('', 'تزام/NN'),
 ('', 'حوادث/NN'),
 ('', 'تاريخ/NN'),
 ('', 'تزاما/NN'),
 ('', 'دقيقا/JJ'),
```

جذر وإزالة الكلمات التي تحتوي على أقل من 3 أحرف.

بالنسبة إلى أصل الكلمة، عملنا مع دالتنا الخاصة "LightStemming" حيث حددنا جميع اللواحق والحروف غير المرغوب فيها التي نريد إزالتها من أجل الحصول على أصل الكلمة word's stem. لقد طبقنا هذه الدالة فقط على الكلمات التي تحتوي على أكثر من 3 أحرف، والكلمات التي تحتوي على أقل من 3 هي إما كلمات توقف قمنا بإزالتها بالفعل أو كلمات أخرى ذات معنى أقل قمنا بإزالتها أيضًا.

القيم المتطرفة

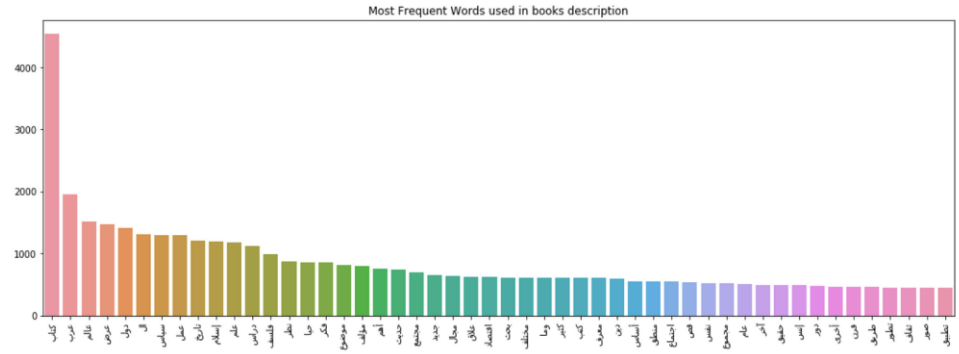
لقد حاولنا اكتشاف القيم المتطرفة outliersفي مجموعة البيانات الخاصة بنا من خلال تقنية تعتمد على التعلم العميق غير الخاضع للأشراف للكشف عن القيم المتطرفة للبيانات النصية. قمنا بتحويل النص إلى ميزات رقمية ثم استخدمنا شبكة عصبية ذات الترميز التلقائي auto-encoder Neural

Network للتدريب الذاتي واستخدامها كنموذج. يمكننا تلخيص العمل الذي قمنا بهفي الخطوات الثلاث التالية:

1- تحويل النصوص إلى نموذج الفضاء المتجه Vector Space Model.
2- تدريب الشبكة العصبية ذات الترميز التلقائي.
3- قياس التشابه Similarity measure،في هذه الخطوة نقوم بفرز جميع قيم تشابه جيب التمام cosine similarity بترتيب متزايد واختيار القيم "k" العليا ('k') هي مدخلات المستخدم المحددة) وستكون القيم المتطرفة المحتملة هي نقاط البيانات المقابلة.

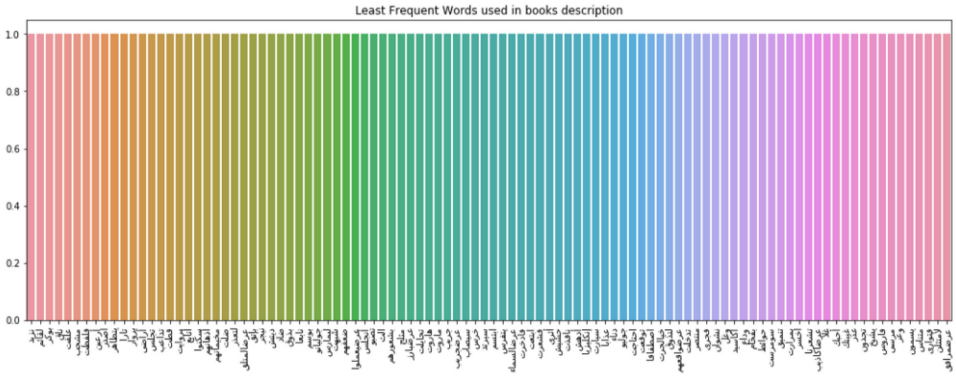
تكرار الكلمات لكل فئة (الكلمات الأكثر تكرارًا والأقل تكرارًا)

لاستكشاف البيانات، حاولنا العثور على الـ 50 كلمة words/الرموز tokens الأكثر شيوعًا الموجودة في الرسومات البيانية.



من المخطط الشريطي أعلاه، يمكننا أن نرى بسهولة أن "الكتاب" هي الكلمة الأكثر شيوعًافي جميع الكتب من هذه الفئة.

بعد ذلك، دعونا نفعل العكس، من خلال معرفة الـ 500 كلمة/الرمز الأقل تكرارًافي جميع الكتب ورسم الرسم البياني.



بفضل الرسم البياني أعلاه، يمكننا أن نرى بوضوح أن هذه الكلمات ليس لها أي معنىفي اللغة العربية. لذلك قررنا إزالة كل هذه الكلمات (الكلمات الأقل تكرارا).

سحابة الكلمات

لدينا أخيرًا تمثيل مرئي أكثر وضوحًا للكلمات الرئيسية الأكثر استخدامًافي أوصاف كل فئة.

هندسة الميزات

في هذه الخطوة، سيتم تحويل بيانات النص الخام إلى متجهات الميزات feature vectors وسيتم إنشاء ميزات جديدة باستخدام مجموعة البيانات الموجودة. سنقوم بتنفيذ الأفكار المختلفة التالية من أجل الحصول على الميزات ذات الصلة من مجموعة البيانات الخاصة بنا.

تمثل درجة TF-IDF الأهمية النسبية للمصطلحفي الوثيقة والمجموعة بأكملها. تتكون درجة TF-IDF من فترتين: الأول يحسب تردد المصطلح المعياري normalized Term Frequency

(TF)، والمصطلح الثاني هو تردد المستند العكسي Inverse Document Frequency (IDF)، والذي يتم حسابه على شكل لوغاريتم عدد المستنداتفي المجموعة مقسومًا على الرقم. من المستندات التي يظهر فيها المصطلح المحدد.

- TF(t) = (عدد مرات ظهور المصطلح tفي المستند) / (إجمالي عدد المصطلحاتفي المستند)
- IDF(t) = log_e (إجمالي عدد المستندات / عدد المستندات التي تحتوي على المصطلح t)

يمكن إنشاء متجهات TF-IDF على مستويات مختلفة من رموز الإدخال (الكلمات words والأحرف characters والجرامات n n-grams)

Word Level TF-IDF: مصفوفة تمثل درجات tf-idf لكل مصطلحفي مستندات مختلفة.

N-gram Level TF-IDF: عبارة عن مزيج من مصطلحات N معًا. تمثل هذه المصفوفة عشرات tf-idf من N-grams.

Character Level TF-IDF: مصفوفة تمثل درجات tf-idf لمستوى الحرف n-gramsفي المجموعة.

بناء نموذج

الخطوة الأخيرةفي إطار تصنيف النص هي تدريب المصنف باستخدام الميزات التي تم إنشاؤها مسبقًا. هناك العديد من الخيارات المختلفة لنماذج التعلم الآلي التي يمكن استخدامها لتدريب النموذج النهائي. سنقوم بتنفيذ المصنفات المختلفة التالية لهذا الغرض:

- مصنف نايف بايز Naive Bayes Classifier.
- المصنف الخطي Linear Classifier.
- آلة المتجهات الداعمة Support Vector Machine.
- نموذج الغابة العشوائية Random forest mode.
- نماذج التعزيز Boosting Models.

دعونا نشغل النماذج ونفهم تفاصيلها. الدالة التالية هي دالة مساعدة يمكن استخدامها لتدريب النموذج. فهو يقبل المصنف ومتجه الميزات لبيانات التدريب وتسميات بيانات التدريب ومتجهات الميزات للبيانات الصالحة كمدخلات. وباستخدام هذه المدخلات، يتم تدريب النموذج وحساب درجة الدقة.

نايف بايز

نايف بايز Naive Bayes هي تقنية تصنيف تعتمد على نظرية بايز Bayes' Theorem مع افتراض الاستقلال بين المتنبئين. يفترض مصنف Naive Bayes أن وجود ميزة معينةفي الفئة لا علاقة له بوجود أي ميزة أخرى. لقد قمنا بتنفيذ نموذج نايف بايز باستخدام تطبيق sklearn مع ميزاتنا المختلفة.

دعونا نلقي نظرة على دقة نموذجنا لكل ميزة لمعرفة الميزة التي تمنحنا أعلى دقة.

```
from sklearn import naive_bayes
from sklearn import metrics
# Naive Bayes on Count Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xtest_count)
print("NB, Count Vectors: ", accuracy)

# Naive Bayes on Word Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf, train_y, xtest_tfidf)
print("NB, WordLevel TF-IDF: ", accuracy)

# Naive Bayes on Ngram Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram, train_y, xtest_tfidf_ngram)
print("NB, N-Gram Vectors: ", accuracy)

# Naive Bayes on Character Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram_chars, train_y, xtest_tfidf_ngram_chars)
print("NB, CharLevel Vectors: ", accuracy)
```

```
NB, Count Vectors:  0.5859728506787331
NB, WordLevel TF-IDF:  0.5701357466063348
NB, N-Gram Vectors:  0.4242081447963801
NB, CharLevel Vectors:  0.5361990950226244
```

المصنف الخطي

يقيس الانحدار اللوجستي Logistic regression العلاقة بين المتغير التابع الفئوي categorical dependent variable وواحد أو أكثر من المتغيرات المستقلة independent variables عن طريق تقدير الاحتمالات باستخدام دالة لوجستية/سينية logistic/sigmoid function.

```
from sklearn import linear_model
# Linear Classifier on Count Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, train_y, xtest_count)
print("LR, Count Vectors: ", accuracy)

# Linear Classifier on Word Level TF IDF Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf, train_y, xtest_tfidf)
print("LR, WordLevel TF-IDF: ", accuracy)

# Linear Classifier on Ngram Level TF IDF Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngram, train_y, xtest_tfidf_ngram)
print("LR, N-Gram Vectors: ", accuracy)

# Linear Classifier on Character Level TF IDF Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngram_chars, train_y, xtest_tfidf_ngram_cha
print("LR, CharLevel Vectors: ", accuracy)
```

```
/Users/lamiaeaitmbirik/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarni
ng: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/lamiaeaitmbirik/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:469: FutureWarni
ng: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
LR, Count Vectors:  0.5995475113122172
LR, WordLevel TF-IDF:  0.6018099547511312
LR, N-Gram Vectors:  0.4117647058823529
LR, CharLevel Vectors:  0.581447963800905
```

نموذج SVM

آلة المتجهات الداعمة(Support Vector Machine (SVM عبارة عن خوارزمية تعلم آلي خاضعة للإشراف والتي يمكن استخدامها لكل من تحديات التصنيف أو الانحدار. يستخرج النموذج أفضل مستوى/خط فائق ممكن يفصل بين الفئتين.

