

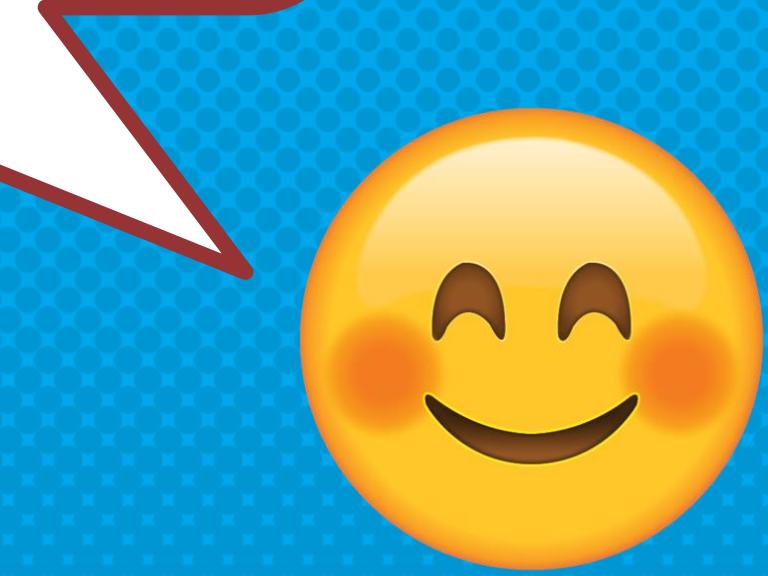
عنوان:

گزارش پروژه پایانی درس پردازش تصویر رقمه‌ی

ارائه دهنده‌گان:

فاطمه صالح زیا

نفیسه بختیاری



۶. مقایسه سه
مدل مختلف

۷. ترکیب سه
مدل مختلف

۸. ایجاد یک
کلاسی فایر
بساده

۹. ایجاد یک
شبکه RNN در
سطح کلمه

مراحل انجام کار

۱۰. ایجاد یک
شبکه
کانولوشن روی
دیتاست آماده

۱۱. استفاده از
مدل کانولوشن
قبلی روی دیتای
جديد

۱۲. استخراج داده
از توبیتر

این دیتاست شامل متن هایی است
که با توجه به احساسات درون آنها
برچسب خورده اند.

کد مربوطه:

```
import pandas as pd
from keras.utils.data_utils import get_file
import nb_utils

emotion_csv = get_file('text_emotion.csv',
                      'https://www.crowdflower.com/wp-content/' +
                      'uploads/2016/07/text_emotion.csv')
emotion_df = pd.read_csv(emotion_csv)

emotion_df.head()
```

tweet_id	sentiment	author	content
1956967341	empty	xshayzres @tiffanyhue	i know i was listenin to bad habi...
1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...
1956967696	sadness	oofunkyl	Funeral ceremony...gloomy friday...
1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
1956968416	neutral	xkilljoyx @dannycastillo	We want to trade with someone w...

ایجاد یک کلاسی فایر ساده

۱. دانلود دیتاست آماده crowdFlower

۲. استخراج تعداد هر احساس بنا به اطلاعات دیتاست

۳. استفاده از TFIDFVectorizer

۴. استفاده از LabelEncoder

۵. ایجاد یک کلاسی فایر بیزین روی دیتاست

ایجاد یک کلاسی فایر ساده

می توان دید که از هر حالت حسی موجود (برچسب)، چه تعداد در دیتاست وجود دارد.

کد مربوطه:

```
emotion_df['sentiment'].value_counts()
```

neutral	8638
worry	8459
happiness	5209
sadness	5165
love	3842
surprise	2187

۱. دانلود دیتاست آماده crowdFlower

۲. استخراج تعداد هر احساس بنا به اطلاعات دیتاست

۳. استفاده از TFIDFVectorizer

۴. استفاده از LabelEncoder

۵. ایجاد یک کلاسی فایر بیزین روی دیتاست

ایجاد یک کلاسی فایر ساده

برای وزن دهی به کلمات استفاده می شود.

منطق: تکرار بیشتر = بار معنایی کمتر

TF: تکرار یک کلمه در متن،

IDF: بر عکس تعداد تکرار در کل متون

۱. دانلود دیتاست آماده crowdFlower

۲. استخراج تعداد هر احساس بنا به اطلاعات دیتاست

۳. استفاده از TFIDFVectorizer

۴. استفاده از LabelEncoder

۵. ایجاد یک کلاسی فایر بیزین روی دیتاست

کد مربوطه:

```
tfidf_vec = TfidfVectorizer(max_features=VOCAB_SIZE)
```

```
linear_x = tfidf_vec.fit_transform(emotion_df['content'])
```

به هر برچسب یک عدد صحیح منحصر بفرد نسبت می دهد.

کد مربوطه:

```
label_encoder = LabelEncoder()  
linear_y = label_encoder.fit_transform(emotion_df[ 'sentiment' ])
```

۱. ایجاد یک کلاسی فایر ساده

۱. دانلود دیتاست آماده crowdFlower

۲. استخراج تعداد هر احساس بنا به اطلاعات دیتاست

۳. استفاده از TFIDFVectorizer

۴. استفاده از LabelEncoder

۵. ایجاد یک کلاسی فایر بیزین روی دیتاست

در این مرحله، کلاسی فایر بیزین، روی داده ها استفاده شده و نتیجه ارزیابی آن با دقت مشخص بدست می آید.

کد مربوطه:

```
bayes = MultinomialNB()
bayes.fit(linear_x, linear_y)
pred = bayes.predict(linear_x)
precision_score(pred, linear_y, average='micro')
```

0.28022727272727271

ایجاد یک کلاسی فایر ساده

۱. دانلود دیتاست آماده crowdFlower

۲. استخراج تعداد هر احساس بنا به اطلاعات دیتاست

۳. استفاده از TFIDFVectorizer

۴. استفاده از LabelEncoder

۵. ایجاد یک کلاسی فایر بیزین روی دیتاست

مقایسه پا ساید روش ها

ساعت
پایین تر

```
classifiers = {'sgd': SGDClassifier(loss='hinge'),  
               'svm': SVC(),  
               'random_forest': RandomForestClassifier()}  
  
for lbl, clf in classifiers.items():  
    clf.fit(X_train, y_train)  
    predictions = clf.predict(X_test)  
    print(lbl, precision_score(predictions, y_test, average='micro'))
```

- 0.2186363636

SVM



- 0.28393939

RandomForest



- 0.3254545455

sgd



دقت تا دری
بخت



بحث

هر فاکتور مستقل از سایرین بررسی می شود.

ساده و سریع هستند.

ویژگی بیزین

همیشه خوب عمل نمی کند.

