به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس پردازش زبان طبیعی

پاسخ تمرین1

نام و نام خانودگی: مهدی موسوی

شماره دانشجویی:۸۱۰۱۰۲۲۶۴

اسفند ماه ۱۴۰۲

فهرست

سوال اول

پاسخ بخش اول

Tokenizer مد نظر مبتنی بر کلمه است زیرا میبینیم که در ابتدا و انتهای آن الگوی آن \b وجود دارد به این معنا که مرز کلمه هستند و به طور کلی میتوان گفت که این الگو شناساگر کلمات است.

در مورد نقاط ضعف tokenizer مبتنی بر کلمه به نکات زیر میتوان اشاره کرد:

۱. این حالت باعث میشود تعداد token های منحصر به فرد ما زیاد شود.
۲. باعث میشود که در لغات دیده نشده ضعیفتر باشیم چون اگر با زیر کلمه کار کنیم احتمال بیشتری دارد که زیر کلمه ها را دیده باشیم و بتوانیم معنی کلمه جدید را تشخیص دهیم.
۳. در مواردی همچون $ یا . بین کلمات برای اختصار به مشکل برمیخورد و نمیتواند مفهوم را به درستی انتقال دهد.
۴. در مورد کلمات دوبخشی همچون new York یا کلماتی مثل can’t به مشکل میخورد.

پاسخ بخش دوم

بعد از اجرای الگوریتم به خروجی زیر می رسیم:

['Just', 'received', 'my', 'M', 'Sc', 'diploma', 'today', 'on', '2024', '02', '10', 'Excited', 'to', 'embark', 'on', 'this', 'new', 'journey', 'of', 'knowledge', 'and', 'discovery', 'MScGraduate', 'EducationMatters']

دو نمونه از مشکلات این خروجی را میتوان به صورت زیر نام برد :

۱. این الگوریتم متوجه نشده است که .M.Sc مخفف است و M و Sc را جدا کرده است.
۲. متوجه تاریخ بودن عبارت 2024/02/10 هم نشده است و آن را به صورت سه عدد متوالی نشان داده است که معنای مد نظر ما را انتقال نمی دهد.

پاسخ بخش سوم

برای رفع مشکل تاریخ, الگو را از \b\w+\b به\b\w+\b | \d{4}/\d{2}/\d{2} تغییر میدهیم. در این الگوی جدید به دنبال الگوی ۴ عدد بعد \ ۲عدد\ ۲عدد نیز خواهیم گشت که نماینده حالت روز\ماه\ سال است و این مساله را حل میکند. خروجی پس از استفاده از این الگوی جدید به صورت زیر خواهد بود:

['Just', 'received', 'my', 'M', 'Sc', 'diploma', 'today', 'on', '2024/02/10', 'Excited', 'to', 'embark', 'on', 'this', 'new', 'journey', 'of', 'knowledge', 'and', 'discovery', 'MScGraduate', 'EducationMatters']

سوال دوم

بخش اول

هر دو مدل BERT و GPT از توکنایزر هایی استفاده میکنند که مبتنی بر زیرکلمه هستند.برای انتخاب این شیوه به عنوان روش مد نظر میتوان گفت که با این روش به نسبت حالت مبتنی بر کلمه شرایط بهتری در مورد کلماتی که خارج از دایره واژگان ما بوده اند داریم. در روش مبتنی بر کاراکتر هم مشکلی که وجود دارد این است که طول دنباله ها طولانی میشود و دریافت معنی دنباله برای مدل مشکل میشود. به طور کلی میتوان گفت روش مبتنی بر زیرکلمه یک تعادل نسبی میان دو حالت دیگر است و مشکلات این دو روش را رفع کرده است.

بخش دوم

در مدل BERT ما از توکنایزر wordpiece و در مدل GPT از توکنایزر BPE استفاده میکنیم. هر دوی این روش ها همانطور که اشاره شد از گونه های مبتنی بر زیر کلمه هستند. در روش BPE ما از یک دیکشنری پایه شروع میکنیم و در هر مرحله هر ترکیبی که بیشترین تکرار را داشت به دیکشنری خود اضافه میکنیم تا در نهایت لغات بلند تر تشکیل شوند.

در روش wordpiece هم ما در ابتدا یک دیکشنری پایه داریم و روند مشابه حالت قبل است با این تفاوت که برای اینکه انتخاب شود کدام دو ترکیب ادغام شوند از قاعده متفاوتی استفاده میشود به این صورت که یک امتیاز به صورت :

$$score = \frac{freq_{of\ pair}}{freq_{first\ element} \times freq_{second\ element}}$$ [1]

محاسبه میشود و بر اساس این امتیاز تعیین میشود که کدام زوج بهترین گزینه برای اضافه شدن به دیکشنری است.

بخش سوم

بعد از اجرای الگوریتم دیده شد که تعداد واژگان ساخته شده توسط روش wordpiece نزدیک ۸۸۰۰ و در روش BPE حدود ۱۱۵۰۰ کلمه بود. به طور کلی به نظر می رسد بین این دو الگوریتم معمولا wordpiece واژگان کمتری تولید میکند و در صورتی که دادگان هدف ما هم مشابه داده در دست ما است و کلمات جدید زیادی نداریم انتخاب بهتری است اما اگر در دادگان آزمون کلمات جدید زیادی داریم بهتر است که از BPE استفاده کنیم تا با تشخیص زیرکلمات بیشتر در کلمات جدید موفقتر عمل کنیم.

در مورد جمله اول تنها تفاوتی که دیده میشود در کلمه snew است که الگوریتم BPE به درستی آن را به دو قسمت s و new تقسیم کرده است اما روش دیگر آن را sne و w تفسیر کرده است.

