**به نام خدا** 

**دوره کارآموزی هوش مصنوعی**

**پردازش زبان طبیعی**

**بخش چهارم پروژه - طبقهبندی با مدلهای آماده**

**محدثه رهنما - زینب تقوی**

**آبان الی دی 99**

****

از آنجا که هدف این فاز طبقهبندی متن با استفاده از مدلهای از پیش آموزش دیده است، بنابراین از مدلهای مبتی بر bert استفاده میشود. بدین منظور در سایت face hugging برخی مدلهای آموزش دیده با متون فارسی مورد استفاده قرار گرفتهاست. همچنین مدل چند زبانهای )lingual multi )که زبان فارسی نیز در فهرست زبانهای آموزش دیدهاش موجود است، مورد بررسی قرار گرفته است. 

در ابتدا تالش شد تا با استفاده از کتابخانهی tensorflow پیاده سازی صورت گیرد که به دلیل عدم موفقیت کتابخانهی pytorch جایگزین گردید. معماری به کار رفته بدین صورت است که در ابتدا ورودی به bert اعمال شده و در نهایت خروجی bert برای طبقهبندی به کار گرفته میشود. در واقع خروجی برت به دو الیهی تماماً متصل هدایت میشوند. واضح است که در آخرین الیه تعداد نرونها به تعداد کالسهای موردنظر است. تعداد الیهها و تعداد نرونها نیز از پارامترهایی است که با آزمون و خطا به دست میآید.

به دلیل نتایج بسیار پایین و مقادیر تک رقمی درصد معیارهای ارزیابی، معماری از جهات مختلف بررسی شد که در ادامه بحث میشود:

∙ تابع loss :به دلیل ویژگی ذاتی مسئلهی موردنظر، مبنی بر این که هر نمونه میتواند به بیش از یک کالس تعلق داشته باشد، بنابراین ضروری است تا از تابع loss مناسب استفاده شود. در pytorch برای محاسبهی زیان به روش entropy cross با استفاده از یکی از توابع زیر امکان پذیر است:

BCEWithLogitsLoss o

BCELoss o

نکتهی قابل توجه این است که تفاوت این دو روش در نوع مقادیر ورودی است. در BCEWithLogitsLoss باید مقادیر نرونهای الیهی آخر بدون اعمال تابع سیگموید به تابع داده شود. زیرا در پیاده سازی این تابع سیگموید اعمال شده و سپس entropy cross binary محاسبه میشود. در پایان هنگام تست مدل نیز باید تابع سیگموید به خروجی مدل آموزش دیده اعمال شود تا عملیات از نظر منطقی درست باشد. در مقابل در BCELOSS باید مقدار نرونهای الیهی آخر پس از اعمال سیگموید به عنوان ورودی به این تابع داده شود.

∙ بهینه ساز: واضح است که روش بهینهسازی تأثیر بسزایی در همگرایی وزنها دارد. به طوری که ممکن است برخی روشها نتنها موجب کاهش زیان )loss )نشده بلکه سبب افزایش آن شود و بنابراین



میزان loss در طول فرایند آموزش بدون تغییر باقی بماند. بدین منظور از روش stochastic .شد استفاده gradient descent(sgd)

∙ محاسبهی وزن کالسها: به دلیل عدم تعادل در نمونههای هر کالس، بهتر است تابع loss برای هر کالس با در نظر گرفتن وزن محاسبات را انجام دهد. برای محاسبهی وزن در مسائل چند کالسه، توابع آمادهای موجود است. اما از آنجا که مسئلهی موردنظر label multi است، این محاسبات بدون استفاده از تابع آماده انجام شد و وزن هر کالس به دست آمد. روش محاسبهی وزنها به این ترتیب است که برای تعداد نمونههای هر کالس بر تعداد کل نمونهها به دست آمده و وزنهای به دست آمده به تنسور تبدیل میشود. ذکر این نکته ضروری است که در این حالت باید نوع تنسورهای وزن و تنسورهای برچسبها )الیهی آخر( از نوع float باشد تا خطای محاسباتی رخ ندهد. 

∙ پیش پردازش: به دلیل مشکالت پردازش زبان فارسی، انجام پیش پردازش امری ضروری است. در میانهی کار مشخص شد که عدم انجام پیش پردازش طبق مدل آمادهی bert فارسی، یکی از مهمترین دالیل نتایج پایین در این فاز است. بنابراین با بررسی بیشتر و مشاهدهی نمونهای در گیتهابِ مدلِ parsbert ،پیش پردازش متون مشابه پردازشهای مدل آماده انجام شد. بدین منظور از کتابخانههای hazm و cleantext استفاده گردید. همین امر مقدار score-f1 را از حدود 5 درصد به حدود 82 درصد افزایش داد.

∙ نرخ یادگیری: یکی از مهمترین پارامترهایی که در همگرایی شبکه مؤثر است نرخ یادگیری است که معموالً دو استراتژی برای تنظیم آن به کار گرفته میشود: سادهترین راه این است که نرخ یادگیری در طول فرآیند یادگیری ثابت باشد و راه دیگر این است که در ابتدا نرخ یادگیری زیاد بوده )نزدیک

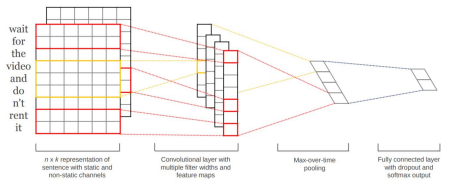
یک( و حین آموزش به تدریج کاهش یابد. بدین ترتیب نرخ یادگیری با هر دو استراتژی تنظیم شد و در نهایت راه دوم انتخاب گردید.