مثال: تاثیر Not در حالت

احساسی جمله

قاعدۀ کلی: اگر مدل بیزین نتواند نتیجه خوبی روی داده بدست آورد، استفاده از مدل پیچیده تم کمکی نخواهد کرد.



۲. ارزیابی یک کلاسی فایل سعاده:

چطور می توان فهمید یک کلاسی فایل چه چیزی یادگرفته است؟



با استفاده از استقلال در بررسی کلمات، نظر مدل را راجع به هر کلمه به طور مستقل می پرسیم.

فرض: مجموعه متنه داریم که در هر کدام از آنها تنها یک کلمه مجزا آمده، مدل یک ماتریس مربعی خواهد بود که روی قطر اصلی آن یک و بقیه صفر هستند. حال می توان احتمال انتساب برچسب برای هر کلمه را تخمین زد.

```
d = eye(len(tfidf_vec.vocabulary_))  
word_pred = bayes.predict_proba(d)
```

```
by_cls = defaultdict(Counter)
for word_idx, pred in enumerate(word_pred):
    for class_idx, score in enumerate(pred):
        cls = label_encoder.classes_[class_idx]
        by_cls[cls][inverse_vocab[word_idx]] = score
```

کد ارائه نتایج

```
for k in by_cls:
    words = [x[0] for x in by_cls[k].most_common(5)]
    print(k, ':', ' '.join(words))
```

نتایج

```

happiness : excited woohoo excellent yay wars
hate : hate hates suck fucking zomberellamcfox
boredom : squeaking ouuut cleanin soooooo candyland3
enthusiasm : lena_distractia foolproofdiva attending krisswouldhowse tatt
fun : xbox bamboozle sanctuary oldies toodaayy
love : love mothers mommies moms loved
surprise : surprise wow surprised wtf surprisingly
empty : makinitrite conversating less_than_3 shakeyourjunk kimbermuffin
anger : confuzzled fridaaaaayyyyy aaaaaaaaaaaa transtelcom filthy
worry : worried poor throat hurts sick
relief : finally relax mastered relief inspiration
sadness : sad sadly cry cried miss
neutral : www painting souljaboytellem link frenchieb
```

روال کار

- متیاز کلمه برای هر کلاس برسی می‌شود.
- ذخیره اطلاعات در counter
- تعیین کلمات که بیشترین تاثیر را دارند.

بحث

نتایج قابل انتظار بود. مثل sad در کلاس sadness کلمات عجیب مثل foolproofdrive به معنای مشتاق، می‌تواند بسته به نیاز فیلتر شود.

۳. استفاده از شبکه کانولوشن برای تحلیل احساسات

راه حل: استفاده از شبکه کانولوشن

مسئله: استفاده از یک شبکه عمیق برای درک احساس یک متن

یک پنجره روی متن حرکت کرده، یک دنباله طولانی از آیتم‌ها را به یک دنباله کوتاه‌تر از ویژگی‌ها تبدیل کند.

ایده

نکته: وزن‌ها در همه گام‌ها یکسان است ← نیاز نیست یک چیز را چند بار یاد بگیریم

کد مربوط به کانولوشن

```
char_input = Input(shape=(max_sequence_len, num_chars), name='input')

conv_1x = Conv1D(128, 6, activation='relu', padding='valid')(char_input)
max_pool_1x = MaxPooling1D(6)(conv_1x)
conv_2x = Conv1D(256, 6, activation='relu', padding='valid')(max_pool_1x)
max_pool_2x = MaxPooling1D(6)(conv_2x)

flatten = Flatten()(max_pool_2x)
dense = Dense(128, activation='relu')(flatten)
preds = Dense(num_labels, activation='softmax')(dense)

model = Model(char_input, preds)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])
```

دولایه کانولوشن



دولایه max_pool



دولایه تمام متصل



مدل

توضیحات مربوط به کانولوشن

- در اینجا از یک پنجره به طول ۶ استفاده شده است یعنی هر لحظه ۶ کاراکتر.

- برای یک توییت به طول ۱۲۰، ۱۲۵ گام باید پنجره را حرکت دهیم.

نکته: در ۱۲۰ نورون، وزن ها یکسان است در نتیجه همه یک چیز یاد گرفته اند.

در maxpool، سایز ۱/۶ خواهد شد.

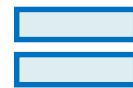
دولایه کانولوشن



دو لایه



دو لایه تمام متصل



مدل

نکات:

پیش از اجرای مدل، باید داده به بردار تبدیل شود،
اینجا از onehot استفاده شده است.

هر کاراکتر یک بردار که همه درایه های آن به جز n
صفر است. n همان کاراکتر فعلی است.

کد:



```
chars = list(sorted(set(chain(*emotion_df['content']))))
char_to_idx = {ch: idx for idx, ch in enumerate(chars)}
max_sequence_len = max(len(x) for x in emotion_df['content'])

char_vectors = []
for txt in emotion_df['content']:
    vec = np.zeros((max_sequence_len, len(char_to_idx)))
    vec[np.arange(len(txt)), [char_to_idx[ch] for ch in txt]] = 1
    char_vectors.append(vec)
char_vectors = np.asarray(char_vectors)
char_vectors = pad_sequences(char_vectors)
labels = label_encoder.transform(emotion_df['sentiment'])
```

```
def split(lst):
    training_count = int(0.9 * len(char_vectors))
    return lst[:training_count], lst[training_count:]

training_char_vectors, test_char_vectors = split(char_vectors)
training_labels, test_labels = split(labels)

# آموزش و ارزیابی مدل
char_cnn_model.fit(training_char_vectors, training_labels,
                    epochs=20, batch_size=1024)
char_cnn_model.evaluate(test_char_vectors, test_labels)
```

کد:



آموزش و ارزیابی مدل

نکات:

داده ها باید به دو مجموعه آموزشی و تست تقسیم شود.

در نهایت مدل آموزش داده و ارزیابی می شود.

بعد از ۲۰ گام:

دقت بدست آمده روی داده آموزشی ۳۹ درصد

روی داده تست، دقت ۳۱ درصد

تفاوت دو مقدار بیانگر بیش برآذش روی مدل است. مدل یاد نگرفته و فقط بخشی از داده آموزشی را حفظ کرده است.

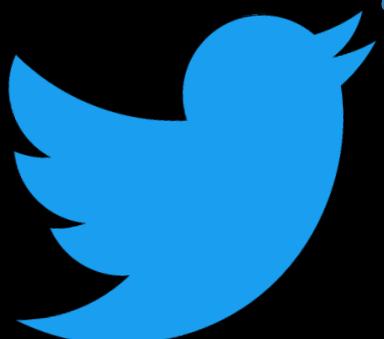
بحث

وقتی می خواهیم چیزی را یاد بگیریم که نمی دانیم دقیقا کجاست، کانولوشن خوب کار می کند.