```
from sklearn.linear_model import SGDClassifier

sgd = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=42, max_iter=5, tol=None
               ])
sgd.fit(X_train, y_train)

y_pred = sgd.predict(X_test)

print('accuracy %s' % accuracy_score(y_pred, y_test))
```

```
accuracy 0.5995475113122172
```

نموذج الغابة العشوائية

نماذج الغابة العشوائية Random Forest models هي نوع من نماذج المجموعة، وخاصة نماذج التعبئة bagging models. إنهم جزء من عائلة النماذج القائمة على الشجرة tree based model.

```
from sklearn import ensemble
from sklearn.ensemble import RandomForestRegressor

# RF on Count Vectors
accuracy = train_model(ensemble.RandomForestClassifier(), xtrain_count, train_y, xtest_count)
print ("RF, Count Vectors: ", accuracy)

# RF on Word Level TF IDF Vectors
accuracy = train_model(ensemble.RandomForestClassifier(), xtrain_tfidf, train_y, xtest_tfidf)
print ("RF, WordLevel TF-IDF: ", accuracy)
```

```
/Users/lamiaeaitmbirik/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: Th
e default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RF, Count Vectors:  0.502262443438914
```

```
/Users/lamiaeaitmbirik/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: Th
e default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RF, WordLevel TF-IDF:  0.5101809954751131
```

نموذج التعزيز

تعد نماذج التعزيز Boosting models نوعًا آخر من نماذج المجموعة وهي جزء من النماذج القائمة على الشجرة. Boosting عبارة عن خوارزمية وصفية لمجموعة التعلم الآلي لتقليل التحيز biasفي المقام الأول، وكذلك التباين varianceفي التعلم الخاضع للإشراف، ومجموعة من خوارزميات التعلم الآلي التي تحول المتعلمين الضعفاء إلى متعلمين أقوياء. يتم تعريف المتعلم الضعيف على أنه مصنف يرتبط بشكل طفيف فقط بالتصنيف الحقيقي (يمكنه تصنيف الأمثلة بشكل أفضل من التخمين العشوائي random guessing).

```
import xgboost
from sklearn import metrics

# Extereme Gradient Boosting on Count Vectors
accuracy = train_model(xgboost.XGBClassifier(), xtrain_count.tocsc(), train_y, xtest_count.tocsc())
print ("Xgb, Count Vectors: ", accuracy)

# Extereme Gradient Boosting on Word Level TF IDF Vectors
accuracy = train_model(xgboost.XGBClassifier(), xtrain_tfidf.tocsc(), train_y, xtest_tfidf.tocsc())
print ("Xgb, WordLevel TF-IDF: ", accuracy)

# Extereme Gradient Boosting on Character Level TF IDF Vectors
accuracy = train_model(xgboost.XGBClassifier(), xtrain_tfidf_ngram_chars.tocsc(), train_y, xtest_tfidf_ngram_chars.
print ("Xgb, CharLevel Vectors: ", accuracy)
```

```
Xgb, Count Vectors:  0.5163104611923509
Xgb, WordLevel TF-IDF:  0.5039370078740157
Xgb, CharLevel Vectors:  0.5646794150731158
```

```
max_review_length = 50 # The handful tweet are longer than 50 tokens
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
```

LSTM

لنبدأ باستيراد الفئات classes والدوال functions المطلوبة لهذا النموذج ونحتاج إلى اقتطاع تسلسلات الإدخال وحشوها بحيث تكون جميعها بنفس الطول للنمذجة. سيتعلم النموذج أن القيم الصفرية لا تحمل أي معلومات، لذا فإن التسلسلات ليست بنفس الطول من حيث المحتوى، ولكن مطلوب متجهات الطول نفسها لإجراء الحساب في Keras.

```
from keras.preprocessing.sequence import
pad_sequencesX=tokenizer.texts_to_sequences(df['Description2_Parsed
_13_unlist'].values)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
```

نحتاج الآن إلى تحويل الفئات الفئوية categorical categories إلى أرقام numbers.

```
Y = pd.get_dummies(df['Category']).values
```

تقسيم التدريب ـ اختبار:

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size
= 0.10, random_state = 42)
```

يمكننا الآن تحديد وتجميع وملاءمة fit نموذج LSTM الخاص بنا.

الطبقة الأولى هي الطبقة المضمنة التي تستخدم 100 متجه طول لتمثيل كل كلمة. الطبقة التالية هي طبقة LSTM التي تحتوي على 100 وحدة ذاكرة (خلايا عصبية ذكية). أخيرًا، نظرًا لأن هذه مشكلة تصنيف، يجب على طبقة الإخراج إنشاء 13 قيمة إخراج، واحدة لكل فئة.

نظرًا لأنها مشكلة تصنيف متعددة الفئات multi-class classification problem، يتم استخدام categorical_crossentropy كدالة الخسارة (الخطأ) loss function. يتم استخدام خوارزمية تحسين ADAM الفعالة. النموذج مناسب لـ 5 فقط. يتم استخدام حجم دفعة batch size كبير مكون من 64 مراجعة (reviews) لتوزيع تحديثات الوزن.

```
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM,
input_length=X.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(13, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())from keras.callbacks import EarlyStopping
epochs = 5
batch_size = 64
history = model.fit(X_train, Y_train, epochs=epochs,
batch_size=batch_size,validation_split=0.1,callbacks=[EarlyStopping
(monitor='val_loss', patience=3, min_delta=0.0001)])
```

Word2vec والتشابه

لقد عملنا مع نموذج مُدرب مسبقًا تم إنشاؤه باستخدام مكتبة بايثون Genism، مدخلاته هي الوصف description والفئات categories ومخرجاته عبارة عن مجموعة من المتجهات vectors (متجهات الميزات التي تمثل الكلماتفي الوصف أو الفئات)، ثم قمنا بتحديد دالة تسمى "Similarity" الذي يحسب المسافة بين vec1 و vec 2، vec1 يمثل الجملة الأولى (الوصف description) و vec2 يمثل الجملة الثانية (الفئة category) نحسب المسافات بين الوصف وجميع الفئات وأقرب مسافة إلى واحدة تتوافق مع فئة الوصف.

```
class TextSimilarity:def __init__(self):
        try:
            self.model =
gensim.models.KeyedVectors.load_word2vec_format('/Users/lamiaeaitmb
irik/Desktop/wiki.ar.vec')
            self.index2word_set = set(self.model.wv.index2word)
        except FileNotFoundError:
            raise FileNotFoundError
    def avg_feature_vector(self, sentence, num_features=300):
        words = word_tokenize(sentence)
        feature_vec = np.zeros((num_features, ), dtype='float32')
        n_words = 0
        for word in words:
            if word in self.index2word_set:
                n_words += 1
                feature_vec = np.add(feature_vec, self.model[word])
        if (n_words > 0):
            feature_vec = np.divide(feature_vec, n_words)
        return feature_vecdef similarity(self, sentence1,
sentence2):
        vec1, vec2 = self.avg_feature_vector(sentence1),
self.avg_feature_vector(sentence2)
        return self.cosine_similarity(vec1, vec2)def
cosine_similarity(self, vec1, vec2):
        return 1 - spatial.distance.cosine(vec1, vec2)
```

الاستنتاج

كان مشروعنا بشكل عام يدور حول التنبؤ بفئة الكتب باستخدام الوصف فقط دون المحتوى الفعلي للكتاب نفسه، والبيانات التي كانت لدينا للتعلم هيفي الواقع باللغة العربية وهي متناثرة جداً، وهي مشكلة قد تواجههافي العالم الحقيقي، نحن كان علينا أن نواجه العديد من التحديات من أجل تنفيذ مصنف أفضل (لغة غير منظمة Unstructured language، أخطاء إملائية Orthographic mistakes، تناقضات إملائية Spelling inconsistencies، أحرف غير معروفة Unknown caracters، أحرف متكررة Repeated letters ومسافاتفي الكلمات). لذلك، تعلمنا تقنيات جديدة وأصبحنا أكثر دراية بمجموعة واسعة المهامفي المعالجة اللغوية الطبيعية من الأساسي إلى المتقدم.

يمكن العثور على مشروع "Jupyter Notebook" لهذا الغرض هنا.

المصدر:

https://medium.com/@mustaphaamine.kamil/arabic-text-classification-using-nlp-b530565f18d

11) التلخيص التلقائي للنص العربي باستخدام بايثون
Automatic Arabic Text Summarization using Python

اللغة العربية هي اللغة الخامسة الأكثر انتشارًافي العالم؛ وهي من اللغات التي لم تتأثر بالتغيرات عبر القرون فهي لغة القرآن الكريم. شهد التلخيص التلقائي للنصوص automatic text summarizationفي هذه السنوات الأخيرة تطوراً مستمراً وملحوظاً نظراً لأهميته وتطبيقاته. ومع ذلك، فإن عدد الدراسات البحثية التي تتناول تلخيص النص العربي Arabic text summarization صغير نسبيًا مقارنة باللغات الأخرى.في هذا المشروع، نقترح تطوير أداة تمييز النص text highlighter tool التي تسمح للمستخدم بتلقي ملاحظات ملخصة حول النص العربي المحدد باستخدام EASC (Essex Arab Summaries Corpus). سيأخذ النظام المقترح نص مستند عربي واحد كمدخل ويعيد الجمل ذات الترتيب الأعلى المميزة كنتيجة نهائية. يتم تنفيذها بناءً على ثلاث طرق مختلفة، الطريقة الأولى التي حققت دقة 59% هي حقيبة الكلمات Bag of Words (BoW)، الطريقة الثانية التي حققت دقة 60% هي تكرار المصطلح ـ تردد الوثيقة العكسي erm Frequency - Inverse Document Frequency (TF-IDF)، والطريقة الثالثة التي حققت دقة 56% للمظهر الأول و48% للأقرب إلى المركز هي Word2Vec: نموذج SkipGram مع تكتل K-Means.

البدء

ستوفر لك هذه التعليمات نسخة من المشروع جاهزة للعمل على جهازك المحلي لأغراض التطوير والاختبار. راجع النشر للحصول على ملاحظات حول كيفية نشر المشروع على نظام مباشر.