∙ افزودن الیهی normal batch :برای تسریع در همگرایی شبکه و یکسان کردن توزیع ورودی و خروجی هر الیه، این کار انجام شد.

∙ با تغییر تعداد الیهها و افزایش آنها به سه و چهار الیه، این نتیجه حاصل شد که افزایش تعداد پارامترها بدین شیوه در بهبود نتایج تأثیر ندارد. بنابراین به نظر میرسد دو الیه برای این مسئله مناسبتر است و باید معماری از جنبههای دیگری بررسی گردد.



∙ در بررسیهای انجام شده مشخص شد که مقدار بیشتر نرونها در هنگام آموزش برابر با صفر میشوند. علت اصلی این است که اکثر مقادیر خروجی برت منفی هستند و هنگامی که مقادیر منفی از تابع فعال کنندهی relu عبور میکنند سبب صفر شدن نرونها شده و در نتیجه صفر شدن گرادیان خواهد شد. برای جلوگیری از این مشکل از تابع فعال کنندهی elu در الیههای پنهان استفاده شد. 

∙ اضافه کردن الیهی cnn :در فاز قبل شبکهی پیشنهادی در مقاله (2014, Kim (پیاده سازی شد: به دلیل این که معماری فوق در فاز قبل نتایج خوبی داشت، تالش شد تا معماری مشابه آن پیاده سازی شود. به طوری که تفاوت آن با فاز قبل در نوع ورودیهای شبکه باشد. یعنی در فاز قبل مقادیر

embedding جمالت به الیهی کانولوشن وارد میشد و حال در این مرحله خروجی bert وارد الیهی کانولوشن شود و سپس با اعمال pooling و چند الیهی تماماً متصل خروجیها به دست آیند. گرچه نتایج به دست آمده از این معماری نیز مناسب نبودهاست.

∙ به دلیل نامتعادل بودن دادههای هر کالس و ذات مسئله -که هر نمونه میتواند متعلق به بیش از یک کالس باشد- بنابراین به نظر میرسد یکی از مهمترین دالیلی که شبکه نمیتواند به نتایج

قابل قبولی دست یابد دادهها هستند. همچنین طول متن ورودی به bert نیز از پارامترهایی

است که در عملکرد معماری تأثیرگذار است. با توجه به این که bert در تعداد ورودیها

محدودیت داشته و حداکثر طول جمالت برابر با 518 است، این مقدار به عنوان طول جمالت در

نظر گرفته شد. الزم به ذکر است که طول جمالت دادهها طبق هیستوگرام زیر بیشتر از این

مقدار است:





در ادامه دادهها محدودتر شدند و این کار با این هدف انجام شد تا از صحت معماری و برنامهی نوشته شده اطمینان حاصل شود. بدین منظور تنها کالسهایی انتخاب شد که نمونههای بیشتری از آن در اختیار است و کالسهای با نمونههای کمتر کنار گذاشته شد.

با بررسی تعداد نمونههای هر کالس، با توجه به این که میانگین تعداد نمونهها تقریباً برابر با 057

نمونه است، چند بازه برای محدود سازی کالسها در نظر گرفته شد. بدیهی است که با این کار تعدادی از کالسها حذف میشوند؛ اما پیشتر بیان شد که این کار تنها با هدف اطمینان از صحت شبکه انجام میشود. بدین ترتیب سه دسته محدودیت برای کالسها اعمال شد:

1 .کالسهایی که تعداد نمونههای آنها بین 585 تا 057 نمونه باشد: با اعمال این محدودیت

53 کالس به دست آمد. در این حالت نمودارهای تابع زیان در حین آموزش به صورت زیر است. )epoch=30(

:train\_loss نمودار





: validation\_loss نمودار 



نتایج به دست آمده بر اساس تابع report\_classification:

|  |  |  |  |
| --- | --- | --- | --- |
|  | precision | recall | F1-score |
| Micro avg | %85 | %55 | %55 |
| Macro avg | %83 | %58 | %55 |
| Weighted avg | %82 | %55 | %55 |

8 .کالسهایی که تعداد نمونههای آنها بین 057 تا 1577 نمونه باشد )تعداد کالسها = 83:)

:train\_loss نمودار



:validaton\_loss نمودار



نتایج به دست آمده بر اساس تابع report\_classification:

|  |  |  |  |
| --- | --- | --- | --- |
|  | precision | recall | F1-score |
| Micro avg | %55 | %50 | %55 |
| Macro avg | %53 | %53 | %55 |
| Weighted avg | %50 | %50 | %55 |

5 .کالسهایی که تعداد نمونههای آنها بین 1185 تا 1577 نمونه باشد )تعداد کالسها=3 :)با توجه به این که نتایج به دست آمده در حالتی که کالسها نمونههایی بین 057 تا 1577 تا داشته باشند بهتر شده، تالش شد تا این محدوده کمتر در نظر گرفته شود تا نتیجهی حاصل از مجموعهی دادهی محدودتر مشاهده شود:

:train\_loss نمودار



: validation\_loss نمودار 



نتایج به دست آمده بر اساس تابع report\_classification:

|  |  |  |  |
| --- | --- | --- | --- |
|  | precision | recall | F1-score |
| Micro avg | %35 | %57 | %53 |
| Macro avg | %35 | %57 | %53 |
| Weighted avg | %35 | %57 | %55 |

متاسفانه نتایج به دست آمده در این فاز بهتر از فاز قبل نبوده و به نظر میرسد در این مسئله و با این دادهها روش scratch from نتایج بهتری داشته است. تنها با کوچک کردن مجموعه داده یافتههای به نسبت بهتری به دست آمد.