مثلا: یادبگیریم love هرجای توییت آمد، می تواند یک برچسب خوب باشد. در CNN این کار به کمک پنجره انجام

می شود که اندازه پنجره ۶ است.

جمع آوری داده‌های توییتر



۴. استفاده از API توییتر

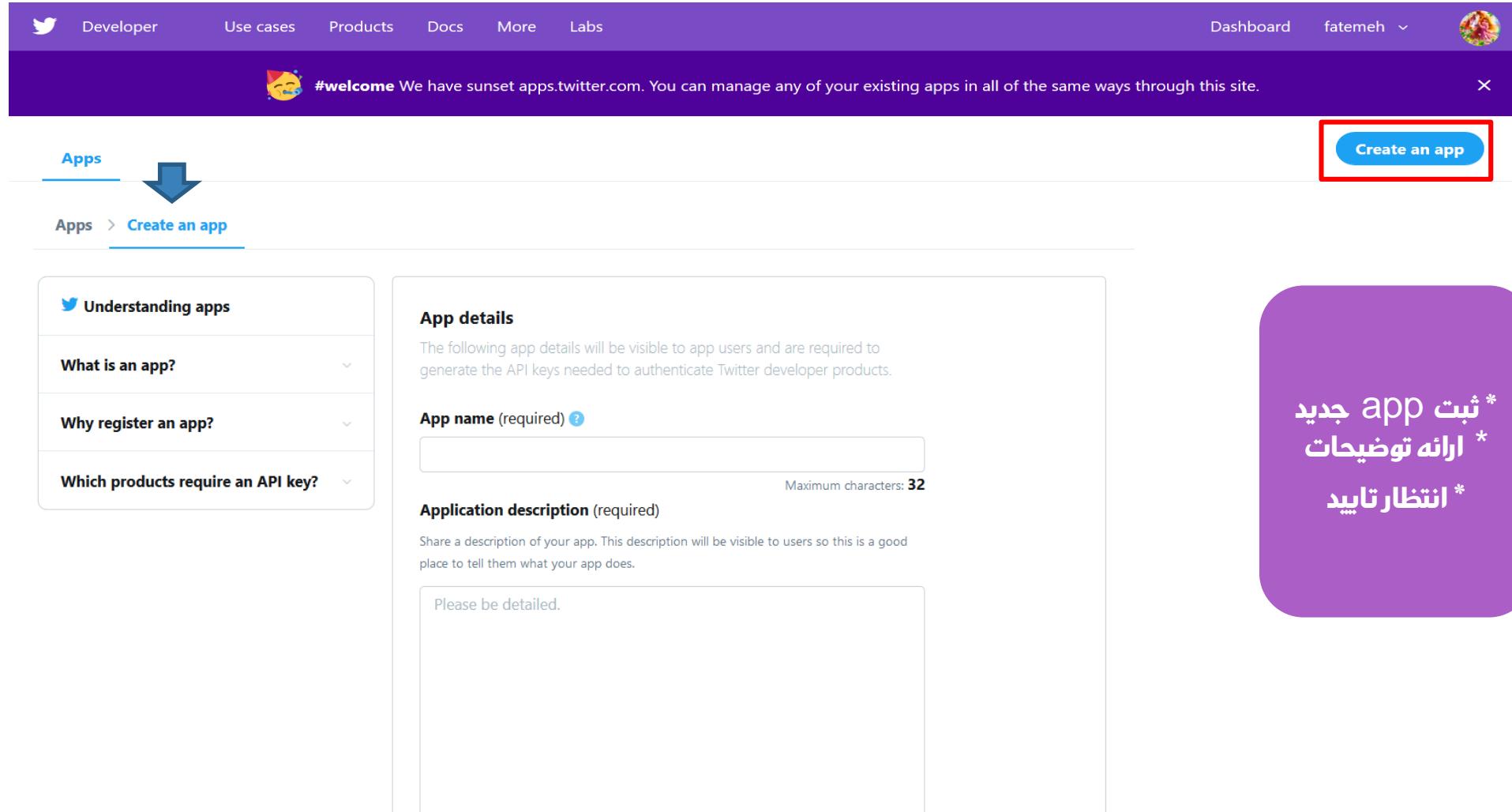
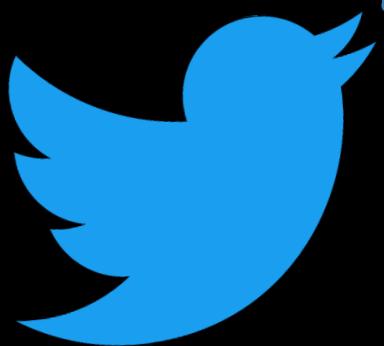
ساخت object
برای احراز هویت
(کدنویسی)

درباره کلیدها و
کدهای رمز

ثبت app جدید
ارائه توضیحات
انتظار تایید

رختن به آدرس
apps.twitter.com

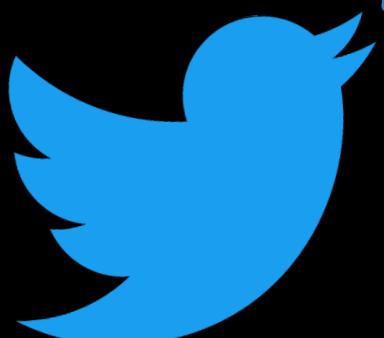
چگونه اپلیکیشن توئیتر را ساخته و API کیوی را تولید کنیم؟



The screenshot shows the Twitter Developer website's 'Create an app' page. At the top, there is a purple header with the Twitter logo, 'Developer', 'Use cases', 'Products', 'Docs', 'More', 'Labs', 'Dashboard', 'fatemeh', and a user icon. Below the header, a purple banner says '#welcome We have sunset apps.twitter.com. You can manage any of your existing apps in all of the same ways through this site.' On the left, there is a sidebar titled 'Understanding apps' with sections for 'What is an app?', 'Why register an app?', and 'Which products require an API key?'. The main content area is titled 'App details' and contains fields for 'App name (required)' and 'Application description (required)'. A red box highlights the 'Create an app' button at the top right of the main form.

* ثبت app جدید
* ارائه توضیحات
* انتظار تایید

چگونه از روش داده‌گیری تغذیه کنیم



Keys and tokens

Keys, secret keys and access tokens management.

