**بسم الله الرحمن الرحیم**

**پروژه پردازش متن**

**برای محصولات دیجیکالا**

نسخه شماره 4

تیم ارائه دهنده: داده کاوانِ فونیکس

ناظر و سرپرست تیم: دکتر علیرضا وفایی صدر

IPMدرخواست دهنده: ستاد علوم شناختی

بهار 1399

فهرست مطالب

|  |  |  |
| --- | --- | --- |
| 2 | ------------------------------------------------------------ | **تعریف پروژه** |
| 2 | ------------------------------------------------------------ | **تولید متن با معماری pre-trained قبلی** |
| 3 | ------------------------------------------------------------ | **برگرداندن داده های تولید شده به جملات** |
| 6 | ------------------------------------------------------------ | **میزان شباهت بین هر کلاس** |
| 8 | ------------------------------------------------------------ | **تولید لغات ابری به شکل درختی** |
| 10 | ------------------------------------------------------------ | **جمع بندی و تحقیقات آتی** |

تعریف پروژه

در ادامه فاز دوم و بحث های صورت گرفته، به ادامه کار تولید متن با لایک های بالا پرداختیم.

- تولید متن با همان مدل قبلی classifier ولی با نگهداری وزن ها و استفاده از مدل pre-trained

- معیاری برای میزان شباهت بین هر کلاس

- تولید لغات ابری با اشکال مختلف (درختی)

تولید متن با معماری pre-trained قبلی

برای اینکه کامنت های تولید کنیم که classifier آن ها را در کلاس با لایک بالا قرار دهد، از همان مدل قبلی که دقت 75% برای چهار کلاسه به ما داد استفاده کردیم. ولی این بار به جای چهار کلاس، دو کلاس like-dislike و verylike-dislike را انتخاب کرده و به عنوان کلاس با برچسب 0 و 1 به مدل دادیم. داده های این دو کلاس را روی این مدل دو کلاسه از اول train کردیم.

مدل bi-classification را که ترین کردیم روی داده های ترین دقت 95% و روی داده ولیدیشن دقتی بین 74% تا 80% داد. سپس برای تولید نویز، لایه embedding همین معماری را برداشتیم بعد از آن یکسری لایه trainable قرار دادیم که کار generator برای تولید داده های نویز را انجام دهد و تمامی لایه های معماری classifier قبلی را Freeze کرده، و وزن های لایه های classifier را save کردیم و لایه های آن را از حالت trainable خارج کردیم تا بتوان داده های نویز را با همان دقت قبل به کلاس verylike بیاندازد.

داده های نویز از لایه های trainable عبور کرده و تولید می شود و سپس وارد classifier فریز شده با وزن های قبلی، می شود که حکم discriminator را دارد که تشخیص دهد این داده های نویز وارد کلاس verylike شده اند یا نه.

معماری کلی برای تبدیل داده های نویز به کلاس لایک بالا دقتی معادل 97% داده است و چون از مدل قبلی که pre-trained شده استفاده کردیم و داده های واقعی قبلا توسط مدل دیده شده بود، به این خاطر دقت بالایی به دست آورده ایم.

برای تولید داده های نویز با کلاس، داده های واقعی را گرفته و در ستون کامنت هایی که کلاس verylike داشتند (ستون تارگت مقدار 1 داشت) بعد از Vectorize کردن، به جای کامنت های واقعی، مقادیر رندوم نویز در همان بازه ی مقادیر vectorize شده برای متون کامنت، قرار دادیم.

معماری مربوط به تولید نویز و تبدیل این داده های به کامنت هایی با کلاس verylike در فایل model\_final(Fake Comments).ipybn قرار دارد که توسط خانم رقیه فرجی انجام شده است.

مسیر فایل:

Report2-> roghaye\_faraji

برگرداندن داده های تولید شده به جملات

قرار شد بعد از تولید داده هایی با کامنت بالا، آن داده های را تبدیل به جملات کنیم و ببینیم که مدل پیشنهادی، چه جملات و کامنت هایی را تولید کرده است.

