به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

درس پردازش متن و زبان طبیعی

سحر رجبی
شماره دانشجویی
۸۱۰۱۹۹۱۶۵

اردیبهشت ماه 1400

سوال ۱

انجام پیش‌پردازش‌ها:

هر دوی مدل‌های پیاده‌سازی‌شده با استفاده از XLNet و Bert از آنجایی که از مدل‌های از پیش‌آموزش‌دیده هستند، نیاز به پیش‌پردازش‌های خاصی دارند که بتوانند عملکرد موردانتظار را داشته‌باشند. بخش زیادی از این پیش‌پردازش‌ها مشترک هستند که در ادامه توضیح می‌دهیم.

در ابتدا باید منشن‌ها، لینک‌ها و هشتگ‌ها از متن توییت‌ها حذف شوند. سپس بایستی تمامی حرف‌ها lower-case شوند و علائم نگارشی هم حذف شوند. بعد از انجام این کارها، نوبت به tokenize کردن متن می‌رسد که هر دوی این معماری‌ها، یک تابع برای انجام این کار -با توجه به دادگانی که با آن‌ها ترین شده‌اند- دارند که باید از آن‌ها استفاده کنیم تا در فرآیند آموزش به لغات ناشناخته برنخوریم (این tokenizerها این مشکلات را پوشش خواهند داد).

همچنین هر دو مدل XLNet و Bert نیاز به اضافه کردن توکن‌های خاصی به بخش قبل دارند. در واقع باید توکن [CLS] به ابتدای هر جمله و توکن [SEP] به انتهای آن اضافه شود. و سپس جملات کوتاه‌تر با استفاده از توکن [PAD] به طول مناسب خواهند رسید (تعدادی از این جملات ممکن است از بیشینه طول مد نظر ما طولانی‌تر باشند؛ که باید آن‌ها را هم از کلمه‌ای به بعد قطع کنیم). همچنین با توجه به پیشنهاد صورت سوال، بیشینه طول جملات برابر با ۱۲۸ درنظرگرفته شده است. برای اینکه از padding تاثیر نگیریم، یک attention mask تعریف می‌کنیم که در هر جایی که [PAD] داشته‌باشیم مقدار صفر دارد و در نتیجه آن بخش از دنباله در کانتکست خروجی این دو مدل، درنظر گرفته نخواهد شد.

طراحی شبکه‌ها:

برای هر دو، از پارامترهای پیشنهادشده در صورت سوال استفاده شده است (نرخ یادگیری برابر ۰.۰۰۰۲، اندازه‌ی دسته‌ها ۳۲ و اندازه‌ی لایه‌ی feedforward به اندازه‌ی خروجی مدل‌های bert و xlnet) و ما از مدل‌های پایه‌ای XLNetModel و BertModel استفادهه کردیم و سپس خروجی آن را به بخش feed forward شبکه دادیم. (مدل‌های مختص text classification هم موجودند که در اینجا استفاده نشده‌اند).

همچنین از CrossEntropyLoss به عنوان تابع خطا و از Adam optimizer برای پیش‌برد آموزش شبکه استفاده شده‌است.

پیاده‌سازی این مدل‌ها، در کدهای ارسال شده قابل مشاهده است.

سوال ۲

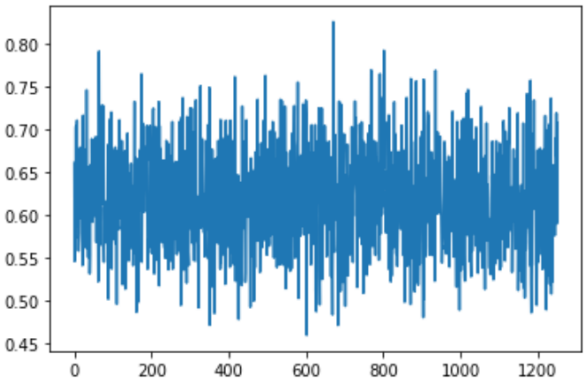
به‌طور کلی یکی از مزیت‌های استفاده از این مدل‌ها، نیاز به تکرار بسیار پایین است که می‌تواند باعث صرف زمان کمتری بشود.

ما برای هر یک از این مدل‌ها، سه مدل با تعداد تکرار ۴، ۷ و ۱۰ آموزش‌داده‌ایم که نتیجه‌ی هرکدام در ادامه گزارش می‌شود.

مقایسه‌ها و توضیحات بعد از گزارش‌های هر سه مدل آورده شده است.