Consumer API keys

2HDII0TMxCjWjhM7QAVihCRvq (API key)

gctVK1YRJK4TMIuURtMAS6NhOqlkcbPkVzEoU5PuZMVhf9NYSc (API secret key)

Regenerate

Access token & access token secret

1008098850958176257-XboDX2szcJT94pXyUw9kwRym6CwsFr (Access token)

nEaa91mWTE58PaxlsmIOK0g7dam7vjdPJYHpDQm4A9xoJ (Access token secret)

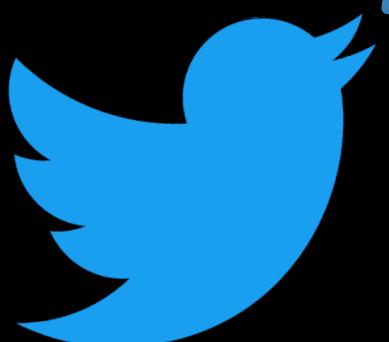
Read and write (Access level)



```
CONSUMER_KEY = '<your value>'  
CONSUMER_SECRET = '<your value>'  
ACCESS_TOKEN = '<your value>'  
ACCESS_SECRET = '<your value>'
```

دستیابی کلیدها و
کدهای رمز

چگونه آرای داده های Twitter را با Python

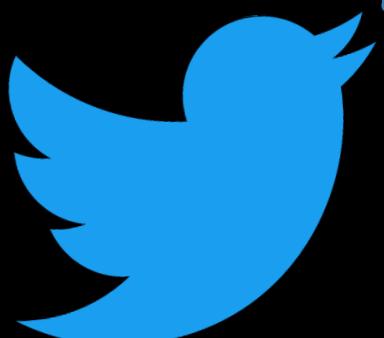


```
auth=twitter.OAuth(  
    consumer_key=CONSUMER_KEY,  
    consumer_secret=CONSUMER_SECRET,  
    token=ACCESS_TOKEN,  
    token_secret=ACCESS_SECRET,  
)  
  
status_stream = twitter.TwitterStream(auth=auth).statuses  
  
[x['text'] for x in itertools.islice(stream.sample(), 0, 5) if x.get('text')]
```

نمایش بخشی از توییت ها

ساخت object
برای احراز هویت
(کدنویسی)

نمایه‌آوری داده‌های توییت



فیلتر کردن توییت ها

```
def english_has_emoji(tweet):
    if tweet.get('lang') != 'en':
        return False
    return any(ch for ch in tweet.get('text', '') if ch in emoji.UNICODE_EMOJI)
```



```
tweets = list(itertools.islice(
    filter(english_has_emoji, status_stream.sample()), 0, 100))
```



توییت های دارای فقط یک ایموجی



```
stripped = []
for tweet in tweets:
    text = tweet['text']
    emojis = {ch for ch in text if ch in emoji.UNICODE_EMOJI}
    if len(emojis) == 1:
        emoji = emojis.pop()
        text = ''.join(ch for ch in text if ch != emoji)
    stripped.append((text, emoji))
```

استخراج توییت های
زبان انگلیسی
دارای ایموجی

۵. ایجاد یک پیش‌بینی کننده ساده

می‌توان از فایل آماده دیتا emoji.txt استفاده کرد.

دیتاست را به کمک پانداس می‌خوانیم، از هر ایموجی که کمتر از ۱۰۰۰ بار تکرار شده صرف نظر می‌شود:

```
all_tweets = pd.read_csv('data/emojis.txt',
                         sep='\t', header=None, names=['text', 'emoji'])
tweets = all_tweets.groupby('emoji').filter(lambda c: len(c) > 1000)
tweets['emoji'].value_counts()
```





مسئله:

لین ویتاست برای اینکه برداری فحیره شود خیلی بزرگ است

راه حل:

استفاده از یک data_generator

پانداس می تواند با استفاده از متده sample یک data_generator بسازد:

```
def data_generator(tweets, batch_size):
    while True:
        batch = tweets.sample(batch_size)
        X = np.zeros((batch_size, max_sequence_len, len(chars)))
        y = np.zeros((batch_size,))
        for row_idx, (_, row) in enumerate(batch.iterrows()):
            y[row_idx] = emoji_to_idx[row['emoji']]
            for ch_idx, ch in enumerate(row['text']):
                X[row_idx, ch_idx, char_to_idx[ch]] = 1
        yield X, y
```



مشابه مدل اولیه ای که ساخته شد، مدل با داده های فعلی آموزش داده می شود:

```
train_tweets, test_tweets = train_test_split(tweets, test_size=0.1)
BATCH_SIZE = 512
char_cnn_model.fit_generator(
    data_generator(train_tweets, batch_size=BATCH_SIZE),
    epochs=20,
    steps_per_epoch=len(train_tweets) / BATCH_SIZE,
    verbose=2
)
char_cnn_model.evaluate_generator(
    data_generator(test_tweets, batch_size=BATCH_SIZE),
    steps=len(test_tweets) / BATCH_SIZE
)
```

وقت مدل، ۴۰ درصد
روی داده تست ۳۵ درصد

بحث

با مکمل مدل قبلی، امکان پیشخواه ایموجی وجود دارد چون ایموجی ها همان برچسب های تعیین شده توسط نویسنده هستند.
چون برچسب های بیشتر از ایموجی ها هستند، نویزی تر خواهند بود.

