بسم الله الرحمن الرحیم

پروژه پردازش متن

برای محصولات دیجیکالا

نسخه شماره ۲

تیم ارائه دهنده: داده کاوانِ فونیکس

ناظر و سرپرست تیم: دکتر علیرضا وفایی صدر

درخواست دهنده: ستاد علوم شناختی IPM

زمستان ۱۳۹۸

.

فهرست مطالب

تعریف پروژه

در فاز دوم این پروژه قرار است سه کار انجام شود:

- نگهداری و حفظ ایموجی ها و سمبل های شناختی مانند علامت سوال و علامت تعجب و ... و پیش پردازش متن با وجود این علائم و مقایسه با نتایج قبلی

- مدلسازی داده ها با شبکه هایی با معماری Attention

- تولید متن با GAN و تبدیل کامنت های با لایک کم به کامنت هایی با لایک بالا

پیش پردازش و آنالیز داده با حفظ ایموجی و سمبل ها

در این مرحله به هنگام پاکسازی و پیش پردازش متن، علائم ایموجی ها و سمبل های زیر نگه داشته شد:

? ! $ & # % ( )

نتایج جدید فرق چندانی با نتایج قبلی نداشت و فقط علامت کوتیشن ' زیاد به چشم میخورد و آنهم به خاطر استفاده زیاد از گیومه در کامنت هاست. همچنین نیم فاصله های بین کاراکترها هم به عنوان یک کاراکتر بسیار پر تکرار بسیار دیده می شد که با کد u200c مشخص شده بود که به دلیل اهمیت نداشتن، آن را حذف کردیم. در کل، ایموجی ها و سمبل های خاص، جزء ۶۰ کاراکتر پرتکرار در متون، دیده نشدند. کلیه فایل ها و word cloud های مربوط به عملیات پیش پردازش متن در مسیر زیر موجود است و می توانید در مسیر زیر مشاهده کنید:

Report2->Setare Toranj

دیتاست پاکسازی شده را در فایلی به اسم digi_clean_final.xlsx قرار دادیم و فایل مربوط به پیش پردازش متن با حفظ ایموجی ها در فایل Text Exploration4.ipynb قرار دارد.

همچنین برای عنوان های کامنت ها در ستون Title هم کلمات پرتکرار به دست آمد و برای کامنت های مثبت و منفی هم این لغات در عنوان کامنت ها بررسی شد. برای دو دسته بندی IT و AC هم بررسی شد و به صورت word cloud درآمده است که در فولدر output_title موجود است.

عملیات پیش پردازش داده و پاکسازی متون توسط *خانم ستاره ترنج* انجام شد.

مدلسازی داده با لایه Attention

برای کار با لایه های attention در شبکه ها، ابتدا کاربرد شبکه را یاد گرفته و سپس چهار نوع معماری مختلف را برای لایه attention تعریف کردیم. هدف این است که ببینیم اضافه کردن لایه attention دقت مدل را بالا می برد یا نه.

یک لایه توسط کلاس و سه لایه ی دیگر با توابع تعریف شده اند که در شکل زیر می بینید:

```
#Attention Class
class Attention(Model):
    def __init__(self, units):
        super(Attention, self).__init__()
        self.W1 =Dense(units)
        self.W2 = Dense(units)
        self.V = Dense(1)

    def call(self, features, hidden):
        hidden_with_time_axis = expand_dims(hidden, 1)
        score = nn.tanh(self.W1(features) + self.W2(hidden_with_time_axis))
        attention_weights = nn.softmax(self.V(score), axis=1)
        context_vector = attention_weights * features
        context_vector = reduce_sum(context_vector, axis=1)

        return context_vector, attention_weights

#Attention function
def Att1_layer(output, units):
    attention = Dense(embedding_dim)(output)
    attention = Permute((2,1))(attention)
    attention = Activation('softmax', name='attention_vec')(attention)
    attention = Permute((2,1))(attention)
    attention = Lambda(lambda x: K.mean(x,axis=2), name='attention', output_shape=
(int(output.shape[1]),))(attention)
    attention = RepeatVector(units)(attention)
    attention = Permute((2,1))(attention)
    output = multiply([output,attention])

    return output

#Attention2 function
def Att2_layer(output, units):
    attention = Dense(1, activation='tanh')(output)
    attention = Flatten()(attention)
    attention = Activation('softmax', name='attention_vec2')(attention)
    attention = RepeatVector(units)(attention)
    attention = Permute([2, 1])(attention)
    attention = multiply([output,attention])

    return attention

#Attention3 function
def Att3_layer(output, units):
    attention = TimeDistributed(Dense(1, activation='tanh'))(output)
    attention = Flatten()(attention)
    attention = Activation('softmax', name='attention_vec3')(attention)
    attention = RepeatVector(units)(attention)
    attention = Permute([2, 1])(attention)

    attention = multiply([activations, attention])
    attention = Lambda(lambda xin: K.sum(xin, axis=1))(attention)
    return attention
```

این لایه های را به غیر لایه ای که توسط کلاس، تعریف شده است روی دو مدل شبکه بازگشتی GRU و LSTM و همچنین دو مدل CNN اضافه کرده و همینطور برای اطمینان از اینکه لایه embedding فارسی مشکلی را ایجاد نخواهد کرد، همین مدل شبکه ها را با همین لایه های attention و با پارامترهای یکسان، برای داده های انگلیسی توییتر هم امتحان کردیم و نتایج و مقایسه در شکل زیر به دست آمد:

Twitter Data (English)				Digikala			
Model Accuracy		Attention Function	Result	Model Accuracy		Attention Function	
GRU	57%		SAME	GRU	25%		Attention Helped the model
AGRU	57%	Att3_layer()		AGRU	31%	Att3_layer()	
LSTM	64%		SAME	LSTM	30%		SAME
ALSTM	64%	Att2_layer()		ALSTM	30%	Att2_layer()	
CNN	63%		Attention ruined the model!	CNN	48%		Attention ruined the model!
ACNN	59%	Att2_layer()		ACNN	45%	Att2_layer()	
CNN2	57%		Attention Helped the model	CNN2	40%		Attention Helped the model
ACNN2	60%	Att2_layer()		ACNN2	42%	Att2_layer()	
ACNN2	58%	Att1_layer()		ACNN2	47%	Att1_layer()	

همانطور که در تصویر بالا می بینید برای دیتاست دیجیکالا لایه attention در مدل GRU دقت را بهطرز خوبی افزایش داده است. برای LSTM تغییری نداده و برای یکی از شبکه های CNN مدل را تقویت و دیگری را ضعیف کرده است و این بسته به نوع معماری شبکه و همینطور نحوه جایگیری لایه attention و خود معماری attention نیز دارد. ولی در کل با مقایسه با داده توییتر نیز مشخص می شود لایه attention اگر خوب تولید شود و در جای مناسب در مدل قرار گیرد می تواند به کارکرد مدل کمک کند.

همچنین با نگاهی به داده توییتر و مقایسه آن با داده دیجیکالا مشاهده شد که لایه embedding توییتر به خاطر ماهیت انگلیسی بودن کلمات روی شبکه های یکسان، بهتر کار کرده است و دقت بالاتری داده است. این نکته قابل ذکر است که شبکه های توییتر همگی دو کلاسه بوده اند و نه چهار کلاسه. تصاویر دو شبکه attentional و فایل نتایج و مقایسه ی کار و همگی در مسیر زیر قرار دارد:

Report2-->Armita Razavi

شبکه های attentional توسط *آرمیتا رضوی* تا الان انجام شده است.

تولید متن با معماری GAN

در این بخش، کار تولید متن با شبکه های GAN را داشتیم که قرار شد کامنت با لایک کم داده شود و کامنت با لایک بالا تولید شود.

تصمیم گرفته شد که generator داده هایی با لایک کم را به عنوان ورودی بگیرد و discriminator داده هایی با لایک زیاد به عنوان ورودی بگید علاوه بر داده های فیک تا بتواند کامنت های با لایک کم را به کامنت هایی با لایک بالا تبدیل کند.

در ابتدا قرار شد از لایه فریز شده و embedding مدل قبلی استفاده کنیم ولی از انجا که GAN برای داده های متنی تولید نشده است به مشکل خوردیم مخصوصا در لاس فانکشن و تا الان چهار GAN داریم که همگی با معماری هایی غیر از معماری قبلی شبکه، بدون embedding و با lstm و RNN تولید شده اند و دارند کار می کنند.

آقای وحید غفوریان، خانم رقیه فرجی، خانم زهرا نفریه و خانم طیبه قربی دارند روی GAN کار می کنند و به عنوان نمونه، دو مدل GAN را به عنوان دمو آماده کرده که در مسیرهای زیر موجود است:

Report2->roghaye_faraji

Report2-> Zahra_Nafarieh

*معماری خانم فرجی برای discriminator ، مثل مدل قبلی خودشان است که CNN بود ولی به جای embedding از LSTM استفاده کرده اند.

برای generator از دو معماری استفاده کرده اند، یکی CNN با LSTM و یکی CNN بدون LSTM که هر دو جواب میده ولی برای بدون lstm لاس فانکشن مهم است که چه باشد.

*معماری خانم نفریه برای discriminator ، از مدل FC یا Fully connected ها استفاده کردند و برای generator از دو معماری استفاده کرده اند، یکی FC با LSTM و یکی FC بدون LSTM که هر دو جواب میده ولی در معماری generator ای که FC و LSTM هر دو وجود دارند لاس dis و gen به همگرایی رسیده است و جواب بهتری گرفته اند.

شکل معماری و فایل های GAN هر کدام را می توانید در فولدرهایشان در مسیری که گفته شده است ببینید.

همچنین آقای غفوریان هم برای GAN تا الان از دو معماری FC برای generator و discriminator استفاده کرده اند. خانم قربی هم از معماری LSTM دارند استفاده می کنند و در حال بررسی و کار روی مدل های مختلف هستند.

کارها و تحقیقات آتی

تیم ما همچنان روی پروژه کار میکند. در قدم بعدی پیش پردازش و آنالیز داده میخواهیم اماری در بیاوریم از تعداد ایموجی ها و سمبل های موجود در متن و ببینیم چه نسبتی از متون را شامل می شوند.

بر روی شبکه های attentional همچنان کار می کنیم تا به مدلی پایدار با دقتی بالا و لایه ای مناسب برسیم.

همچنین روی GAN کار را ادامه داده تا بتوانیم کامنت هایی جدید تولید کنیم که هم خوانا باشد و هم اگر به کلاسیفایری داده شد، آن را در کلاسی با کامنته ای با لایک بالا قرار دهد. تلاش می کنیم از لایه های فریز شده هم استفاده کنیم اگر ممکن شد.

کلیه فایل های مربوط به این گزارش نسخه ۲ در گیت هاب زیر و در فولدر **Report2** موجود است.

https://github.com/phoenix-dataminers/Digikala

پایان