در معماری خانم فرجی، برای این کار، خروجی آخرین لایه قبل از لایه freeze شده در descriminator را گرفتیم و آن را در Inverse ماتریس embedding که وزن های آن را نگهداشته بودیم ضرب کردیم و ماتریسی شبیه one-hot vector به دست آمد که حاوی 20 سطر (طول یک جمله) و حدود 53000 ستون (هر ستون معادل یک کلمه vectorize شده است) بود. در هر سطر، ستونی را که بیشترین عدد را داشت به عنوان کلمه انتخابی برداشته و از Inverse دیکشنری word2vec که کلمات را به اعداد تبدیل کرده بودیم استفاده کرده و معادل کلمه آن عدد در آن ستون را دریافت کردیم. برای هر سطر به همین ترتیب عمل کرده و بزرگترین عدد موجود در هر ستون را به عنوان کلمه انتخابی در آن سطر، برگرداندیم و به این ترتیب 20 کلمه انتخاب شد که معادل یک کامنت فیک تولید شده است.

خروجی تولید شده از داده های fake که در دسته verylike قرار میگیرند را در زیر برای 5 کامنت مشاهده می کنید: کلمات و صفاتی که قرمز رنگ نشان داده شده اند کلمات با معنی ای هستند که امکان لایک خوردن را فراهم کرده اند.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| کامنت 5 | کامنت 4 | کامنت 3 | کامنت 2 | کامنت 1 |
| [['چنگالش'],  ['بلنیچر'],  ['احتیاجتون'],  ['اسپلیتر'],  ['افتضاحی'],  ['معذرت'],  ['بابات'],  ['قلبلمه'],  ['آمپرش'],  ['بدردنمیخوره'],  ['سامسونگش'],  ['معضل'],  ['یکسالهاین'],  ['کارمون'],  ['خکین'],  ['درهر'],  ['شارژکردم'],  ['تمومه'],  ['کر'],  ['شارژکردم'],  ['۶ماه'],  ['وسیعی'],  ['۵روز'],  ['بیصدا'],  ['جهات'],  ['هرکسی'],  ['اوکی'],  ['همیار'],  ['پیشنهادمیکنم'],  ['افتابی']] | [['اندونزی'],  ['خریدکالا۲۴'],  ['امکانتش'],  ['غلط'],  ['جاروی'],  ['قابلیتهاش'],  ['پاییز'],  ['دانم'],  ['کوهنوری'],  ['بدنسازی'],  ['خریدتون'],  ['بهترینش'],  ['ساعتی'],  ['یکسالهاین'],  ['تصاویرش'],  ['متقبل'],  ['نرو'],  ['نقص'],  ['پوستو'],  ['عالییه'],  ['میشونه'],  ['جاروی'],  ['تواین'],  ['میرسد'],  ['چسبونه'],  ['بادمجون'],  ['باطریش'],  ['همگان'],  ['پیشمون'],  ['جی۷']] | [['پولمو'],  ['انصافن'],  ['مان'],  ['افتابی'],  ['افتابی'],  ['موافق'],  ['افتابی'],  ['باحالیه'],  ['موافق'],  ['اسپلیتر'],  ['خوشکل'],  ['موافق'],  ['آشپزی'],  ['جی۷'],  ['چنینی'],  ['بنزیننش'],  ['افتابی'],  ['چنگالش'],  ['وتو'],  ['اسپلیتر'],  ['انصافن'],  ['افتابی'],  ['بادوام'],  ['استاندار'],  ['مخصوصن'],  ['ساویج'],  ['مخصوصن'],  ['وتو'],  ['خوشکل'],  ['تفلون']] | [['ضعیفیه',  'خریدکالا۲۴',  'هیجان',  'انکیز',  'تاحالا',  'ومناسب',  'پشیمانم',  'پک',  'موزد',  'معرکه',  '۹۴۵۰',  'تواین',  'حجمش',  'داشتنیه',  'ارزشمندی',  'چتری',  'سولفاتش',  'سامسونگش',  'قشنگی',  'ششم']] | [['دماش',  'عالییه',  'دیجیکالاییم',  'البت',  'چسپندگی',  'استحکام',  'یسری',  '۶ماه',  'اسپلیتر',  'عسلشه',  'بازدهه',  'قیمش',  'چنگالش',  'جبران',  'همراهانش',  'نظرمن',  'مدادی',  'اقدام',  'جیغهه',  'یورو']] |

البته دو کامنت تولید شده اول، با طول 20 و سه کامنت بعدی با طول جمله 30 تولید شده است.

اینکار نیز توسط خانم فرجی در همان فایل تولید کامنت، صورت گرفته است.