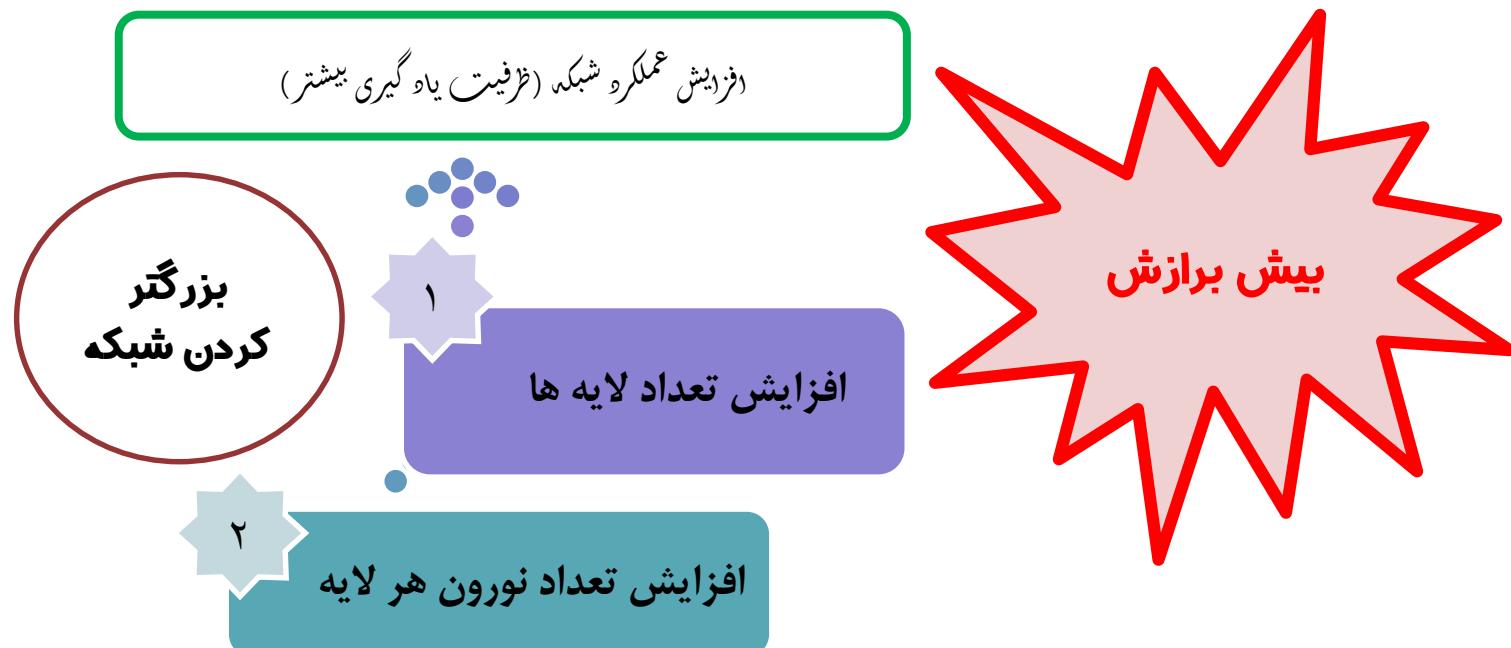
۶. چندگانه و پنجره های DropOut

مسئله:

چگونه می توان عملکرد شبکه را بهبود بخشد؟

راه حل:

استفاده از تکنیک دراپ اوت برای افزایش تعداد متغیرهای قابل یادگیری



سوال: چرا ۶ گام؟

- هر سه اندازه پنجره را لحاظ کرده و نتایج ترکیب می شود.
- تغییر تعداد کاراکترهای همسایه به ۴ یا ۵؟

```
layers = []
for window in (4, 5, 6):
    conv_1x = Conv1D(128, window, activation='relu',
                     padding='valid')(char_input)
    max_pool_1x = MaxPooling1D(4)(conv_1x)
    conv_2x = Conv1D(256, window, activation='relu',
                     padding='valid')(max_pool_1x)
    max_pool_2x = MaxPooling1D(4)(conv_2x)
    layers.append(max_pool_2x)

merged = Concatenate(axis=1)(layers)
```

دقت آموزش: ۴۷ درصد

دقت تست: ۳۷ درصد

- افزایش بیش برازش

- رفع بیش برازش با **dropout**

- استفاده از دراپ اوت

- به جای آموزش کل نودهای شبکه فقط برخی از نودها را به صورت تصادفی انتخاب کرده و آموزش می دهد.
- تکیه به برخی نورون های خاص را کم می کند.

```
for window in (4, 5, 6):
    conv_1x = Conv1D(128, window,
                      activation='relu', padding='valid')(char_input)
    max_pool_1x = MaxPooling1D(4)(conv_1x)
    dropout_1x = Dropout(drop_out)(max_pool_1x)
    conv_2x = Conv1D(256, window,
                      activation='relu', padding='valid')(dropout_1x)
    max_pool_2x = MaxPooling1D(4)(conv_2x)
    dropout_2x = Dropout(drop_out)(max_pool_2x)
    layers.append(dropout_2x)

merged = Concatenate(axis=1)(layers)

dropout = Dropout(drop_out)(merged)
```

استفاده از dropout با مقدار ۰،۲
صحبت داده های آموزشی ۴۳ درصد
داده های تست: ۳۹ درصد

بحث

هنوز هم می توان دقت را افزایش داد.

تعداد لایه

دراپ اوت

اندازه چگره

Hyperparameter Tuning

تغییر هایپرپارامترها

یافتن بهترین هایپرپارامترها

بینه سازی مدل

یافتن بهترین مدل

ساخت یک مدل سطح کلمه



سافت یک مدل سistem کلمه

مسئله: توییت ها توالی کلمات هستند نه توالی رندوم کاراکترها. چگونه می توانیم از اتفاق به خوبی استفاده کنیم؟

راه حل: مدلی را آموزش بدهید که به عنوان ورودی دنباله کلمات را به جای دنباله کاراکتر ها بگیرد.

ساخت یک مدل در سطح کلمه

در این مرحله جداساز کلمه که ۵۰۰۰۰ کلمه بالا را نگه می دارد، می سازیم. سپس آن را به مجموعه train, test اعمال می کنیم. بعد به آنها صفر اضافه کرده تا طول منحصر به فرد داشته باشند.

۱. جدا سازی توییت ها (Tokenization)

۲. آموزش با استفاده از embedding های موجود

۳. ساخت مدل شبیه به مدل کاراکتری

۴. بهبود دقیق

کد مربوطه:

```
VOCAB_SIZE = 50000
tokenizer = Tokenizer(num_words=VOCAB_SIZE)
tokenizer.fit_on_texts(tweets['text'])
training_tokens = tokenizer.texts_to_sequences(train_tweets['text'])

test_tokens = tokenizer.texts_to_sequences(test_tweets['text'])
max_num_tokens = max(len(x) for x in chain(training_tokens,
test_tokens))
training_tokens = pad_sequences(training_tokens,
 maxlen=max_num_tokens)
test_tokens = pad_sequences(test_tokens, maxlen=max_num_tokens)
```

ساخت یک مدل در سطح کلمه

این تابع word2vec embedding را لود کرده و آنها را با کلمات داخل پیکره مطابقت می دهد، سپس ماتریس می سازد که یک ردیف برای هر کدام از توکن ها دارد که از وزن هایی که از مدل word2vec لود شده اند، تشکیل شده است.

