درس : NLP

استاد/TA : آقای دکتر هشام فیلی-آقای علی ابراهیمی

دانشجوی مهمان : رقیه یزدانی

پروژه CA شماره 4

پروژه استفاده از روشهای RNN و HMM برای POS Tagging

هدف این پروژه، استفاده از مدلهای RNN و HMM برای تعیین ادات سخن (POS) می‌باشد. دو فایل Rnn_train.xlsx و Rnn_test.xlsx به ترتیب برای آموزش و ارزیابی مدل RNN و دو فایل viterbi_train.xlsx و viterbi_test.xlsx برای آموزش و ارزیابی مدل HMM در اختیار قرار گرفته است.

روش HMM

اطلاعات کلی دادگان آموزشی و تست برای این بخش در جدول زیر آمده است :

viterbi_test.xlsx	viterbi_train.xlsx	
1,001	4,000	تعداد جملات
22,242	87,151	تعداد توکن/تگ
4,507	10,318	vocab_size
41	40	tag_size

فایل notebook با نام ***CA4_HMM_yazdani_98465104*** حاوی کدهای نوشته شده برای این بخش به همراه توضیحات مربوط به هر مرحله است. در بخش اول پس از خواندن فایل viterbi_train.xlsx، متدی به نام *sents_tags_tokenize* نوشته شده که اطلاعات دادگان آموزشی را خوانده و لیستی شامل زوج مرتبهای (word,tag) بر می‌گرداند. همچنین در این متد دو تگ _START_ و _END_ به ابتدا و انتهای هر جمله در لیست خروجی اضافه شده است.

در بخش بعد برای بدست آوردن ماتریسهای Transition و Emission ابتدا تعداد دفعات تکرار هر pos و تعداد دفعات تکرار ترکیبهای دوتایی posها و همچنین تعداد دفعات تکرار هر زوج word,pos در دادگان آموزشی بدست آمده و سپس ماتریسهای *trans_prob_matrix* و *emission_prob_matrix* محاسبه شد. با توجه به اینکه تعداد احتمالات با مقدار صفر در این ماتریسها زیاد بوده لذا از روش add-one برای smoothing استفاده شد.

مقادیر ماتریس Transition بر اساس احتمال زیر بدست آمد و به دلیل smoothing فرض شد که یک ترکیب دوتایی برای هر دو تگ انتخابی اضافه شده است (tag_size*tag_size زوج مرتب به دارگان آموزشی اضافه شده)

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

مقادیر ماتریس Emission بر اساس احتمال زیر بدست آمد و به دلیل smoothing فرض شد که یک ترکیب دوتایی برای هر کلمه از vocab و هر تگ انتخابی اضافه شده است (vocab_size*tag_size زوج مرتب به دارگان آموزشی اضافه شده)

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

سپس تابعی برای پیاده سازی الگوریتم *viterbi* با همین نام پیاده سازی شد که یک لیست از توکنهای یک جمله را دریافت کرده و لیست تگها(pos) های تعیین شده را بر میگرداند.

پس از پیاده سازی و تست ماتریسها و الگوریتم، مدل بر اساس دادگان تست (فایل viterbi_test.xlsx) مورد ارزیابی قرار گرفت. بدین ترتیب که ابتدا دادگان تست به کمک متد *new_sents_tags_tokenize* به صورت دو لیست از sequence ها تبدیل شد. یک لیست برای sequence های جملات و یک لیست حاوی sequence های تگهای صحیح متناظر. سپس sequence کلمات هر جمله به الگوریتم viterbi داده شد و تگهای predict شده بدست آمد و سپس به کمک کتابخانه sklearn.metrics مقادیر accuracy، recall و precision محاسبه گردید.

```
print("precision : ", precision_score(y_test, y_pred, average="micro"))
print("recall : ", recall_score(y_test, y_pred, average="micro"))
print("f1_score : ", f1_score(y_test, y_pred, average="micro"))
print("accuracy_score : ", accuracy_score(y_test, y_pred))
#print(confusion_matrix(y_test, y_pred))
```

```
precision :  0.880666584168626
recall :  0.880666584168626
f1_score :  0.880666584168626
accuracy_score :  0.880666584168626
```

تگهای predict شده برای داده های تست در ستون F از فایل اکسل viterbi_test.xlsx قرار داده شده است.