المتطلبات الأساسية

المكتبات المستخدمة لملخص حقيبة الكلمات (BoW):

- للوصول إلى عنوان URL لمقالة واحدة.

```
from urllib import request
```

- لاستخراج معلومات الويب (محتوى المقالة).

```
from bs4 import BeautifulSoup as bs
```

- لتنظيف معلومات الويب (محتوى المقالة).

```
import re
```

- لاستيراد الجملة ومرمز الكلمات words tokenizer.

```
import nltk
```

- لاستيراد قائمة كلمات التوقف العربية Arabic stopwords.

```
from nltk.corpus import stopwords
```

- لاستيراد الجذع العربي Arabic stemmer (ISRI Arabstemer)

```
from nltk.stem.isri import ISRIStemmer
```

المكتبات المستخدمة لملخص تردد المصطلح - تردد المستند العكسي (TF-IDF):

- لاستيراد الجملة ومرمز الكلمات.

```
import nltk
```

المكتبات المستخدمة لملخص Word2Vec:

قم بتثبيت النموذج الذي تم تدريبه مسبقًا لتخطي جرام skip-gram بطول 100 متجه (ويكيبيديا) وملف الأداة المساعدة.

```
github.com/bakrianoo/aravec
install: full_grams_sg_100_wiki
```

- للوصول إلى عنوان URL لمقالة واحدة.

```
from urllib import request
```

- لاستخراج معلومات الويب (محتوى المقالة).

```
from bs4 import BeautifulSoup as bs
```

- لتنظيف معلومات الويب (محتوى المقالة).

```
import re
```

- لاستيراد الجملة ومرمز الكلمات.

```
import nltk
```

- لاستيراد قائمة كلمات التوقف العربية.

```
from nltk.corpus import stopwords
```

- لاستيراد واستخدام مجموعة KMEANS.

```
from sklearn.manifold import TSNE
from sklearn import cluster
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.cluster import KMeans
```

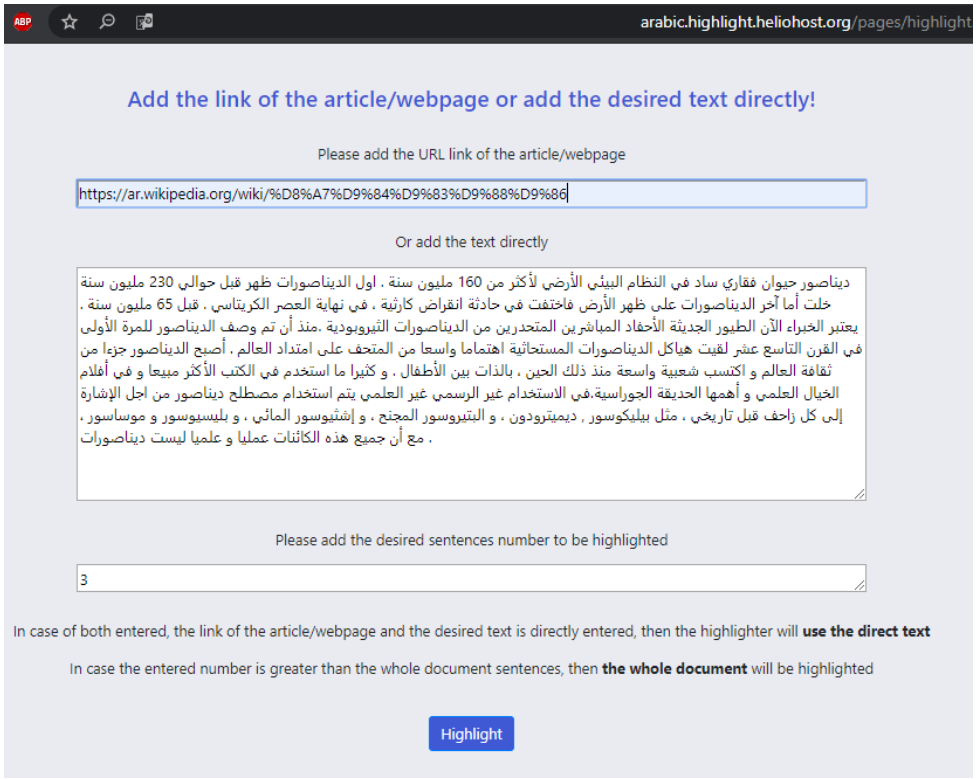
النشر

تم دمج ملفات python كنصوص CGI مع ملفات الويب التي تم تطويرها باستخدام HTML5\CSS3\JavaScript\PHP ونشرها باستثناء نموذج Word2Vec على الرابط التالي:
http://arabic.highlight.heliohost.org/

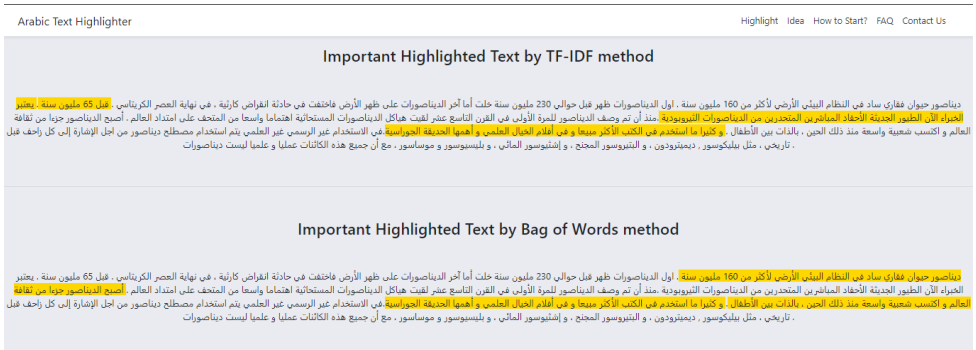
الاختبار

اختبار النشر

- **الإدخال:** المقالة أو الرابط الخاص بها مثلاً: https://ar.wikipedia.org/wiki/%D8%A7%D9%84%D9%83%D9%88%D9%86



- الإخراج: يتم تمثيل الملخص كعينة إخراج نص مميز highlighted text.



بُنيت مع:

- Anaconda Spyder: بيئة بايثون المستخدمة.

- heliohostRicky: خادم مضيف الويب.
- Visual Studio Code ـ محرر أكواد الويب.

المصدر:

https://github.com/RaghadAlshaikh/Automatic-Arabic-Text-Summarizer#arabic-text-summarization--text-highlighter-for-important-information-in-arabic-text

12) كيفية إنشاء بوت تويتر باستخدام التعلم العميق How to build Twitter bot using deep learning

كان الأسبوع الماضي هو الذكرى السنوية الثانية عشرة لي على تويتر. وبدلاً من الاحتفال به بتغريدة، قررت أن أفعل شيئًا أفضل: بوت الشخصي على تويتر my personalized Twitter bot.

الفكرة هي إنشاء بوت يكتب تغريدات منطقية ويبدو أنها كتبتها بنفسي.

المشكلة الأولى التي واجهتها هي أن معظم تغريداتي كانت باللغة العربية، وكانت معظم موارد المعالجة اللغوية الطبيعية Natural Language Processing (NLP) المتاحة باللغة الإنجليزية. الأمر الثاني هو أنني أكتب باللهجة المصرية، وهي مختلفة عن اللغة العربية الفصحى المستخدمةفي معظم الكتب والمقالات الإخبارية، وتستخدم بشكل أساسيفي ويكيبيديا. هذه الأشياء الثلاثة هي المصادر الأساسية لمجموعات البيانات؛ وبالتالي فإن معظم مجموعات بيانات ونماذج المعالجة اللغوية الطبيعية موجودة باللغة العربية الفصحى.

لقد كان تحديًا مثيرًا!

الأدوات

لقد استخدمت مكتبة بايثون لمحولات Hugginface (Hugginface transformers) لبناء النموذج. فهو يوفر الآلاف من النماذج المدربة مسبقًا pre-trained models لمهام مختلفة، بمافي ذلك إنشاء النص.

لاختبار وتشغيل الكود الخاص بي، استخدمت Google Colab واستخدمت وحدات معالجة الرسومات GPUs الخاصة بهم.

النموذج الأساسي

النموذج الأساسي base model الذي استخدمته كان نموذجًا عربيًا GPT-2 من مركز نماذج Huggingface. تم تدريب النموذج باستخدام تفريغ ويكيبيديا العربية الكلاسيكية classic Arabic Wikipedia dump. يمكن استيراد النموذج وتهيئته كما يلي:

```
from transformers import AutoTokenizer
from transformers import Trainer,
TrainingArguments,AutoModelWithLMHead

tokenizer = AutoTokenizer.from_pretrained("akhooli/gpt2-small-
arabic")
model = AutoModelWithLMHead.from_pretrained("akhooli/gpt2-small-
arabic")
```

الآن، لدينا مرمز tokenizer لمعالجة النص العربي ونموذج لإنشاء نص جديد بناءً على نص السياق (التاريخ history). دعونا نسمي هذا النموذج النموذج العربي الكلاسيكي classic Arabic model.

لتشغيل النموذج وإنشاء النص، نحتاج إلى إنشاء مسار pipeline.

```
from transformers import pipeline

classic_ar_bot = pipeline('text-generation',model=model,
tokenizer=tokenizer,config={'max_length':60})
```

إذا قمنا بتشغيل المسار لإنشاء بعض التغريدات، فلن يبدو الأمر مثلي، إلا إذا قمت بنسخ تغريداتي من ويكيبيديا باللغة العربية الكلاسيكية.

على سبيل المثال، إذا حاولت استخدام النموذج لإكمال جملة تبدأ بـ "لا اعلم I don't know" باللغة العربية الفصحى:

```
classic_ar_bot('انا لا أعلم')
```

يقوم النموذج بإخراج ما يلي:

```
لا أعلم، حتى إذا كان من المقرر أن تكون هذه المسألة غير قابلة للنقاش،
إلا أن أحد الجانبين قال إنه لا يوجد سوى دليل كاف
```

وتكون النتيجة جملة ذات معنى باللغة العربية الفصحى، وتبدو كأنها جزء من قصة أو مقال. يترجم إلى: I don't know, even if this was to be a non-negotiable issue, but one side said "there was only enough evidence".

وإذا اختبرنا النموذج بجملة عربية مصرية، فسوف نتجاهل النص العربي المصري أو نعتبره اسمًا أو نصًا عشوائيًا. فمثلا لو اختبرنا النموذج بجملة باللغة العربية المصرية بنفس المعنى.

```
cls_ar_bot('انا مش عارف')
```

مخرجات النموذج هي:

```
انا مش عارف. كانت آخر بطولة دوري بطولة في تاريخها كانت في عام 1992
ضد أندية كاغاريغا وموراينغو، وفازت على مانشستر يونايتد في كأس العالم
للأندية
```

يقوم الوضع بإنشاء جملة عشوائية حول الرياضة لا علاقة لها بعبارة الإدخال "I don't know". وهذا يدل على أن النموذج لا "تفهم understand" اللهجة المصرية.

ضبط النموذج باستخدام البيانات العربية المصرية

قررت تحسين النموذج باستخدام مجموعة بيانات نصية باللغة العربية المصرية.

مجموعة البيانات

مجموعة البيانات dataset المستخدمة هي مجموعة التغريدات المصرية العربية، والتي تتكون من 40.000 تغريدة معدة مسبقًا مكتوبة باللغة العربية المصرية. تحتوي التغريدات على تعليقات توضيحية (إيجابية positive وسلبية negative)، لكن هذا ليس في نطاق هذه المهمة.

Rania Kora; Ammar Mohammed, 2019, "Corpus on Arabic Egyptian tweets", https://doi.org/10.7910/DVN/LBXV9O, Harvard Dataverse, V1

الضبط الدقيق

أولاً، أقوم باستيراد التبعيات المطلوبة

```
import pandas as pd
import json
import re
from sklearn.model_selection import train_test_split
```

بعد ذلك، أقوم بتحويل التغريدات TSV إلى مجموعتي بيانات مخزنتين في ملفات نصية.

```
#save dataset as text file
def build_text_files(data, dest_path):
  f = open(dest_path, 'w')
  f.write(" ".join(data))

#load dataset into Pandas Dataframe
data_df = pd.read_csv('40000-Egyptian-tweets.tsv', sep='\t',
header=0)

#split dataset into training and validation
train, val =
train_test_split(data_df["review"].tolist(),test_size=0.20)

#create the textfiles
build_text_files(train,'train_dataset.txt')
build_text_files(val,'val_dataset.txt')
```

لقد قمت بتقسيم مجموعة البيانات إلى 80% تدريب training، و20% للتحقق validation، والآن أقوم بتحميلها في كائن مجموعة بيانات Huggingface.

```
from transformers import
TextDataset,DataCollatorForLanguageModeling

def load_dataset(train_path,test_path,tokenizer):
    train_dataset = TextDataset(
          tokenizer=tokenizer,
          file_path=train_path,
          block_size=128)
    val_dataset = TextDataset(
          tokenizer=tokenizer,
          file_path=val_path,
          block_size=128)

    data_collator = DataCollatorForLanguageModeling(
```

```
        tokenizer=tokenizer, mlm=False,
    )
    return train_dataset,val_dataset,data_collator

train_path = 'train_dataset.txt'
val_path = 'val_dataset.txt'
train_dataset, val_dataset,data_collator =
load_dataset(train_path,test_path,tokenizer)
```

والآن، أصبحت مجموعة البيانات جاهزة للاستخدامفي تحسين النموذج.