۱. جدا سازی توییت ها (Tokenization)

۲. آموزش با استفاده از embedding های موجود

۳. ساخت مدل شبیه به مدل کاراکتری

۴. بهبود دقیق

کد مربوطه:

```
def load_w2v(tokenizer=None):
    w2v_model = gensim.models.KeyedVectors.load_word2vec_format(
        word2vec_vectors, binary=True)

    total_count = sum(tokenizer.word_counts.values())
    idf_dict = {k: np.log(total_count/v)
                for (k,v) in tokenizer.word_counts.items()}

    w2v = np.zeros((tokenizer.num_words, w2v_model.syn0.shape[1]))
    idf = np.zeros((tokenizer.num_words, 1))

    for k, v in tokenizer.word_index.items():
        if < tokenizer.num_words and k in w2v_model:
            w2v[v] = w2v_model[k]
            idf[v] = idf_dict[k]

    return w2v, idf
```

ساخت یک مدل در سطح کلمه

مدل شبیه به مدل کاراکتری می سازیم، اکثر تفاوتش در روش پردازش ورودی است. ورودی دنباله ای از توکن ها می گیرد و لایه های embedding دنبال هر کدام از توکن ها که در

ماتریسی که ساختیم، می گردد:

۱. جدا سازی توییت ها (Tokenization)

۲. آموزش با استفاده از embedding های موجود

۳. ساخت مدل شبیه به مدل کاراکتری

۴. بهبود دقیق

کد مربوطه:

```
message = Input(shape=(max_num_tokens,),  
                dtype='int32', name='title')  
  
embedding = Embedding(mask_zero=False, input_dim=vocab_size,  
                      output_dim=embedding_weights.shape[1],  
                      weights=[embedding_weights],  
                      trainable=False,  
                      name='cnn_embedding')(message)
```

افزایش دقت با استفاده از تنظیم مقدار ابر پارامتر ها و تغییر
متغیر trainable به true

ساخت یک مدل در سطح کلمه

۱. جدا سازی توییت ها (Tokenization)

۲. آموزش با استفاده از embedding های موجود

۳. ساخت مدل شبیه به مدل کاراکتری

۴. بهبود دقت

بحث

پرا و پگونه

چرا دقّت از مدل کاراکتری پایین تر پود؟



- ما از word embedding برای شروع سریع تر استفاده کردیم. هر کلمه را با یک وکتور معنایی نمایش می دهیم، و آن را به عنوان ورودی به مدل می دهیم.
- دقّت بهتر از مدل سطح کاراکتر نبوده، و خیلی هم از مدل بیزی بهتر کار نمی کرد.
- پس word embedding آموزش داده شده، خوب نبوده است.
- چون مدل بر اساس Google news بوده است.

ساخت
embedding
دلخواه خودمان



سافت دلخواه فودمان embedding

مسئله: چگونه می توانیم embedding بسازیم که با پیکره ما همخوانی داشته باشد؟

راه حل: Word embedding را خودتان آموزش بدھید!

ساخت embedding

دلخواه خودمان

با استفاده از generator دنباله‌ای از توکن‌ها را ایجاد شده و از آن برای ساخت واژه و در نهایت آموزش مدل استفاده می‌شود.

۱. آموزش مدل با استفاده از gensim

۲. امتحان روش با استفاده از داده‌های فیلتر شده

۳. ساخت مدل با gensim



کد مربوطه:

```
class TokensYielder(object):
    def __init__(self, tweet_count, stream):
        self.tweet_count = tweet_count
        self.stream = stream

    def __iter__(self):
        print('!')
        count = self.tweet_count
        for tweet in self.stream:
            if tweet.get('lang') != 'en':
                continue
            text = tweet['text']
            text = html.unescape(text)
            text = RE_WHITESPACE.sub(' ', text)
            text = RE_URL.sub(' ', text)
            text = strip_accents(text)
            text = ''.join(ch for ch in text if ord(ch) <
                          128 and ch.isalnum())
            if text.startswith('RT '):
                text = text[3:]
            text = text.strip()
            if text:
                yield text_to_word_sequence(text)
                count -= 1
                if count <= 0:
                    break
```

می توانیم توییت های مثلاً یک هفته را در فایل ذخیره کرده، سپس بعد از آن یک generator از آن ها عبور دهیم. می توانیم روی آن ۱۰۰۰۰۰ داده فیلتر شده کار کنیم، بینیم اصلاً کار می کند یا نه؟

کد مربوطه:

```
tweets = list(TokensYielder(100000,  
                             twitter.TwitterStream(auth=auth).status-  
                             es.sample()))
```

ساخت embedding

دلخواه خودمان

۱. آموزش مدل با استفاده از gensim

۲. امتحان روش با استفاده از داده های فیلتر شده

۳. ساخت مدل با gensim

ساخت embedding

دلخواه خودمان

مدل را با gensim می سازیم

۱. آموزش مدل با استفاده از gensim

۲. امتحان روش با استفاده از داده های فیلتر شده

۳. ساخت مدل با gensim

کد مربوطه:

```
model = gensim.models.Word2Vec(tweets, min_count=2)
```

خروجی:

```
model.wv.most_similar(positive=['love'], topn=5)
```

```
[('hate', 0.7243724465370178),  
 ('loved', 0.7227891087532043),  
 ('453', 0.707709789276123),  
 ('melanin', 0.7069753408432007),  
 ('appreciate', 0.696381688117981)]
```

“Melanin” is slightly less expected.

بحث

پرا و پگونه

چرا word embedding په صورت آماده نه ؟



- در زمان عدم تطابق پیکره ها، ما دیتاست بزرگ و مناسبی داریم به راحتی می توانیم خودمان را بسازیم.
- تنظیم ویژگی trainable به true است.

استفاده از شبکه عصبی بازگشتی برای طبقه بندی

استفاده از شبکه عصبی بازگشتی برای طبقه بندی

مسئله: مطمئنا راهی برای استفاده از این واقعیت که یک توابیت دنباله‌ای از کلمات است، وجود دارد. حالا چگونه

این کار را انجام دهیم؟

راه حل: از یک شبکه بازگشتی سطح کلمه برای طبقه بندی استفاده کنید.

استفاده از شبکه عصبی بازگشتی برای طبقه بندی

شبکه های کانولوشن برای پیدا کردن الگوهای محلی در ورودی خوب است. برای تحلیل احساسات، اغلب خوب کار می کند، در صورتی که پیشنهاد ایموجی، یک عنصر زمان در خود دارد. معمولاً در انتهای توییت و در نتیجه گیری ایموجی می آید.

۱. شبکه CNN

۲. شبکه LSTM و RNN

استفاده از شبکه عصبی بازگشتی برای طبقه بندی

ما می توانیم از RNN برای پیشنهاد ایموجی استفاده کنیم.
کاملاً مانند یک CNN در سطح کلمه، ما کلمات را ورودی
داده و به معادل های embedding تبدیل می شود. یک
لایه LSTM خیلی خوب کار می کند.