روش RNN

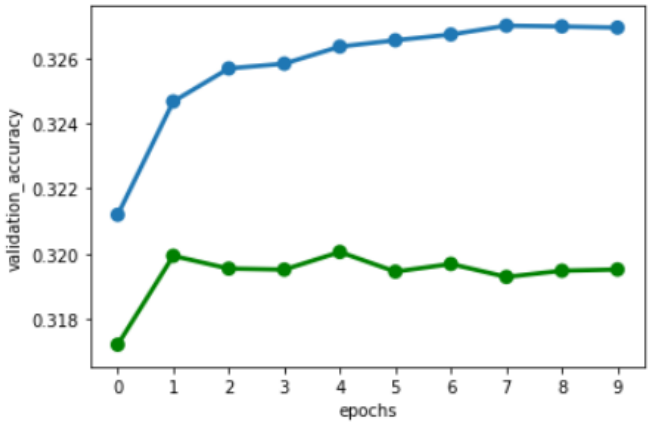
اطلاعات کلی دادگان آموزشی و تست برای این بخش در جدول زیر آمده است :

Rnn_test.xlsx	Rnn_train.xlsx	
5,000	20,001	تعداد جملات
108,827	437,958	تعداد توکن/تگ
10,865	21, 124	vocab_size
42	41	tag_size
104	70	ماکزیمم طول جملات

فایل notebook با نام ***CA4_RNN_yazdani_98465104*** حاوی کدهای نوشته شده برای این بخش به همراه توضیحات مربوط به هرمرحله است. در بخش اول پس از خواندن فایل Rnn_train.xlsx، متدی به نام *sents_tags_tokenize* نوشته شده که اطلاعات دادگان آموزشی را خوانده و چهار لیست برمی گرداند. یک لیست برای sequence های جملات و یک لیست حاوی sequence های تگهای صحیح متناظر و همچنین لیست کلیه کلمات و لیست کلیه تگها را بر می گرداند. همچنین در این متد تگ _PAD_ به ابتدا و انتهای هر جمله در لیستهای خروجی اضافه شده است.

همچنین برای تخصیص یک عدد به هر کلمه از vocab و همچنین تخصیص یک عدد به هر تگ و سپس استفاده از این اعداد برای تولید بردارهای one-hot از کلاسهای LabelEncoder و OneHotEncodel از کتابخانه sklearn.preprocessing استفاده شده است. و در مراحل بعدی نیز از کتابخانه keras برای ایجاد مدل و آموزش آن استفاده شد. با توجه به اینکه امکان آموزش مدل روی GPU فراهم نبود، لذا فرایند آموزش آن بسیار به کندی انجام شده و امکان تکرار تستهای مختلف با پارامترهای متفاوت برای رسیدن به نتیجه بهتر مقدور نبود. مقدار معیار accuracy پس از آموزش مدل برابر 0.31 بدست آمد.

مدل در دو مرحله و هر بار با 10 epoch آموزش داده شد که خروجی معیار ارزیابی در زمان آموزش مدل برای 10 epoch دوم بصورت زیر بوده است:



و پس از ارزیابی روی داده های تست؛ نتیجه زیر از متد evaluate از کتابخانه keras.models حاصل شد:

```
scores = model.evaluate(X_test_sents_padded, Y_test_onehot_tags)
print(f"{model.metrics_names[1]}: {scores[1] * 100}")
```

```
5000/5000 [==============================] - 11s 2ms/step
acc: 31.159999923706057
```

اما پس از اجرای متد predict از model آموزش دیده برای داده های تست و سپس بدست آوردن معیارهای ارزیابی مختلف بر اساس آن، نتیجه 0.94 برای accuracy بدست آمد.

```
print("precision : ", precision_score(y_test, y_pred, average="micro"))
print("recall : ", recall_score(y_test, y_pred, average="micro"))
print("f1_score : ", f1_score(y_test, y_pred, average="micro"))
print("accuracy_score : ", accuracy_score(y_test, y_pred))
#print(confusion_matrix(y_test, y_pred))
```

```
precision :  0.945667293312314
recall :  0.945667293312314
f1_score :  0.945667293312314
accuracy_score :  0.945667293312314
```

تگهای predict شده برای داده های تست در ستون F از فایل اکسل Rnn_test.xlsx قرار داده شده است. با توجه به اینکه حین عملیات تست، برخی از کلمات دادگان تست کنار گذاشته شدند، لذا تعداد تگ های predict شده با تعداد تگهای تعیین شده در فایل تست مغایرت داشت، به همین دلیل در ستون E از فایل اکسل Rnn_test.xlsx تگ معادل از فایل تست گذاشته شد.