لقد قمت بتدريب النموذج باستخدام المعلمات التالية:

```
training_args = TrainingArguments(
    output_dir="/content/model_40k", #The output directory
    overwrite_output_dir=False, #overwrite the content of the
output directory
    num_train_epochs=7, # number of training epochs
    per_device_train_batch_size=16, # batch size for training
    per_device_eval_batch_size=32,  # batch size for evaluation
    eval_steps = 100, # Number of update steps between two
evaluations.
    logging_steps = 100,
    save_steps=800, # after # steps model is saved
    warmup_steps=100,# number of warmup steps for learning rate
scheduler
    prediction_loss_only=True,
    evaluation_strategy='steps',
    learning_rate=5e-5,
    weight_decay=0.01
    )


trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)

trainer.train()
```

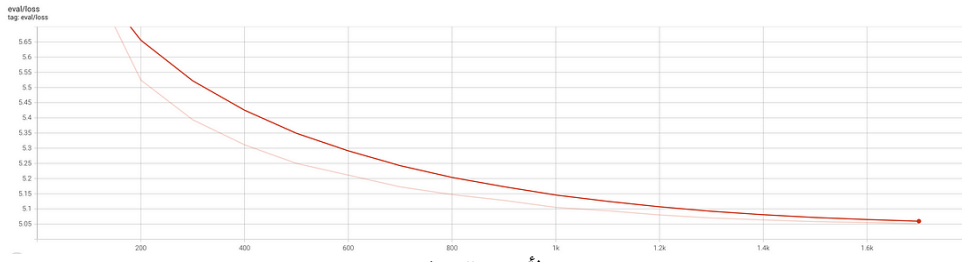
لقد قمت بتدريب النموذج لمدة 7 فترات، وبعد ذلك يبدأ خطأ التحقق من الصحة validation loss في الانخفاض ببطء (التشبع saturate).

```
***** Running training *****
  Num examples = 4100
  Num Epochs = 7
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 1799
```

[1799/1799 11:13, Epoch 7/7]

Step	Training Loss	Validation Loss
100	6.637100	5.874622
200	5.777900	5.524735
300	5.485500	5.393721
400	5.341800	5.311175
500	5.273400	5.249892
600	5.156900	5.211305
700	5.107400	5.172646
800	5.065500	5.147205
900	4.988500	5.128150
1000	4.964200	5.104940
1100	4.925600	5.093812
1200	4.897200	5.080115
1300	4.869400	5.070085
1400	4.826100	5.063729
1500	4.843500	5.058012
1600	4.811400	5.054990
1700	4.803400	5.051064

التدريب النموذجي (الضبط الدقيق)



خطأ/تقييم النموذج

الآن، دعونا نختبر النموذج الجديد المضبوط بدقة fine-tuned model:

```
model = AutoModelWithLMHead.from_pretrained("/content/model_40k")
egy_ar_bot = pipeline('text-generation',model=model,
tokenizer=tokenizer,config={'max_length':60})
```

اختبار النموذج مع نفس الجملة باللغة العربية المصرية التي تترجم إلى "I don't know":

```
egy_ar_bot('انا مش عارف')
```

الإخراج هو:

```
'انا مش عارف ايه اللى انا انا فى كل حاجة مخفي حد ده يا اخى. انا بعشق امتحانات اكثر امتحان'
```

والذي يترجم إلى:

"I do not know what I am in everything, I will hide this limit, my brother. I love exams more exams"

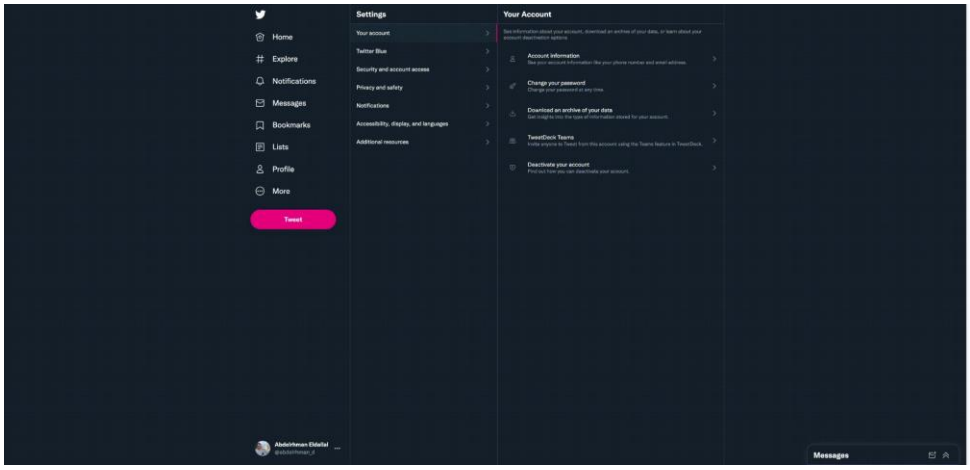
يستطيع النموذج الآن توليد جمل باللغة العربية المصرية، ولكن هناك شيء واحد مفقود. لا يزال النموذج لا يبدو مثلي **لأنني لا أحب الامتحانات**.

تخصيص البوت

تنزيل بيانات تويتر

قررت أن أقوم بضبط النموذج مرة أخرى باستخدام بيانات تويتر الخاصة بي لجعله يبدو أكثر شبهًا بي.

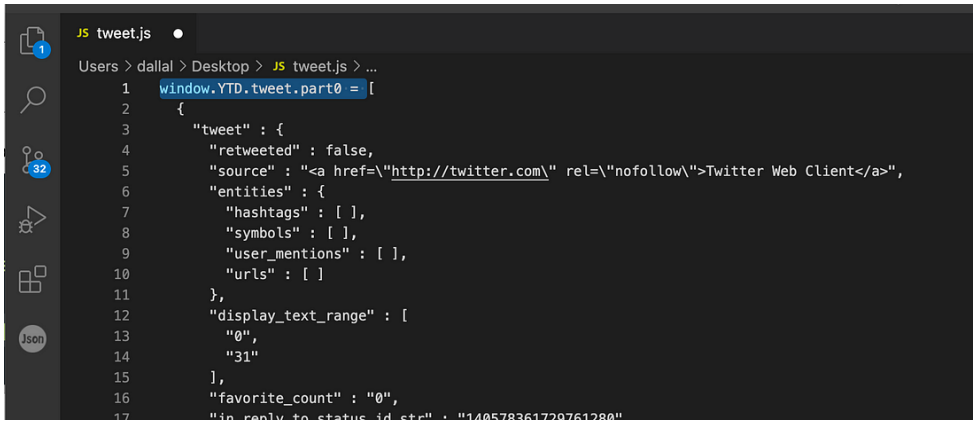
أولاً، طلبت الحصول على بياناتي، واضطررت إلى الانتظار لمدة يوم واحد حتى أتمكن من تنزيل البيانات من تويتر. هذا الخيار متاح لأي شخص، وليس هناك حاجة إلى حساب مطور.



طلب البيانات من إعدادات تويتر

بعد تنزيل البيانات، يُسمى الملف المطلوب للتغريدات "tweet.js"، وهو موجودفي مجلد "data".

بعد ذلك، أنسخ النص الموجود بعد القوس "["في السطر الأول إلى ملف JSON وأسميه "tweet.json"



ثم أقوم بتحميل البيانات.

```
f = open('tweet.json').read() #read JSON file
data = json.loads(f) #load file content into list of Python
dictonaries

texts = []
for tweet in data:
  text = "".join(re.findall(r'[\u0600-\u06FF]| ',
tweet['tweet']['full_text'],
                          re.UNICODE))  #only store Arabic text
  if len(text)>3: #igonre tweets with length less than 3 characters

    texts.append(' '.join(text.split()))
```

الآن، يمكنني تحويل البيانات إلى مجموعة بيانات لاستخدامها مع Huggingface باستخدام الكود التالي:

```
train, test = train_test_split(texts,test_size=0.1)

build_text_files(train,'train_dataset.txt')
build_text_files(test,'test_dataset.txt')
```

يمكن ضبط النموذج باستخدام مجموعة البيانات الجديدة على النحو التالي:

```
training_args = TrainingArguments(
    output_dir="/content/personal", #The output directory
    overwrite_output_dir=False, #overwrite the content of the
output directory
    max_steps=2000, # maximum number of steps
    per_device_train_batch_size=16, # batch size for training
    per_device_eval_batch_size=32,  # batch size for evaluation
```

```
    eval_steps = 100, # Number of update steps between two
evaluations.
    logging_steps = 100,
    save_steps=800, # after # steps model is saved
    warmup_steps=100,# number of warmup steps for learning rate
scheduler
    prediction_loss_only=True,
    evaluation_strategy='steps',
    learning_rate=5e-5,
    weight_decay=0.01
    )


trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
```

هذه المرة قمت بتعيين الحد الأقصى لعدد الخطوات على 2000، لأنني سأستخدم نقطة حفظ checkpoint النموذج المضبوطة بدقة بعد 1600 خطوة.

تم تدريب النموذج على فترتين two epochs:

```
***** Running training *****
  Num examples = 4100
  Num Epochs = 8
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 2000
  Continuing training from checkpoint, will skip to saved global_step
  Continuing training from epoch 6
  Continuing training from global step 1600
  Will skip the first 6 epochs then the first 58 batches in the first epoch. If th
Skipping the first batches: 100% 58/58 [00:00<00:00, 642.32it/s]
[2000/2000 02:28, Epoch 7/8]
```

Step	Training Loss	Validation Loss
1700	4.803500	5.048420
1800	4.805800	5.045307
1900	4.794200	5.044737
2000	4.773100	5.044055

```
***** Running Evaluation *****
  Num examples = 1016
  Batch size = 32
***** Running Evaluation *****
  Num examples = 1016
  Batch size = 32
***** Running Evaluation *****
  Num examples = 1016
  Batch size = 32
***** Running Evaluation *****
  Num examples = 1016
  Batch size = 32
```

تدريب النموذج

الآن، يمكن للنموذج إنشاء الجملة التالية:

"انا مش عارف ايه . أنا من اللي بجد من الهم لما حد يتاكدك يا عم. ان شاء الله هو خير له".

تحتوي الجملة على عبارات أستخدمها عادةً. العبارات منطقية، لكن الجملة الكاملة تبدو كما لو أنها مأخوذة من سياقها.

ومع ذلك، فإن النموذج يبدو مثلي!

الاستنتاج

لقد استغرق الأمر بضع ساعات للاحتفال بالذكرى السنوية لاشتراكي على تويتر من خلال إنشاء بوت مخصص لإنشاء النصوص. إن مفتاح تدريب مثل هذه النماذج هو توفر مجموعات البيانات، حيث أن النماذج المدربة مسبقًا جعلت عملية إنشاء مثل هذه الأنظمة أسهل بكثير.