کد مدل bi-classifier را در شکل زیر مشاهده می کنید:

شکل زیر نیز معماری gan برای تولید کامنت های لایک بالا می باشد:

معیاری برای میزان شباهت بین هر کلاس

در بار اول که دو کلاس like-dislike و like-verylike را با هم ادغام کرده و تبدیل به یک کلاس کردیم و دو کلاس verylike-dislike و verylike-verydislike را نیز با هم یکی کردیم و اقدام به تولید کامنت کردیم، مدل bi-classifier دقت معادل 50% یا رندوم و تصادفی تولید میکرد. این احتمال را دادیم که بعد از ادغام هر دو کلاس، میزان شباهت و نزدیکی کلاسها با هم بسیار زیاد است و تفاوت چندانی بین دو کلاس جدید like و verylike وجود ندارد و مدل نمی تواند تمایزی بین دو کلاس قائل شود.

برای پی بردن به این موضوع، *آقای غفوریان* الگوریتمی را برای شباهت بین کامنت ها در هر چهار کلاس به صورت جدا به دست آورد. به این صورت که برای هر کلاس، کلمه های موجود در کامنت های آن کلاس را گرفته و در بین کلاسهای دیگر جستجو میکرد تا ببیند این کلمه در کامنت های کلاس دیگر، چند بار تکرار شده است. و برای هر کلمه این کار را انجام داده و تعداد تکرارهای هر کلمه را در آن کلاس با هم جمع کرد.

این کار را بین هر کلاس با سه کلاس دیگر انجام داد و برای مجموع تعداد کلمات تکرار شده بین هر دو کلاس، یک عدد به دست آمد که این عدد را تقسیم بر تعداد کل کلمات موجود در کامنت ها ( 20000 کلمه منحصربفرد) کرد و توانست عددی به دست بیاورد که نشانگر میزان شباهت و نزدیکی بین کلاس ها باشد.

میزان شباهت بین هر دو کلاس را در جدول زیر مشاهده می کنید:

|  |  |
| --- | --- |
| مقدار شباهت و نزدیکی دو کلاس به هم | نام کلاس ها |
| 24 | like-dislike & verylike-dislike |
| 19 | like-dislike & like-verydislike |
| 24 | like-dislike & verylike-verydislike |
| 19 | verylike-dislike & like-verydislike |
| 24 | verylike-dislike & verylike-verydislike |
| 24 | like-verydislike & verylike-verydislike |

اگر هر کدام از این اعداد را در تعداد کل کلمات منحصریفرد یعنی 24000 ضرب کنیم مجموع تعداد کلمات تکرار شده در بین دو کلاس را به ما میدهد که اعداد بزرگی است.

این میزان اعداد برای شباهت اعداد بزرگی هستند که نشان میدهد کلاسها از لحاظ تعداد کلمات مشترک بسیار به هم نزدیک هستند و برای همین ادغام هر دو کلاس باعث بیشتر شدن این شباهت شد.

الگوریتم آقای غفوریان در شکل زیر دیده می شود:

کد مربوط به این الگوریتم در فایل CalculatTheSubscription.py نوشته شده است.

مسیر فایل: Report2-> VahidGhafourian

تولید لغات ابری با اشکال مختلف (درختی)

در این بخش به تولید لغات ابری یا word cloud پرداختیم اما به شکل درختی که توسط *خانم زهرا نفریه* انجام شد. از همان کتابخانه قبلی Persian\_wordcloud استفاده کردیم اما حالت mask آن را روی tree گذاشتیم تا شکل تولید شده حالت درختی به خود گیرد.

کد در شکل زیر مشاهده می شود:

فایل کدهای لغات ابری و تصاویر آن در مسیر زیر موجود است:

Report2-> Zahra\_Nafarieh->word\_cloud

شکل های زیر توسط کد بالا یک با پس زمینه درخت و دیگری با پس زمینه پرنده توییتر تولید شده است:

جمع بندی و تحقیقات آتی

در این گزارش، به بررسی تولید کامنت های فیک با کلاس verylike پرداختیم. همچنین میزان شباهت بین هر چهار کلاس را سنجیدیم و شکل جدید از تشکیل لغات ابری را به دست آوردیم.

کلیه فایل های مربوط به این گزارش نسخه 4 در گیت هاب زیر و در فولدر Report2 موجود است.

https://github.com/phoenix-dataminers/Digikala2