۱. شبکه CNN

۲. شبکه LSTM و RNN

کد مربوطه:

```
def create_lstm_model(vocab_size, embedding_size=None,  
embedding_weights=None):  
    message = layers.Input(shape=(None,), dtype='int32',  
name='title')  
    embedding = Embedding(mask_zero=False, input_dim=vo-  
cab_size,  
                           output_dim=embed-  
ding_weights.shape[1],  
                           weights=[embedding_weights],
```

کد مربوطه:

```
trainable=True,  
name='lstm_embedding'))(message)  
  
lstm_1 = layers.LSTM(units=128, return_sequences=False)(embedding)  
category = layers.Dense(units=len(emojis), activation='softmax')(lstm_1)  
  
model = Model(  
    inputs=[message],  
    outputs=[category],  
)  
model.compile(loss='sparse_categorical_crossentropy',  
              optimizer='rmsprop', metrics=['accuracy'])  
return model
```



بحث

پرا و پگونه

چهرا LSTM



- مدل LSTM که ما اینجا استفاده کردیم، بسیار بهتر از مدل CNN عمل کرد.
- سطح کلمه از سطح کاراکتر بهتر عمل می کند.

محور سازی توافقات

تصویرسازی توافقات

مسئله: شما احتمالا دوست دارید که به صورت تصویری ببینید که چگونه مدل های متفاوت که ساختیم، در عمل با هم تفاوت دارند.

راه حل: از pandas برای نشان دادن شباهت ها و تفاوت ها استفاده کنید.

تصویر سازی توافقات:

با گرفتن دیتای تست برای مدل کاراکتری مان که به صورت یک وکتور به جای یک generator ذخیره شده است، شروع می‌کنیم.

۱. تست برای مدل کاراکتری

۲. تست برای ۱۰۰ مورد

۳. پیش‌بینی ایموجی

کد مربوطه:

```
test_char_vectors, _ = next(data_generator(test_tweets,  
None))
```

تصویر سازی توافقات:

برای ۱۰۰ مورد تست می کنیم.

۱. تست برای مدل کاراکتری

۲. تست برای ۱۰۰ مورد

۳. پیش بینی ایموجی

کد مربوطه:

```
predictions = {  
    label: [emojis[np.argmax(x)] for x in pred]  
    for label, pred in (  
        ('lstm', lstm_model.predict(test_tokens[:100])),  
        ('char_cnn', char_cnn_model.predict(test_char_vec-  
            tors[:100])),  
        ('cnn', cnn_model.predict(test_tokens[:100])),  
    )  
}
```

تصویر سازی توافقات:

با استفاده از pandas dataframe را با ۲۵ پیش بینی اول برای هر کدام از مدل ها، ایموجی اصلی و پیش بینی شده را، ساخته و نمایش می دهیم.

۱. تست برای مدل کاراکتری

۲. تست برای ۱۰۰ مورد

۳. پیش بینی ایموجی

#	content	true	char	cnn	cnn	lstm
0	@Gurmeetramrahim @RedFMIIndia @rjraunac #8DaysToLionHeart Great					
1	@suchsmallgods I can't wait to show him these tweets					
2	@Captain_RedWolf I have like 20 set lol WAYYYYYYY ahead of you					
3	@OtherkinOK were just at @EPfestival, what a set! Next stop is @whelanslive on Friday 11th November 2016.					
4	@jochendria: KathNiel with GForce Jorge. #PushAwardsKathNiels					

5	Okay good				
6	"Distraught means to be upset" "So that means confused right?" -@ReevesDakota				
7	@JennLiri babe wtf call bck I'm tryna listen to this ring tone				
8	does Jen want to be friends? we can so be friends. love you, girl. #BachelorIn-Paradise				
9	@amwalker38: Go Follow these hot accounts @the1stMe420 @DanaDeelish @So_deelish @aka_teenoney38 @CamPromoXXX @SexyLThings @l...				

10 @gspisak: I always made fun of the parents that show up 30+ mins early to pick up their kids today that's me At least I got a...



11 @ShawnMendes: Toronto Billboard. So cool! @spotify #ShawnXSpotify go find them in your city



12 @kayleeburt77 can I have your number? I seem to have lost mine.



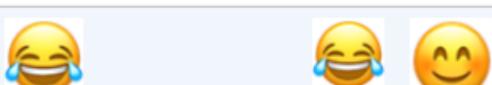
13 @KentMurphy: Tim Tebow hits a dinger on his first pitch seen in professional ball



14 @HailKingSoup...



15



@RoxeteraRibbons Same and I have to figure to prove it



16 @theseoulstory: September come-backs: 2PM, SHINee, INFINITE, BTS, Red Velvet, Gain, Song Jieun, Kanto...



17 @VixenMusicLabel - Peace & Love



18 @iDrinkGallons sorry



19 @StarYouFollow: 19- Frisson



20 @RapsDaiLy: Don't sleep on Ugly God



21 How tf do all my shifts get picked up so quickly?! Wtf



نتیجه:

با مرور این نتایج، ما متوجه می شویم که اغلب زمانی مدل اشیاه می کند که دو تا برچسب خیلی به هم شبیه اند.

- 22 @ShadowhuntersTV: #Shadowhunters fans, how many s would YOU give this father-daughter #FlashbackFriday bonding moment betwee...



- 23 @mbaylisxo: thank god I have a uniform and don't have to worry about what to wear everyday



- 24 Mood swings like...





بحث

پرا و پگونه

مدل پرا گاهی اوّقات اشتباه می کند و چه راه محلی داریم؟



- اشتباه در ایموجی های مشابه و هم معنی است؛
- می توانیم آنها را یکی در نظر بگیریم
- می توانیم تابعی برای محاسبه خطای داشته باشیم

ادغام مدل‌ها

ادغام مدل ها

مسئله: می خواهیم از ادغام کردن مدل ها برای افزایش قدرت مدل و به دست آوردن جواب بهتر استفاده کنیم.

راه حل: مدل های مختلف را به صورت یک مدل گروهی در بیاوریم.

ادغام مدل‌ها

می‌توانیم سه مدل را به یک مدل تبدیل می‌کنیم. ورودی را به سه مدل می‌دهیم سپس برای لایه خروجی، یک لایه با استفاده از `average` قرار می‌دهیم.

۱. ایده استفاده از خرد جمعی

۲. استفاده از تولید کننده‌های داده مختلف

۳. آموزش مدل



کد مربوطه:

```
def prediction_layer(model):
    layers = [layer for layer in model.layers
              if layer.name.endswith('_predictions')]
    return layers[0].output

def create_ensemble(*models):
    inputs = [model.input for model in models]
    predictions = [prediction_layer(model) for model in
models]
    merged = Average()(predictions)

    model = Model(
        inputs=inputs,
        outputs=[merged],
    )
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='rmsprop',
                  metrics=['accuracy'])
    return model
```

ادغام مدل‌ها

ما به data generator می‌توانیم مدل‌های مختلفی برای آموزش این مدل احتیاج داریم. به جای یک ورودی الان سه تا داریم. از آنجایی که آنها اسامی مختلفی دارند؛ ما می‌توانیم data generator طوری بسازیم که سه ورودی را پوشش دهد.