المصدر:

https://medium.com/@abdelrhman.d/how-i-created-my-own-twitter-bot-with-deep-learning-e9c7bf814ae8

13) التعلم العميق والأرقام العربية المكتوبة بخط اليد Deep Learning & Handwritten Arabic Digits

غالبًا ما يكون "hello world" للتعلم العميق هو مجموعة بيانات الأرقام المكتوبة بخط اليد MNIST (MNIST handwritten number dataset)، وأردت تطبيق نفس التقنيات على تطبيق أكثر إثارة للاهتمام: مجموعة بيانات الحروف العربية المكتوبة بخط اليد Arabic Handwritten Characters Dataset (AHCD)، وهي مجموعة بيانات طورتها الجامعة الأمريكيةفي القاهرة.

في هذا المثال، أستخدم مكتبة fast.ai لتدريب شبكة عصبية تلافيفية convolutional neural net (CNN) لتصنيف AHCD بشكل صحيح بدقة 99+%. إليك الطريقة:

أولاً، قم باستيراد المكتبات التي نحتاجها وقم بضبط وحدة معالجة الرسومات GPU الخاصة بنا لاستخدام cuda:

```
%reload_ext autoreload
%autoreload 2
%matplotlib inlinefrom fastai.vision import *
from fastai.metrics import error_rate
import csv
import numpy as np
import PIL
import pandas as pddefaults.device = torch.device('cuda')
```

كما هو الحال مع العديد من مسارات عمل علم البيانات، تعد المعالجة المسبقة data pre-processing للبيانات العنصر الأكثر أهمية. فيما يلي الخطوات اللازمة لتجهيز البيانات لشبكتنا العصبية التلافيفية:

1) الاستيراد من ملف CSV

مثل إصدار MNIST بالأبجدية اللاتينية، يتم تقديم AHCD كملف CSV مكون من 784 عمودًا حيث يحتوي كل صف على صورة واحدة مقاس 28 × 28 تم تسويتهافي صف واحد من القيم الرقمية.

المهمة الأولى هي تحميل هذافي الذاكرة، وبما أن مجموعة البيانات تتكون من 60 ألف صف لتسريع العملية، فقد قمت بتعيين حد تعسفي لمجموعة التدريب 4k. لقد قمنا باستيراد Pandas كـ pd، لذلك يستخدم هذا دالة read_csv من Pandas المضمنة:

```
trainrows = 4000
train = pd.read_csv('csvtrain.csv', nrows=trainrows)
```

2) التحويل إلى بنية بيانات ثلاثية الأبعاد لمعالجة الصور

لدينا البياناتفي الذاكرة، ولكن كل صورة مطلوبة لا تزال مسطحة (1 طول × 784 عرض) ونريدها أن تكون مربعة ومتعددة الأبعاد حتى نتمكن من تحويلها إلى صورة RGB باستخدام matplotlib. لماذا RGB؟ سنستخدم نموذج Restnet34 المُدرب مسبقًا والذي تم تطويره على صور RGB.

تأخذ هذه الدالة البسيطة إطار بيانات Pandastrain الخاص بنا وتستخرج صفًا واحدًا (يتم تمريره كمتغير)، وتعيد تشكيل هذا الصف إلى معمارية مربعة، وتسوية الأرقامفي النطاق [0،1]، وتضيف بعدين إضافيين لجميع الأصفار، وتستخدم مكتبة matplotlib.plot لحفظ الصورة بتنسيق pngفي المجلد path/digits/ الخاص بنا.

ملاحظة:في النهاية سأضيف منطقًا لتمرير المجلد كمتغير.في الوقت الحالي، تم برمجته بشكل ثابت.

```
def pdMakePlot(row):
pixels = np.array(train.iloc[[row]], dtype='uint8')
pixels = pixels.reshape((28, 28)).T
pixels = np.true_divide(pixels, 255)
dim2 = np.zeros((28,28))
dim3 = np.zeros((28,28))
pix = np.stack((pixels, dim2,dim3), axis=2)
row += 1
filename = "digits/%s.png" % row
plt.imsave(filename, pix)
plt.close('all')
return
```

3) إعداد إطار بيانات مصدر الحقيقة الخاص بنا

نحن نستخدم طريقة fast.aiImageDataBunch.from_df لاستيعاب بيانات الصورة لهذه الشبكة العصبية التلافيفية، لذلك نحتاج إلى إطار بيانات Pandas يحتوي على أسماء ملفات التدريب والتسميات الصالحة valid labels.

```
#import training labels into numpy arraycsv = np.genfromtxt
('csvtrainlabel.csv', delimiter=",")
csv = csv[0:trainrows]
csv = csv.astype('int32')
csv = np.add(csv,1)
csv[csv == 10] = 0#np array that we'll make into the filenames
#from 1 to trainrows
trainrange = trainrows +1
files = np.arange(1,trainrange)
files = files.astype(str)#convert to filenames
i = 0;
j = 1;
for file in files:
    files[i] = "%s.png" % j
    i += 1
    j += 1
```

```
    if i >= trainrange: break#combine two arrays into dataframe and
add header
df = pd.DataFrame({'name':files, 'label':csv})
df.head()
```

Out[16]:

	name	label
0	1.png	1
1	2.png	2
2	3.png	3
3	4.png	4
4	5.png	5

إطار البيانات الخاص بنا

مرة أخرى، سأعود إلى قليل من عملية الاستخراج والتحويل والتحميل ETL.

4) معالجة وحفظ الصور التدريبية لدينا

وبهذا، يمكننا استخدام الدالة pdMakePlot() التي حددناها سابقًا لمعالجة صور التدريب. يتم أيضًا تحديد عدد الصور التي تتم معالجتها بواسطة متغير Trainrange الذي قمنا بتعيينه مسبقًا.

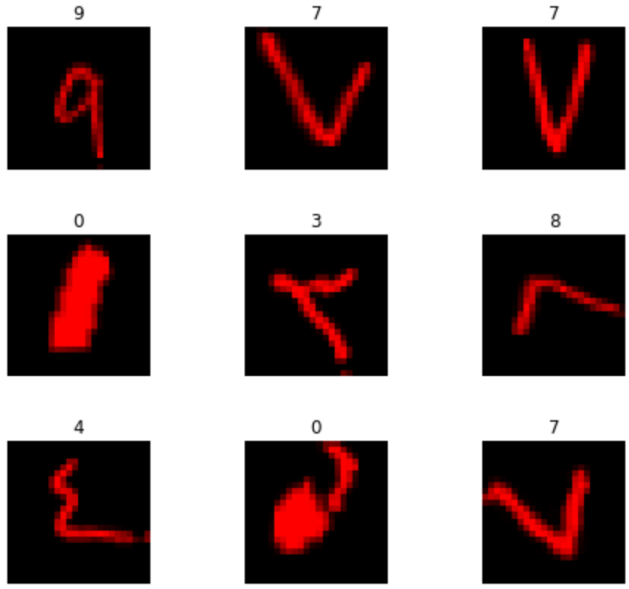
```
i = 0
max = trainrange-1
for x in range(i,max):
pdMakePlot(i)
i += 1
```

نحن الآن جاهزون للتعلم العميق! إنها مجرد بضعة أسطر من التعليمات البرمجية:

```
#define our transforms
tfms = get_transforms(do_flip=False)#define our DataBunch
data = ImageDataBunch.from_df(path=path, df = df, ds_tfms=tfms,
size=24)#define our learner
learn = create_cnn(data, models.resnet34, metrics=accuracy)
```

قبل أن نتدرب، يمكننا إلقاء نظرة على مجموعة صغيرة من DataBunch للتأكد من أننا قمنا بمعالجة كل شيء بشكل صحيح:

```
data.show_batch(rows=3, figsize=(7,6))
```



9 أحرف وتسميات مكتوبة بخط اليد

الامور جيدة! يمكننا أيضًا تشغيل learn.model لإلقاء نظرة تفصيلية على معمارية المتعلم learner architecture. إذا كنت مهتما، فهو متاح. على أية حال، دعونا ندرب!

التدريب الاولي

```
learn.fit_one_cycle(4)
```

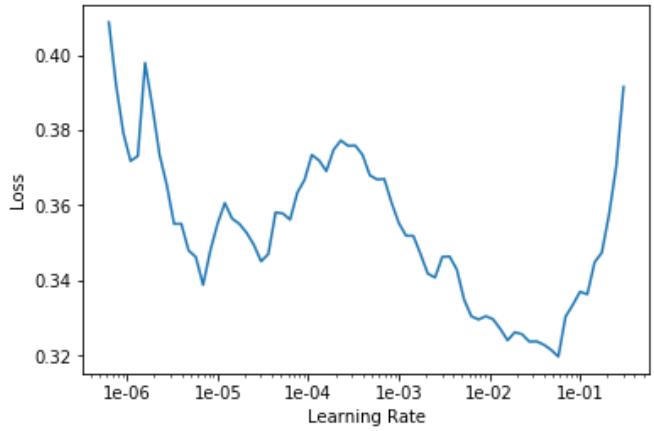
Total time: 00:16

epoch	train_loss	valid_loss	accuracy
1	1.461245	0.665243	0.802500
2	0.790824	0.228773	0.931250
3	0.502614	0.166105	0.961250
4	0.387829	0.167723	0.955000

نقل التعلم من resnet34 بدقة 95% خلال 16 ثانية

أعتقد أننا يمكن أن نفعل ما هو أفضل. دعونا نجد أفضل معدل للتعلم learning rate ونتدرب مرة أخرى.

```
learn.lr_find()LR Finder is complete, type
{learner_name}.recorder.plot() to see the
graph.learn.recorder.plot()
```



معدل التعلم ضد الخطأ

نرى أن معدل التعلم الأمثل مرتفع حقًا. دعونا نحاول الحصول على معدل تعلم أقل قليلاً مما يقلل الخسارة (الخطأ)، على سبيل المثال 05.؟ ثم سنقوم بإلغاء تجميد unfreeze بعض طبقات CNN وإعادة التدريب.

```
learn.unfreeze()
learn.fit_one_cycle(3, max_lr=slice(.006, .004))
```

Total time: 00:15

epoch	train_loss	valid_loss	accuracy
1	0.287015	0.307796	0.928750
2	0.221842	0.030812	0.991250
3	0.124907	0.009651	0.996250

وبعد خمسة عشر ثانية، أصبح لدينا نموذج دقيق بنسبة 99.6% مقابل المجموعة الفرعية من بيانات التدريب التي وضعناها جانبًا للتحقق من صحتها.

استخدام النموذج

والآن بعد أن أصبح لدينا نموذج، فلنستخدمه! بعد استخدام الدالة أعلاه لقراءة بعض بيانات الاختبار من اختبار ملف CSV:

```
img = open_image('/path/3.png')pred_class,pred_idx,outputs =
learn.predict(img)
pred_classCategory 4  <--- that is correct
```

الآن بعد أن أصبح لدينا نموذج عملي ودقيق، أود تحديث كود خط الأنابيب لجعله أكثر سهولة. أود أيضًا تشغيل النموذج مقابل المجموعة الكاملة من بيانات الاختبار لمعرفة كيفية مقارنته بأحدث ما توصلت إليه التكنولوجيا. المزيد قادم!