XLNet:

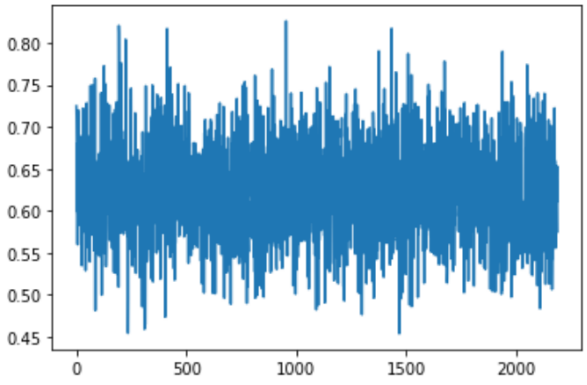
تصویر زیر نمودار تغییرات loss در طول ۴ اپیاک (و با طول batch برابر ۳۲ می‌باشد)



و جدول زیر معیارهای precision، recall، f1 و accuracy را گزارش کرده است.

	precision	recall	f1-score	support
0	0.67	0.54	0.59	1740
1	0.50	0.63	0.56	1260
accuracy			0.58	3000
macro avg	0.58	0.58	0.58	3000
weighted avg	0.60	0.58	0.58	3000

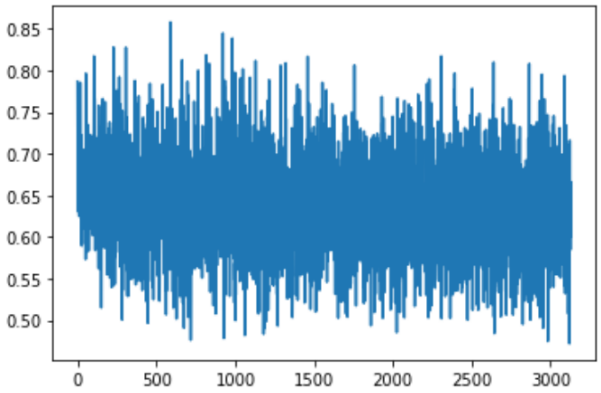
نمودار زیر هم تغییرات خطا در طول ۷ ایپاک با طول دسته‌ی ۳۲ را نشان می‌دهد.



و جدول معیارهای خواسته‌شده بر روی دادگان تست، بعد از انجام آموزش هم مطابق زیر است.

	precision	recall	f1-score	support
0	0.67	0.59	0.63	1740
1	0.51	0.59	0.55	1260
accuracy			0.59	3000
macro avg	0.59	0.59	0.59	3000
weighted avg	0.60	0.59	0.60	3000

در انتها هم مدلی با ۱۰ ایپاک آموزش دیده است که به ترتیب نمودار تغییرات خطا و گزارش معیارهای آن را مشاهده می‌کنید.

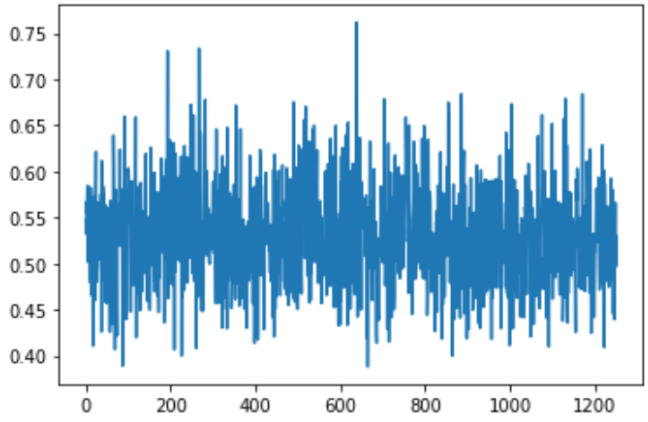


	precision	recall	f1-score	support
0	0.66	0.55	0.60	1740
1	0.50	0.61	0.55	1260
accuracy			0.58	3000
macro avg	0.58	0.58	0.58	3000
weighted avg	0.59	0.58	0.58	3000

همانطور که از تصاویر تغییرات خطا در حین آموزش مشخص است؛ خطا کاملا در هر سه مدل در حال نوسان است و جز چند iteration اولیه، که کاهش جزئی‌ای در مقدار loss مشاهده می‌شود؛ تغییر دیگری به‌چشم نمی‌آید. همینطور معیارهای مختلف که بر روی دادگان تست اندازه‌گیری شده‌اند، در هر سه مدل شباهت خیلی زیادی به یکدیگر دارند (برای مقایسه‌ی بهتر، باید میانگین عملکرد یک مدل گزارش شود؛ اما به علت محدودیت زمان اجرا و GPU متاسفانه امکان این کار نبود و این مقایسه‌ها، مقایسه‌ی دقیقی نیستند). در مجموع و با این توضیحات، به نظر می‌رسد که افزایش تعداد ایپاک، تغییری در عملکرد مدل پدید نمی‌آورد. چرا که وزن‌های قسمت XLNet کاملا فیکس هستند و تنها نیاز است که وزن‌های لایه‌ی feed forward به‌روزرسانی شوند و برای این کار اصلا نیازی به تعداد ایپاک بالا نیست!