۱. ایده استفاده از خرد جمعی

۲. استفاده از تولید کننده‌های داده مختلف

۳. آموزش مدل



کد مربوطه:

```
def combined_data_generator(tweets, tokens, batch_size):
    tweets = tweets.reset_index()
    while True:
        batch_idx = random.sample(range(len(tweets)),
                                   batch_size)
        tweet_batch = tweets.iloc[batch_idx]
        token_batch = tokens[batch_idx]
        char_vec = np.zeros((batch_size,
                           max_sequence_len, len(chars)))
        token_vec = np.zeros((batch_size, max_num_tokens))
        y = np.zeros((batch_size,))
        it = enumerate(zip(token_batch, tweet_batch.iter-
                           rows()))
```

```
for row_idx, (token_row, (_, tweet_row)) in it:
    y[row_idx] = emoji_to_idx[tweet_row['emoji']]
    for ch_idx, ch in enumerate(tweet_row
                                ['text']):
        char_vec[row_idx, ch_idx, char_to_idx
                  [ch]] = 1
    token_vec[row_idx, :] = token_row
yield {'char_cnn_input': char_vec,
       'cnn_input': token_vec,
       'lstm_input': token_vec}, y
```

بحث

پرا و پگونه

چرا مدل های ادغام شده؟



- مدل های ادغام شده یا مدل های گروهی یک راه عالی برای ادغام روش های مختلف برای یک مسئله است.
معمولا تو مسابقات هوش مصنوعی مانند kaggle از این تکنیک استفاده می شود.

<https://deepmoji.mit.edu/>

DeepMoji

https://deepmoji.mit.edu

Help AI research by describing how you felt when tweeting!

TEACH OUR AI! HIDE X

DeepMoji has learned to understand emotions and sarcasm based on millions of emojis. Here's a [video](#) explaining a bit more. Type a sentence to see what our AI algorithm thinks.

I hate exam and projects ...

SUBMIT

Words are highlighted based on emotional impact. Click a word to turn it on/off.

i hate exam and projects ...

🔫 😡 😢 😔 😐

high confidence

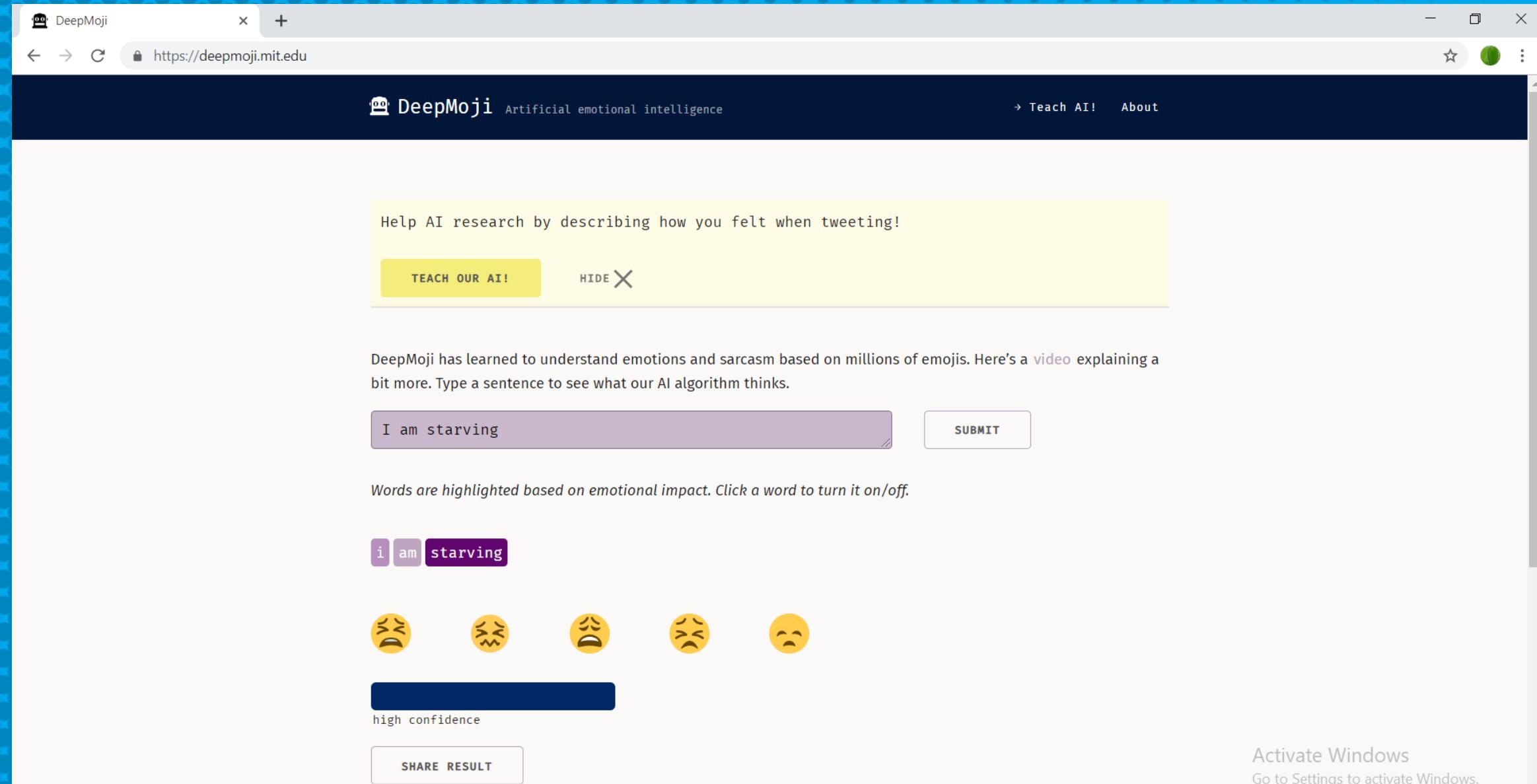
SHARE RESULT

Examples

Click on one!

You love hurting me, huh?

Activate Windows
Go to Settings to activate Windows.



19

DeepMoji x Google مترجم x | +

https://deepmoji.mit.edu

DeepMoji Artificial emotional intelligence

→ Teach AI! About

Help AI research by describing how you felt when tweeting!

TEACH OUR AI! HIDE X

DeepMoji has learned to understand emotions and sarcasm based on millions of emojis. Here's a [video](#) explaining a bit more. Type a sentence to see what our AI algorithm thinks.

you fooled me

SUBMIT

Words are highlighted based on emotional impact. Click a word to turn it on/off.

you fooled me

寐る手

zzz

汗

笑

低自信

SHARE RESULT

Activate Windows
Go to Settings to activate Windows.

<https://github.com/bfelbo/DeepMoji>



با سپاس از توجه شما