المصدر:

https://medium.com/towards-data-science/deep-learning-handwritten-arabic-digits-5c7abc3c0580

14) تصنيف الصور للحرف العربي المكتوب بخط اليد Image classification for Arabic handwritten character

الخلاصة

جرت محاولة للتعرف على الأحرف المكتوبة بخط اليد للأحرف العربية. تتكون مجموعة بيانات التدريب من 13440 صورة للأحرف و28 فئة. ويتم الحصول على تقنية استخراج الميزة feature extraction عن طريق تسوية normalizing قيم البكسل. ستتراوح قيم البكسل من 0 إلى 255 وهو ما يمثل شدة كل بكسل في الصورة ويتم تطبيعها لتمثيل القيم بين 0 ـ 1. يتم استخدام الشبكة العصبية التلافيفية Convolutional neural network كمصنف.

التحليل

في هذه المقالة، سأقوم بتجربة معلمات متعددة للعثور على الأفضل.

- أولاً لنقرأ البيانات:

```
# Load the training data
x_train = pd.read_csv('Arabic Handwritten Characters
Recognition/csvTrainImages 13440x1024.csv', header = None)
# Load training labels
x_label = pd.read_csv('Arabic Handwritten Characters
Recognition/csvTrainLabel 13440x1.csv', names=['count'])
# Load test data
y_test = pd.read_csv('Arabic Handwritten Characters
Recognition/csvTestImages 3360x1024.csv', header = None)
# Load test labels
y_label = pd.read_csv('Arabic Handwritten Characters
Recognition/csvTestLabel 3360x1.csv', header = None)
```

- الآن نحن بحاجة للتأكد من أن البيانات نظيفة:

3. Check for erroneous values

3.1. Check for missing values

The result bellow shows that there are no missing values

```
null_columns=x_train.columns[x_train.isnull().any()]
x_train[null_columns].isnull().sum()
```

```
Series([], dtype: float64)
```

3.2. Check maximum value

Since we are workin with pixel values, the maximum value should not exceed 255.

```
x_train.values.max()
```

```
255
```

3.3. Check for negative values

```
(x_train.values < 0).any(), (y_test.values < 0).any()
```

```
(False, False)
```

Since we are working with pixel values, there should be no negative values. The result shows there are no negative values.

- الخطوة التالية، دعونا نقسم بيانات التدريب لدينا إلى تدريب training والتحقق من الصحة validation.

```
x_train, z_val, x_label, z_label = train_test_split(x_train,
x_label, test_size=0.20, random_state=42)
```

```
print(x_train.shape, x_label.shape, y_test.shape, y_label.shape, z_val.shape, z_label.shape)
```

```
10752, 1024) (10752, 1) (3360, 1024) (3360, 1) (2688, 1024) (2688, 1)
```

طباعة عدد الصفوف والأعمدة

- نحن الآن بحاجة إلى أخذ القيم من البيانات ثم تحويلها إلى عدد عشري حتى نتمكن من تسويتها بين 0ـ1 دون فقدان المعلومات.

```
x_train = x_train.values.astype('float32')
x_label = x_label.values.astype('int32')-1 #Arabic letters are
28(index starts from 0-27)
y_test = y_test.values.astype('float32')
y_label = y_label.values.astype('int32')-1
z_val = z_val.values.astype('float32')
z_label = z_label.values.astype('int32')-1
```

- نحتاج إلى إعادة تشكيل البيانات من [# images, # features(32X32)] إلى [# images, # pixels, # pixels]، للتأكد من أن البيانات المدخلة إلى النموذج بالشكل الصحيح.

```
x_train = x_train.reshape(-1, 32, 32)
y_test = y_test.reshape(-1, 32, 32)
z_val = z_val.reshape(-1, 32, 32)
x_train.shape, y_test.shape, z_val.shape
```

```
x_train = x_train.reshape(-1, 32, 32)
y_test = y_test.reshape(-1, 32, 32)
z_val = z_val.reshape(-1, 32, 32)
x_train.shape, y_test.shape, z_val.shape
```

```
((10752, 32, 32), (3360, 32, 32), (2688, 32, 32))
```

- تحتوي مجموعات البيانات لدينا على قيمةفي كل بكسل تتراوح بين 0ـ255، لذا نقوم الآن بقياسها بين 0ـ1. لتسوية قيم البكسل، قم بتقسيمها على 255 (الحد الأقصى لقيمة البكسل).

```
x_train = x_train / 255
y_test = y_test / 255
z_val = z_val / 255
```

- تم تصميم طبقات Convolution2D للعمل مع 4 أبعاد. لذلك قم بتغيير أبعاد كل صورة إلى (دفعة batch، صفوف rows، أعمدة columns، قنوات channels). تشير القنوات إلى ما إذا كانت الصورة ذات تدرج رمادي أو ملون.في هذه الحالة، تكون الصور ذات تدرج رمادي، لذا يتم إعطاء 1 للقنوات.

```
x_train = x_train.reshape(-1, 32, 32,1)
y_test = y_test.reshape(-1, 32, 32,1)
z_val = z_val.reshape(-1, 32, 32,1)
(x_train.shape[1:], y_test.shape[1:], z_val.shape[1:])
```

- الآن نحتاج إلى استخدام الترميز واحد ساخن One Hot Encoding لتحويل عدد الفئات (28) من عدد صحيح إلى ثنائي، حيث سيتم اختيار واحدة منهافي المرة الواحدة (بقيمة 1 والباقي اصفار).

```
x_label = to_categorical(x_label, num_classes=28)
y_label = to_categorical(y_label, num_classes=28)
z_label = to_categorical(z_label, num_classes=28)
```

العثور على أفضل نموذج

- لنبدأ بإيجاد أفضل قيمة للتسرب dropout. سيمنع التسرب شبكتنا من الضبط الزائد overfitting، لذا فهو يساعد شبكتنا على التعميم generalize بشكل أفضل.

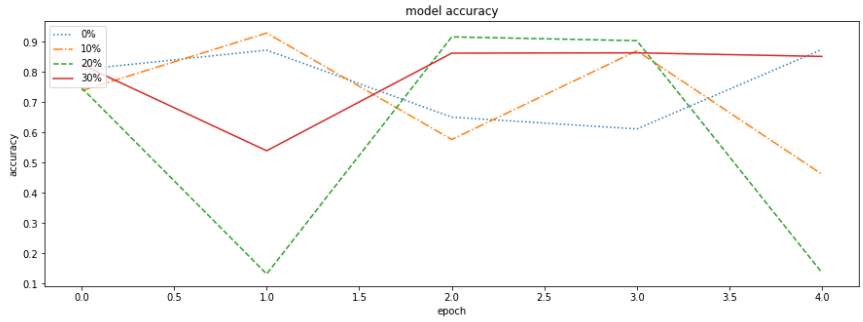
```
# CNN to find the best dropout
nets = 4
model = [0] *nets
input_shape = (32, 32, 1)
history = [0] * netsfor j in range(nets):

    model[j] = Sequential()
    model[j].add(Conv2D(16, (3,3), padding='same',
input_shape=input_shape,
                        kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
    model[j].add(Dropout(rate=j*0.1))model[j].add(Conv2D(32, (3,3),
padding='same', input_shape=input_shape,
                        kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
    model[j].add(Dropout(rate=j*0.1))

    model[j].add(Conv2D(64, (3,3), padding='same',
input_shape=input_shape,
                        kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
    model[j].add(Dropout(rate=j*0.1))model[j].add(Conv2D(64, (3,3),
padding='same', input_shape=input_shape,
                        kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
    model[j].add(Dropout(rate=j*0.1))

    model[j].add(Flatten())
```

```
    model[j].add(Dense(128, activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(Dropout(rate=j*0.1))model[j].add(Dense(28,
activation='softmax'))
    model[j].compile(optimizer="adam",
loss="categorical_crossentropy", metrics=["accuracy"])
```

- الآن دعونا نلائم fit الاختبار.

```
names = ["0%","10%","20%","30%"]
for j in range(nets):
history[j] = model[j].fit(x_train,x_label, batch_size=32, epochs =
5,
validation_data = (z_val,z_label), verbose=0)
print("Dropout {0}: Epochs={1:d}, Train accuracy={2:.5f},
Validation accuracy={3:.5f}".format(
names[j],5,max(history[j].history['acc']),max(history[j].history['v
al_acc']) ))
```

```
styles=[':','-.','--','-',':','-.','--','-',':','-.','--','-']
plt.figure(figsize=(15,5))
for i in range(nets):
    plt.plot(history[i].history['val_acc'],linestyle=styles[i])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(names, loc='upper left')
plt.show()
```



بناءً على النتائج المذكورة أعلاه، قررت اختيار 30% Dropout للنموذج.

- يتيح لك العثور على أفضل القيم لتعيين الفلاترفي CNN.

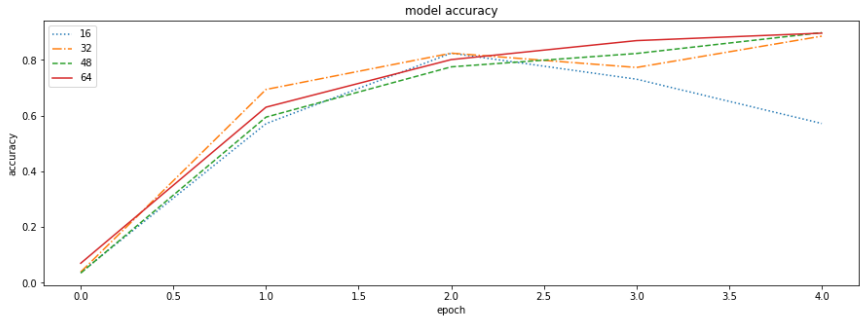
```
# CNN to find the best filter mapping
nets = 4
model = [0] *nets
input_shape = (32, 32, 1)
history = [0] * netsmodel = [0] *netsfor j in range(nets):

    model[j] = Sequential()
    model[j].add(Conv2D((j*16)+16, (3,3), padding='same',
input_shape=input_shape,
                        kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
```

```
    model[j].add(Dropout(rate=0.3))model[j].add(Conv2D((j*32)+32,
(3,3), padding='same', input_shape=input_shape,
                         kernel_initializer='uniform',
activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(MaxPooling2D(pool_size=2))
    model[j].add(Dropout(rate=0.3))

    model[j].add(Flatten())
    model[j].add(Dense(128, activation='relu'))
    model[j].add(BatchNormalization())
    model[j].add(Dropout(rate=0.3))model[j].add(Dense(28,
activation='softmax'))
    model[j].compile(optimizer="adam",
loss="categorical_crossentropy", metrics=["accuracy"])
```

- الآن دعونا نلائم الاختبار.

```
names = ["16","32","48","64"]
for j in range(nets):
history[j] = model[j].fit(x_train,x_label, batch_size=32, epochs =
5,
validation_data = (z_val,z_label), verbose=0)
print("CNN {0}: Epochs={1:d}, Train accuracy={2:.5f}, Validation
accuracy={3:.5f}".format(
names[j],5,max(history[j].history['acc']),max(history[j].history['v
al_acc']) ))
```

```
styles=[':','-.','--','-',':','-.','--','-',':','-.','--','-']
plt.figure(figsize=(15,5))
for i in range(nets):
    plt.plot(history[i].history['val_acc'],linestyle=styles[i])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(names, loc='upper left')
plt.show()
```



تطبيق الميزات لـ CNN

- 32-16
- 64-32

- 96-48
- 160-64

من النتائج أعلاه، يبدو أن 32ـ64، 48ـ96 و64ـ160 هي أفضل القيم لتعيين الفلاتر. لتقليل تكلفة الحساب سأختار 32ـ64.