[1] https://huggingface.co/learn/nlp-course/en/chapter6/6

در جمله دوم تفاوت های بیشتری وجود دارد به این صورت که tokens در روش wordpiece به to,ken,s و در روش دیگر به to,k,ens تبدیل شده است که دیده میشود در روش دوم جمع بودن تشخیص داده نشده است. یا در روش wordpiece کلمه generated به یک تکواژ تبدیل شده است اما روش دیگر به طور دقیقتری آن را به دو قسمت gener,ated تقسیم کرده است.

سوال سوم

بخش اول

در اولین بخش به tokenization متن داده شده می پردازیم و به کمک آن تمام متن داده شده در کتاب تارزان را به تعدادی توکن تبدیل میکنیم.

بخش دوم

در این بخش به آموزش یک مدل bigram میپردازیم.روش انتخاب کلمات بعدی به این صورت است که از بین خروجی هایی که برای bigram در دست داریم یکی را انتخاب میکنیم و به این صورت به ترکیبات پر تکرارتر هم احتمال بیشتری داده ایم. برای حل مساله data sparsity هم راه حل پیاده سازی شده به صورت back-off smoothing است به این معنی که اگر کلمه اگر ترکیب n-1 تایی دیده نشده باشد برای ترکیب n-2 تایی سعی میکنیم پیش بینی کنیم و به همین صورت عمل میکنیم تا به یک تک کلمه برسیم.

بخش سوم

با پیش بینی روی bigram آموزش داده شده به خروجی زیر برای ادامه جملات میرسیم:

1. he already he not know that does not necessarily keep zeyd approached blake
2. and the village would do what do so often hunted north of blood

بخش چهارم

در این قسمت به آموزش برای n های بالاتر میپردازیم.

برای n=3 داریم :

1. trail he did not return and was deflected from its scabbard he could not
2. back and usha tore through the black gave no heed either to time since

همچنین برای n=5 :

1. of the trail he took short cuts swinging through the branches of the trees a hundred
2. on the huge back listening to manu the monkey chattering and scolding among the trees then

مشاهده میشود که عبارت های تولید شده در این قسمت ارتباط های معنایی طولانی تری دارد و به طور کلی این ترکیبات کیفیت بالاتری دارد.

بخش پنجم

افزایش n نیاز به حجم داده بیشتری نیاز دارد تا بتواند ترکیبات بیشتری را پیدا کند.همچنین به این صورت ترکیبات دیده نشده بیشتری در دادگان آزمون دیده میشود و حجم محاسباتی هم به صورت نمایی زیاد میشود. مشکل دیگری که وجود دارد این است که generalization نیز کاهش می یابد و به این صورت با ترکیب طولانی ای که در دست داریم خروجی متناظر به طور تقریبی مشخص است و روی دادگان جدید به مشکل برخواهیم خورد.

سوال چهارم

در این سوال به پیش بینی احساسات به کمک n-gram ها میپردازیم.

بخش اول

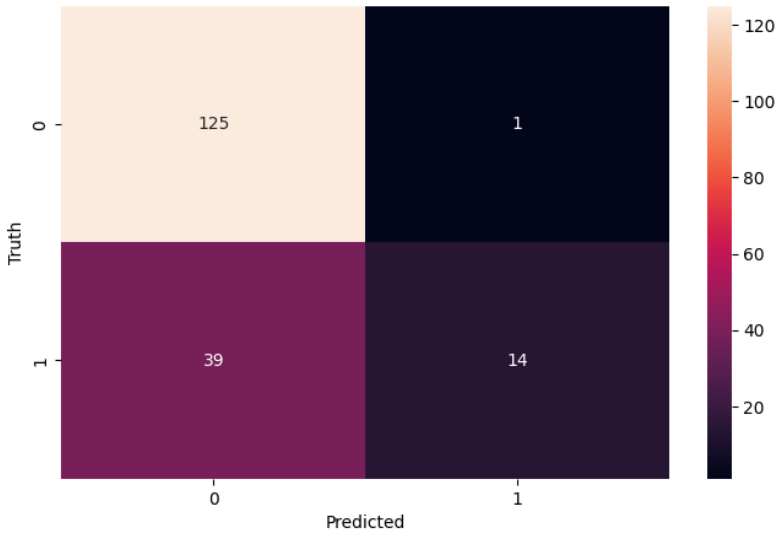
الگوریتمی که در قسمت پیش بینی به این صورت بود که :

۱. برای هر ردیف n-gram های مربوط را استخراج میکنیم.
۲. برای هر n-gram محاسبه میکنیم مقدار متناظر از positive_freq و negative_freq محاسبه میکنیم.
۳. اگر مقادیر متناظر با متن های منفی بیشتر بود روی ردیف برچسب منفی و در غیر این صورت برچسب مثبت میزنیم.

با اجرای مراحل یک تا سه برای تمام ردیف های داده به ازای همه ردیف ها پیش بینی به دست آورده ایم.

بخش دوم

در این مرحله از خروجی بخش استفاده میکنیم تا ببینیم شرایط مدل چطور بوده است. با اجرا کردن الگوریتم بخش دوم به صحت ۷۷/۶۵ رسیدیم و ماتریس آشفتگی متناظر هم در نمودار ۴-۱ آمده است. با توجه به اینکه تعداد ردیف های با احساسات منفی بیشتر از ردیف های با احساسات مثبت بوده است مدل بیشتر به این سمت تمایل دارد که پیش بینی کند که جمله بار منفی داشته است به همین علت میتوان گفت که روی رکورد های با احساسات منفی recall بالاتر و در رکورد های با احساسات مثبت precision بالاتر است.



نمودار ۴-۱: ماتریس آشفتگی خروجی مدل