Bert:

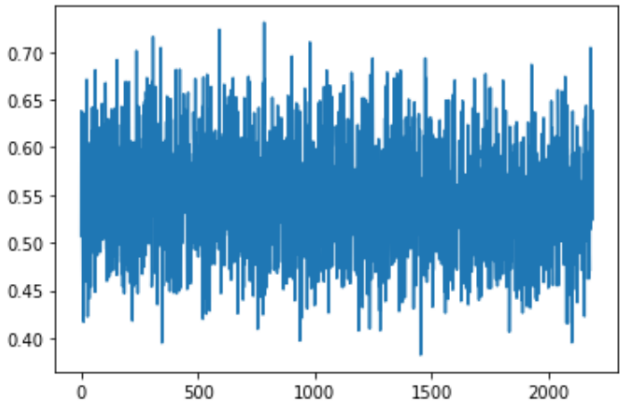
تصویر زیر نمودار تغییرات loss در طول ۴ ایپاک (و با طول batch برابر ۳۲ می‌باشد)



و عملکرد مدل بر روی داده‌های تست در زیر گزارش شده است.

	precision	recall	f1-score	support
0	0.77	0.44	0.56	1740
1	0.51	0.82	0.63	1260
accuracy			0.60	3000
macro avg	0.64	0.63	0.60	3000
weighted avg	0.66	0.60	0.59	3000

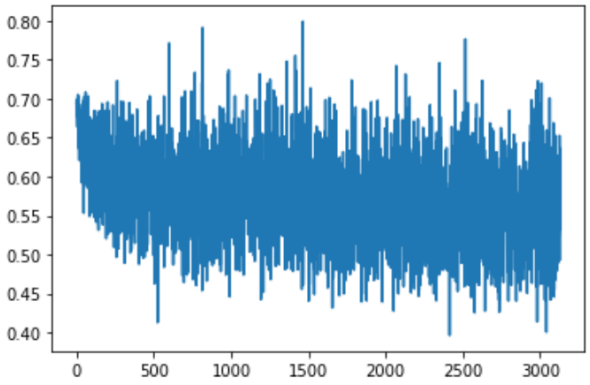
تغییرات تابع خطا برای ۷ ایپاک هم تغییر چندانی نکرده است.



و گزارش عملکرد روی دادگان تست هم به این صورت است.

	precision	recall	f1-score	support
0	0.76	0.48	0.59	1740
1	0.52	0.79	0.63	1260
accuracy			0.61	3000
macro avg	0.64	0.63	0.61	3000
weighted avg	0.66	0.61	0.60	3000

نتیجه‌ی تعداد ایپاک ۱۰ در تغییرات خطا در تصویر زیر قابل مشاهده است.



و به این صورت بر روی دادگان تست عمل کرده است.

	precision	recall	f1-score	support
0	0.78	0.44	0.56	1740
1	0.52	0.83	0.64	1260
accuracy			0.60	3000
macro avg	0.65	0.63	0.60	3000
weighted avg	0.67	0.60	0.59	3000

در مورد مدل bert هم تا حد زیادی توضیحات قبلی صادق است. افزایش تعداد ایپاک باعث کاهش خطا و افزایش دقت نشده و ما شاهد نوسان‌های مداوم در خطا هستیم. همچنین معیارهای مختلف بر روی دادگان تست، در هر سه مدل بسیار شبیه به یکدیگر هستند. در نتیجه در این معماری هم نیازی به افزایش تعداد ایپاک‌ها دیده نمی‌شود.

اما در مورد مقایسه‌ی استفاده از XLNet و Bert؛ از نظر معیار accuracy، مدل پیاده‌سازی‌شده با کمک Bert ظاهرا عملکرد بهتری ارائه داده است (برای این استدلال هم نیاز به تکرارهای مکرر آموزش این مدل‌ها داریم). اما در معیار recall برای داده‌هایی که تنفرآمیز هستند (کلاس ۱) رشد قابل توجهی در مدل bert نسبت به مدل xlnet مشاهده می‌شود (اختلاف حدود ۲۰ درصدی) همچنین رشد ۱۰ درصدی در معیار precision کلاس داده‌های غیرتنفر آمیز هم برتری دیگر مدل Bert نسبت به XLNet است.

البته معیار f1 در دو مدل تفاوت خیلی زیادی با یکدیگر ندارد.

سوال ۳

برتری هر یک از این معیارها بر دیگری، بستگی به کاربرد مدل ما دارد. در اینجا تشخیص هر چه بیشتر داده‌های تنفرآمیز مهم‌تر است. چرا که با جمع‌آوری این داده‌ها و مطالعه‌ی آن‌ها می‌توان به دلایل وجود نفرت در آن موضوع خاص پی برد؛ پس ما باید تلاش کنیم تا جای ممکن تمامی این داده‌ها را جمع‌آوری کنیم و با کمینه‌کردن تعداد توییت‌های غیرتنفرآمیزی که به عنوان تنفرآمیز تشخیص داده شده‌اند، نویز این داده را کاهش دهیم. در نتیجه recall کلاس توییت‌های تنفرآمیز اهمیت بیشتری برای ما دارد. همچنین می‌خواهیم که precision داده‌های غیرتنفرآمیز هم بالاباشد تا نویز کمی داشته‌باشیم! در نتیجه احتمالا برای تسک ما، استفاده از مدل Bert انتخاب هوشمندانه‌تری باشد.

(اگر شخص دیگری، قصد استفاده‌ی دیگری از این داده‌ها داشته‌باشد، احتمال عوض شدن پاسخ این سوال هم تبعا وجود دارد!)