- الآن دعونا ننشئ طريقة لتجربة معلمات مختلفة.

```
def create_model(optimizer='Adam', kernel_initializer='uniform',
activation='relu'):model = Sequential()model.add(Conv2D(16, (3,3),
padding='same', input_shape=input_shape,
                     kernel_initializer=kernel_initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=2))
    model.add(Dropout(rate=0.3))model.add(Conv2D(32, (3,3),
padding='same',
                kernel_initializer=kernel_initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=2))
    model.add(Dropout(rate=0.3))model.add(Conv2D(64, (3,3),
padding='same',
                kernel_initializer=kernel_initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=2))
    model.add(Dropout(rate=0.3))model.add(Conv2D(128, (3,3),
padding='same',
                kernel_initializer=kernel_initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=2))
    model.add(Dropout(rate=0.3))model.add(Conv2D(256, (3,3),
padding='same',
                kernel_initializer=kernel_initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=2))
    model.add(Dropout(rate=0.3))model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(rate=0.3))# fully connected Final layer
    model.add(Dense(28,
activation='softmax'))model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

- تجربة معلمات مختلفة (المحسنات Optimizers وKernel_initializers ودوال التنشيط activation functions) للعثور على أفضل قيم المعلمات.

```
optimizer = ['RMSprop', 'Adam','Adagrad']
kernel_initializer = ['normal', 'uniform']
activation = ['relu', 'linear']for a,b,c in [(x,y,z) for x in
optimizer for z in activation for y in kernel_initializer]:
    params = {'optimizer' : a , 'kernel_initializer' : b ,
'activation' : c}
    print(params)
    curr_model = create_model(a, b, c)
    curr_model.fit(x_train, x_label,
                   validation_data=((z_val,z_label)),
                   epochs=5, batch_size=32, shuffle=True,
verbose=1)
    print("-----------------------------------------------------------
---------------------------")
```

- بعد تشغيل النموذج عدة مرات، قررت استخدام {'optimizer': 'Adam', 'kernel_initializer': 'uniform', 'activation': 'relu'}.

نموذجي النهائي مع أفضل المعلمات:

```
input_shape = (32, 32, 1)model = Sequential()
model.add(Conv2D(32, (3,3), padding='same',
input_shape=input_shape,
                     kernel_initializer='uniform',
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(rate=0.3))model.add(Conv2D(32, (3,3),
padding='same', input_shape=input_shape,
                     kernel_initializer='uniform',
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(rate=0.3))model.add(Conv2D(64, (3,3),
padding='same', input_shape=input_shape,
                     kernel_initializer='uniform',
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(rate=0.3))model.add(Conv2D(64, (3,3),
padding='same', input_shape=input_shape,
                     kernel_initializer='uniform',
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(rate=0.3))model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(rate=0.3))model.add(Dense(28,
activation='softmax'))
model.compile(optimizer="adam", loss="categorical_crossentropy",
```

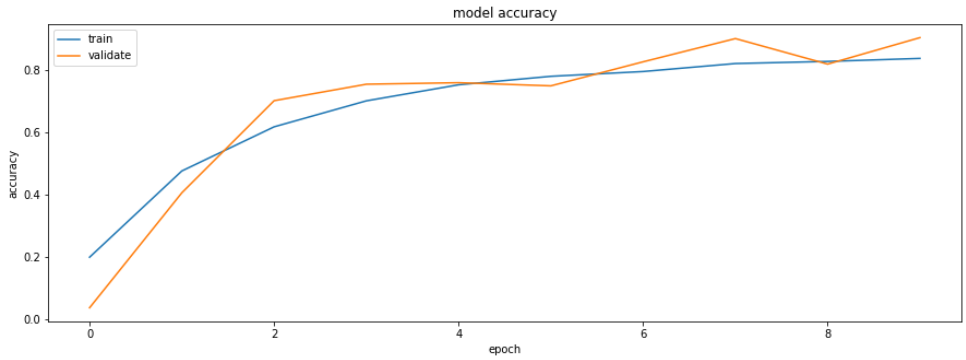
```
metrics=["accuracy"])
model.summary()
```

لقد حان الوقت لملائمة نموذجنا:

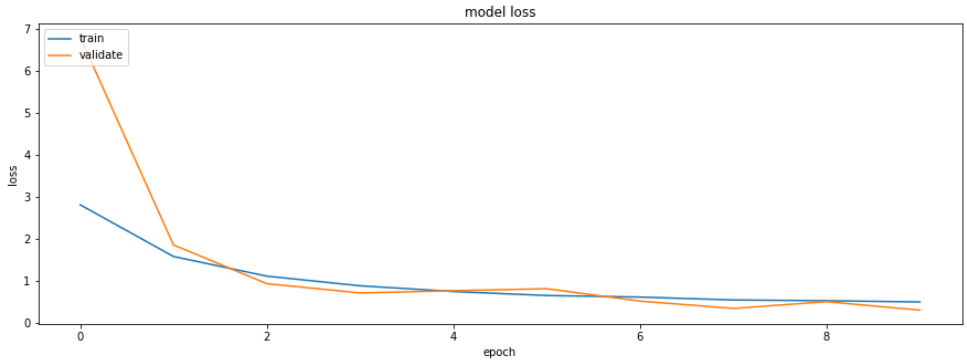
```
history = model.fit(x_train, x_label,
validation_data=(z_val,z_label),epochs=10, batch_size=32,
shuffle=True, verbose=1)
```

- الرسم Visualization

```
# Accuracy VS Epochs
# summarize history for acuuracy
print(history.history.keys())
plt.figure(figsize=(15,5))
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validate'], loc='upper left')
plt.show()
```




```
# Loss VS Epochs
# summarize history for loss
plt.figure(figsize=(15,5))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validate'], loc='upper left')
plt.show()
```



- الآن دعونا نحفظ النموذج لاستخدامه.

```
model.save('my_model.hdf5')
```

اختبار النموذج على مجموعة بيانات الاختبار:

```
evaluate = model.evaluate(y_test, y_label, verbose=1)
```

```
evaluate = model.evaluate(y_test, y_label, verbose=1)
3360/3360 [==============================] - 1s 394us/sample - loss: 0.2838 - acc: 0.9098
```

- لقد حصلنا على 90% من 10 فترات epochs، فلنزيد الفترات للحصول على دقة أفضل.

```
epochs = 25
from keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath='my_model.hdf5', verbose=1,
save_best_only=True)history = model.fit(x_train, x_label,
                    validation_data=(z_val, z_label),
                    epochs=epochs, batch_size=32, verbose=1,
callbacks=[checkpointer])

model.load_weights('my_model.hdf5')
```

- طباعة الدقة بعد 25 فترة.

```
metrics = model.evaluate(y_test, y_label, verbose=1)
print("Test Accuracy: {}".format(metrics[1]))
print("Test Loss: {}".format(metrics[0]))
```

```
3360/3360 [==============================] - 1s 393us/sample - loss: 0.1214 - acc: 0.9649
Test Accuracy: 0.9648809432983398
Test Loss: 0.12135044537551169
```

لنقم الآن بتقييم نموذجنا من خلال تقرير التصنيف classification report (الدقة Precision والاستدعاء recall ودرجة f1 (f1-score) والدعم support)

	precision	recall	f1-score	support
0	0.97	1.00	0.98	120
1	0.99	0.99	0.99	120
2	0.90	0.97	0.93	120
3	0.97	0.92	0.94	120
4	0.99	0.96	0.97	120
5	0.94	0.98	0.96	120
6	0.98	0.97	0.97	120
7	0.91	0.99	0.95	120
8	0.93	0.93	0.93	120
9	0.94	0.98	0.96	120
10	0.98	0.90	0.94	120
11	0.97	1.00	0.98	120
12	0.98	0.98	0.98	120
13	0.97	0.97	0.97	120
14	0.98	0.93	0.96	120
15	0.96	1.00	0.98	120
16	0.99	0.96	0.97	120
17	0.98	0.98	0.98	120
18	0.99	0.97	0.98	120
19	0.90	0.97	0.93	120
20	0.96	0.91	0.94	120
21	0.98	0.97	0.97	120
22	0.99	1.00	1.00	120
23	0.98	0.98	0.98	120
24	0.96	0.93	0.94	120
25	0.97	0.96	0.97	120
26	0.95	0.94	0.95	120
27	1.00	0.99	1.00	120
accuracy			0.96	3360
macro avg	0.97	0.96	0.96	3360
weighted avg	0.97	0.96	0.96	3360

الاستنتاج

كما نرى أعلاه، حصلنا على دقة جيدة جداً ويمكن للنموذج أن يتحسن على مدار المزيد من الفترات. إن تدريب CNN هو عملية عشوائية، وفي كل مرة تقوم فيها بإجراء التجربة تحصل على نتائج مختلفة. يعتمد ذلك على معلمات فائقة hyperparameters متعددة (عدد الطبقات number of layers، عدد خرائط الميزات number of feature maps في كل طبقة، التسربات dropouts، تسوية الدُفعات batch normalization، إلخ...). ولذلك، يجب عليك إجراء تجاربك عدة مرات قبل أن تختار نموذجك النهائي. يمكنك رؤية الكود هنا https://github.com/Hassan-AlHajri/Image-classification-for-Arabic-handwriting-character

المصدر:

https://medium.com/@hass.9964/image-classification-for-arabic-handwritten-character-64209a7aba9d

15) Keras Tuner مع MNIST العربية Keras Tuner with Arabic MNIST

استيراد التبعيات

```
import pandas as pd
import numpy as npimport tensorflow as tf
from tensorflow import kerasfrom tensorflow.keras.layers import
Dropout,BatchNormalization,Conv2D,MaxPooling2D,Dense,Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.datasets import cifar10
from tensorflow.keras import regularizers
from keras import callbacks
from sklearn.preprocessing import StandardScaler
from keras.models import Sequentialfrom kerastuner import
RandomSearch
from kerastuner.engine.hyperparameters import HyperParameters
from tensorflow.keras.optimizers import Adam
!pip install keras-tunerimport matplotlib.pyplot as plt
import seaborn as sns
```

تحميل مجموعة البيانات

يمكنك الحصول على مجموعة البيانات من هنا

تحتوي مجموعة البيانات على 59,999 تدريبًا (28 × 28) و9999 صورة اختبارية للأرقام العربية المكتوبة بخط اليد.

```
X_train=pd.read_csv("/kaggle/input/ahdd1/Arabic Handwritten Digits
Dataset CSV/csvTrainImages 60k x
784.csv")y_train=pd.read_csv("/kaggle/input/ahdd1/Arabic
Handwritten Digits Dataset CSV/csvTrainLabel 60k x
1.csv")X_test=pd.read_csv("/kaggle/input/ahdd1/Arabic Handwritten
Digits Dataset CSV/csvTestImages 10k x
784.csv").valuesy_test=pd.read_csv("/kaggle/input/ahdd1/Arabic
Handwritten Digits Dataset CSV/csvTestLabel 10k x 1.csv")
```

لمحة عن مجموعة البيانات:

```
X_train
```

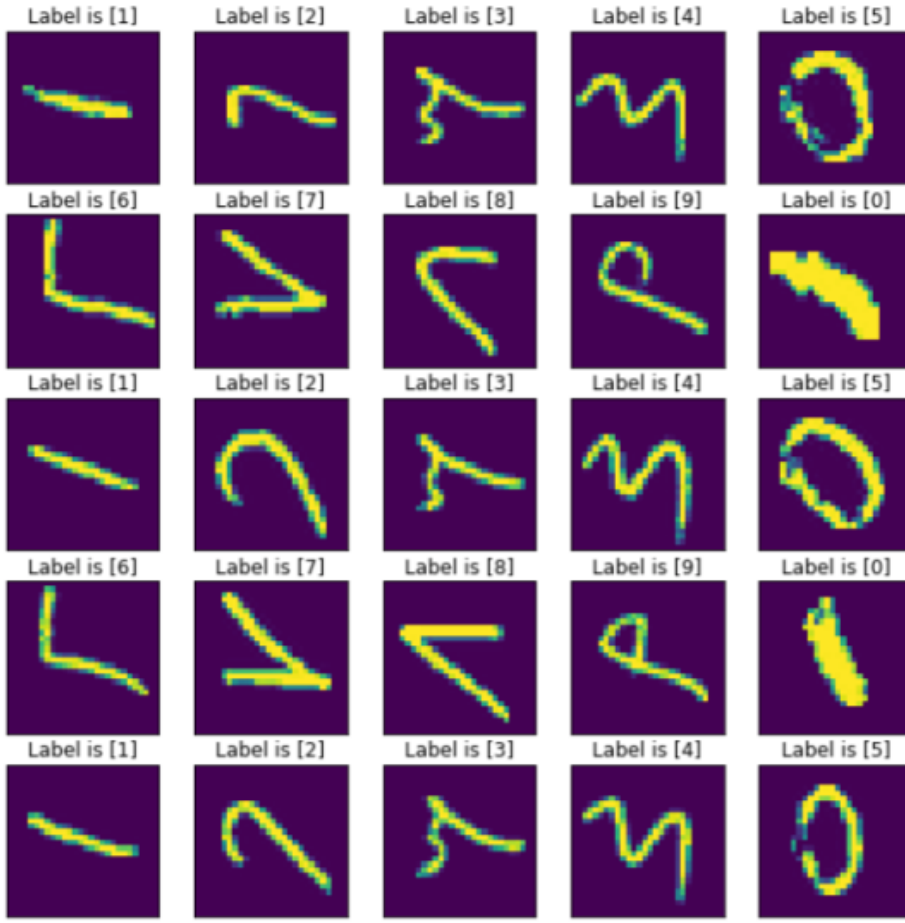
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.486	0.487	0.488	0.489	0.490	0.491	0.492	0.493	0.494	0.495
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59994	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59995	0	0	0	0	0	0	0	170	146	97	...	0	0	0	0	0	0	0	0	0	0
59996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

59999 rows × 784 columns

تحتوي كل صورة على 784 (28 × 28) بكسل، مما يؤدي إلى إنشاء 784 عمودًا في إطار البيانات.

رسم الأرقام العربية

```
X_train=X_train.values
y_train=y_train.values
labels = {0 : "0", 1: "1", 2: "2", 3: "3", 4: "4",
5: "5", 6: "6", 7: "7", 8: "8", 9: "9"}
plt.figure(figsize=(10,10))
for i in range(25):
plt.subplot(5,5,i+1)
plt.xticks([])
plt.yticks([])
plt.grid(False)
plt.imshow(X_train[i].reshape((28,28)))
plt.title('Label is {label}'.format(label=y_train[i]))
plt.show()
```



التحجيم وإعادة التشكيل

تتقارب CNN بشكل أسرع عندما يتم قياس القيم وإعادة تشكيلها لأن Conv2D يقبل القيم كمصفوفة رباعية الأبعاد.

```
X_train = X_train / 255
X_test = X_test / 255X_train = X_train.reshape(-1,28,28,1)
X_test = X_test.reshape(-1,28,28,1)
```

النمذجة

الجزء (أ)- CNN البسيطة

```
#Early stopping
early_stopping = callbacks.EarlyStopping(
    monitor='val_loss',
    min_delta=0.001, # minimium amount of change to count as an
improvement
    patience=5, # how many epochs to wait before stopping
    restore_best_weights=True,
)model=Sequential()
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape
=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2,
2),strides=(2,2)))model.add(Conv2D(32,kernel_size=(3,3),activation=
'relu'))
model.add(MaxPooling2D(pool_size=(2,
2),strides=(2,2)))model.add(Flatten())
model.add(Dense(16,activation='relu'))
model.add(Dense(10,activation='softmax'))%%time
model.compile(optimizer='adam',loss='sparse_categorical_crossentrop
y',metrics='accuracy')model.fit(X_train,y_train,validation_split=0.
2,epochs=20,batch_size=64,callbacks=[early_stopping])
```

```
2022-07-22 19:23:59.707790: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
750/750 [==============================] - 10s 4ms/step - loss: 0.2116 - accuracy: 0.9384 - val_loss: 0.0830 - val_accuracy: 0.9766
Epoch 2/20
750/750 [==============================] - 3s 4ms/step - loss: 0.0489 - accuracy: 0.9861 - val_loss: 0.0583 - val_accuracy: 0.9838
Epoch 3/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0317 - accuracy: 0.9910 - val_loss: 0.0361 - val_accuracy: 0.9895
Epoch 4/20
750/750 [==============================] - 3s 4ms/step - loss: 0.0257 - accuracy: 0.9923 - val_loss: 0.0458 - val_accuracy: 0.9867
Epoch 5/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0211 - accuracy: 0.9935 - val_loss: 0.0371 - val_accuracy: 0.9886
Epoch 6/20
750/750 [==============================] - 3s 4ms/step - loss: 0.0189 - accuracy: 0.9945 - val_loss: 0.0328 - val_accuracy: 0.9895
Epoch 7/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0164 - accuracy: 0.9950 - val_loss: 0.0374 - val_accuracy: 0.9893
Epoch 8/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0149 - accuracy: 0.9954 - val_loss: 0.0381 - val_accuracy: 0.9891
Epoch 9/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0121 - accuracy: 0.9960 - val_loss: 0.0369 - val_accuracy: 0.9880
Epoch 10/20
750/750 [==============================] - 2s 3ms/step - loss: 0.0107 - accuracy: 0.9965 - val_loss: 0.0363 - val_accuracy: 0.9908
Epoch 11/20
750/750 [==============================] - 3s 4ms/step - loss: 0.0099 - accuracy: 0.9968 - val_loss: 0.0415 - val_accuracy: 0.9892
CPU times: user 34.9 s, sys: 3.77 s, total: 38.7 s
Wall time: 36.3 s
```

تقييم النموذج

```
model.evaluate(X_test,y_test)Output:
313/313 [==============================] - 1s 2ms/step -
loss: 0.0382 - accuracy: 0.9880
```

الجزء (ب)-ضبط معلمات التعلم العميق باستخدام Keras Tuner

سنجد القيمة المثلى لعدد الفلاتر kernel size وحجم النواة learning rate ومعدل التعلم learning rate باستخدام Keras Tuner.

```
import keras_tuner as kt
hp = kt.HyperParameters()def build_model(hp):
  model = keras.Sequential([
    keras.layers.Conv2D(
        filters=hp.Int('conv_1_filter', min_value=32,
max_value=128, step=16),
        kernel_size=hp.Choice('conv_1_kernel', values = [3,5]),
        activation='relu',
        input_shape=(28,28,1)
    ),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(
        filters=hp.Int('conv_2_filter', min_value=32, max_value=64,
step=16),
        kernel_size=hp.Choice('conv_2_kernel', values = [3,5]),
        activation='relu'
    ),
    keras.layers.Dropout(0.2),

    keras.layers.Flatten(),
    keras.layers.Dense(
        units=hp.Int('dense_1_units', min_value=32, max_value=128,
step=16),
        activation='relu'
    ),
    keras.layers.Dense(10, activation='softmax')
  ])

model.compile(optimizer=tf.optimizers.Adam(hp.Choice('learning_rate
', values=[1e-2, 1e-3])),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

  return model
```

البحث العشوائي

```
tuner_search=RandomSearch(build_model,
                          objective='val_accuracy',

max_trials=5,directory='output',project_name="Arabic MNIST")%%time
tuner_search.search(X_train,y_train,epochs=3,validation_split=0.2)O
utputTrial 5 Complete [00h 00m 24s]
val_accuracy: 0.9894166588783264

Best val_accuracy So Far: 0.9911666512489319
Total elapsed time: 00h 02m 15s
CPU times: user 1min 52s, sys: 9.11 s, total: 2min 1s
Wall time: 2min 15s
```

ضبط النموذج باستخدام أفضل المعلمات

يمكننا تلقائيًا تحديد أفضل نتائج ضبط معلمات التعلم العميق لنموذج CNN الخاص بنا.

```
model=tuner_search.get_best_models(num_models=1)[0]
```

ملخص النموذج

```
model.summary()Output:Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 96)        960
_________________________________________________________________
dropout (Dropout)            (None, 26, 26, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 64)        153664
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 96)                2973792
_________________________________________________________________
dense_1 (Dense)              (None, 10)                970
=================================================================
Total params: 3,129,386
Trainable params: 3,129,386
Non-trainable params: 0
_________________________________
```

ملائمة النموذج

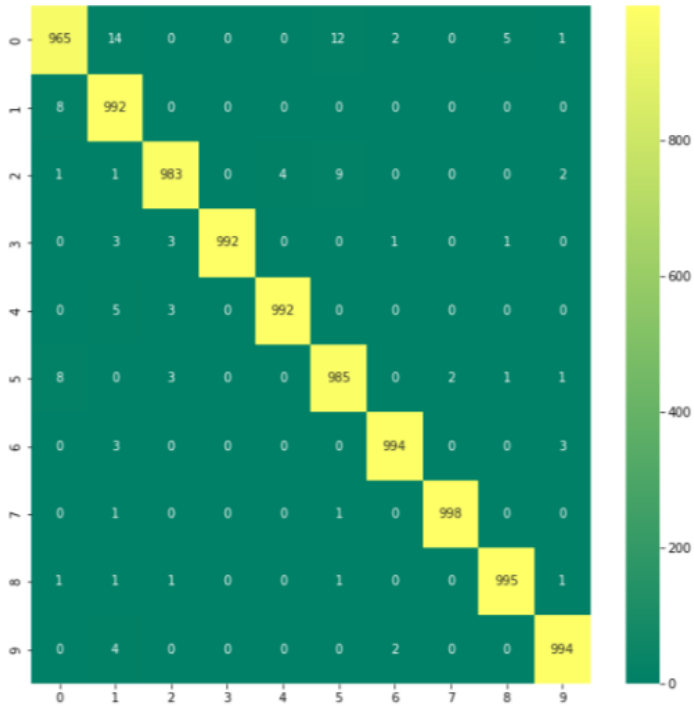
```
model.fit(X_train, y_train, epochs=10, validation_split=0.1,
initial_epoch=3)
```

تقييم النموذج

(أ) الدقة والخطأ

```
model.evaluate(X_test,y_test)Output:
313/313 [==============================] - 1s 2ms/step -
loss: 0.0867 - accuracy: 0.9891
```

(ب) مصفوفة الارتباك



يخطئ نموذجنافي تصنيف 0 إلى 1،5 و8.

تمرين

قم برسم الحالات التي حدث فيها خطأفي نموذجنا، وتحقق مما إذا كانت التسميات صحيحة.

المصدر:

https://medium.com/@ebrahimhaqbhatti516/keras-tuner-with-arabic-mnist-303a9c57c48a

Arabic & AI

Arabic Models Solved with Machine and Deep Learning

By: Dr. Alaa Taima