تمرین سوم پردازش زبان طبیعی مجید ادیبیان ۴۰۰۱۳۱۰۷۸

بخش اول)

در این بخش می‌خواهیم ازمعنی کلمه ابهام زدایی کنیم. برای این کار در داده‌ها در هر جمله کلمه مبهم مشخص شده و معنی خاص آن نیز در فایل دیگر قرار گرفت است.

ابتدا فایل داده‌های آموزشی را می‌خوانیم و کلمه دارای ابهام را در نظر می‌گیریم و ده کلمه قبل و ده کلمه بعد از آن را جدا می‌کنیم و توکن‌های تعیین کننده کلمه مبهم را حذف می‌کنیم. سپس این دنباله کلمه را به BERT می‌دهیم و وکتور مربوط به هر کلمه از این دنباله را می‌گیریم و وکتور کلمه مبهم مورد نظر را از داخل آن جدا می‌کنیم. سپس بردارهای به دست آمده را با استفاده از PCA به بردارهای ۳۰۰ بعدی تبدیل می‌کنیم.

در ادامه تابعی پیاده‌سازی شده است که با گرفتن بردارهای مربوط به هر کلمه مبهم و معنی مربوط به هر بردار، آن‌ها را با استفاده از SVM دسته‌بندی می‌کند. تا مشخص شود که چه بردارهایی چه معنی دارند.

برای تست مدل ساخته شده، مانند قسمت آموزشی ابتدا بردار مربوط به هر کلمه مبهم را به دست می‌آوریم و سپس با استفاده SVM های آموزش دیده شده، آن بردارها را دسته بندی می‌کنیم تا معنی هر کلمه مشخص شود.

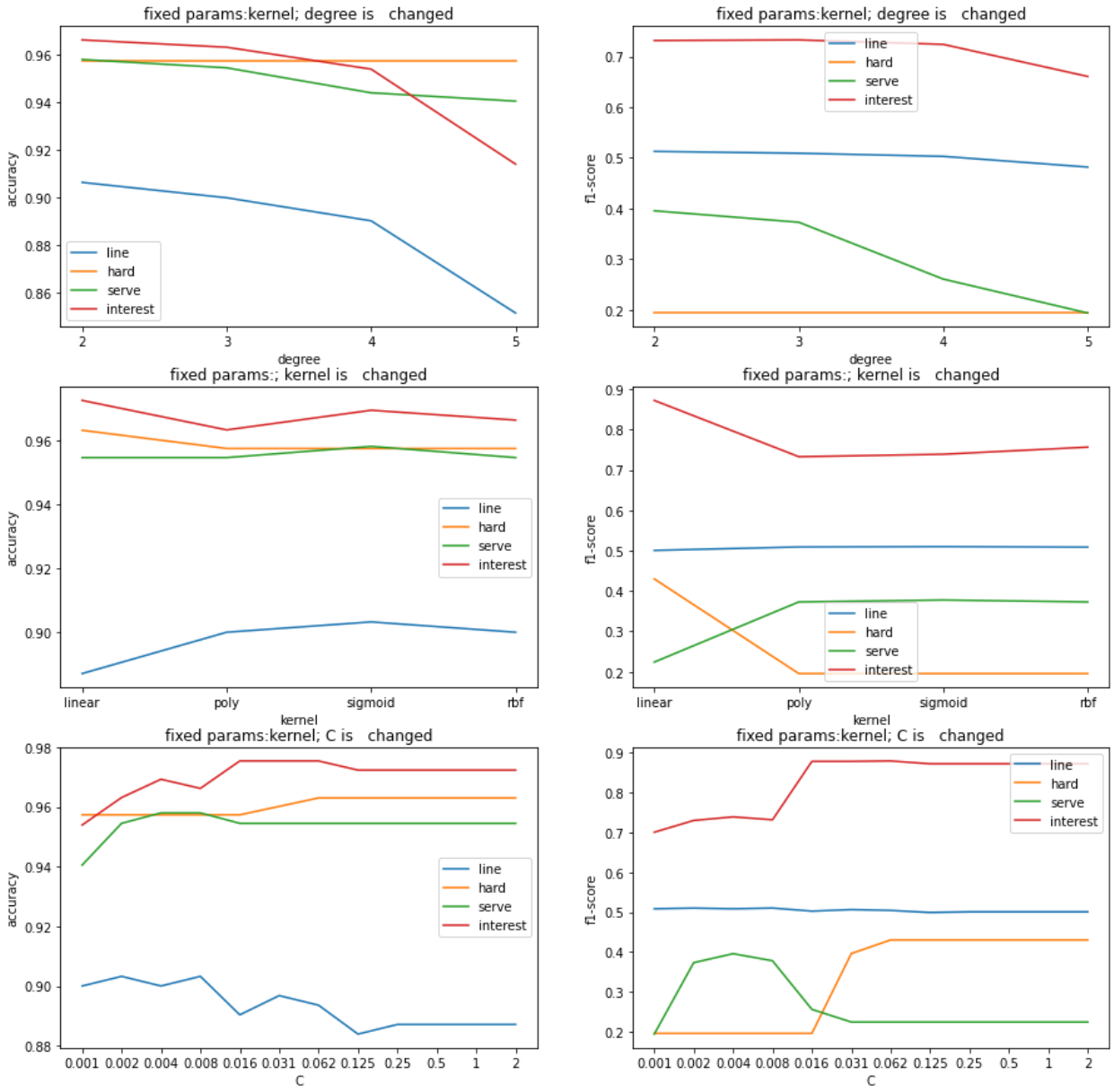
در این فرایند چند نکته در نظر گرفته شده است:

اول آنکه در بسیاری از جملات داده شده کلمه‌ای توکن‌های head را گرفته است که یعنی کلمه مبهم است ولی معنی کلمه دیگه‌ای برای آن مشخص شده است که آن کلمه هم در آن جمله حضور دارد. که برای حل این مشکل، این موارد بررسی شده و قانونی برای آن، موقع خواندن داده‌های آموزش در نظر گرفته شده است. از جمله برای کلمه hard اگر این فرایند را انجام ندهیم تنها یک کلاس خواهیم داشت.

دوم آنکه کلمات دارای ابهام ۴ حالت مختلف دارند ولی هر یک انواعی دارد. برای مثال کلمه hard به صورت‌های hard، harder، hardest وجود دارد که برای یکسان گرفتن داده‌های همه این‌ها در یک دسته‌بند، این موارد به یک مورد که ریشه است تناظر پیدا کرده اند.

یافتن بهترین پارامترهای SVM:

برای یافتن بهترین پارامترهای SVM با مقادیر مختلف پارامترهای آن مدل را آموزش می‌دهیم و سپس دقت را بر روی داده‌های validation بررسی می‌کنیم. داده‌های آموزشی دارای ۰.۸ و داده‌های validation دارای ۰.۲ از داده‌های هر برچسب هستند:



حال بهترین پارامترها را در نظر گرفته و دقت را بر روی داده‌های تست به دست می‌اوریم:

```
line:
  'accuracy': 0.9570, 'f1-score': 0.6865
hard:
  'accuracy': 0.9716, 'f1-score': 0.2892
serve:
  'accuracy': 0.9964, 'f1-score': 0.7996
interest:
  'accuracy': 0.9403, 'f1-score': 0.7978
```

همان طور که دیده می‌شود مقدار صحت برای هر یک از دسته‌بندها مناسب است و مقدار f1-score برای دسته‌بند مربوط به hard کم است که دلیل آن استفاده از macro در میانگین‌گیری بین کلاس‌های مختلف

است چون در دسته‌بند hard تعداد داده‌های کلاس‌های مختلف بسیار نامتوازن است و عدم پیش‌بینی یکی از برچسب‌های کلاس‌هایی که تعداد کمی دارند مقدار recall و در نتیجه f1 را به شدت تحت تاثیر قرار می‌دهد.

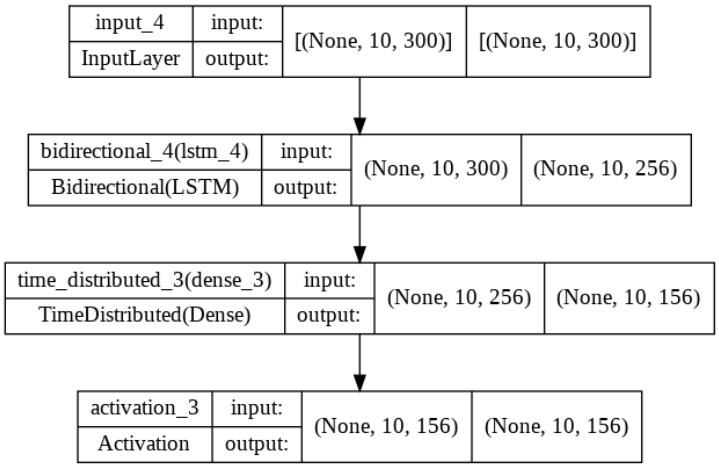
بخش دوم)

در این مسئله ابتدا داده‌ها دانلود شده اند و سپس یک پیش پردازش بر روی آن انجام شده است. تابعی نوشته شده که برچسب هر کلمه را به صورتی که در فایل سوالات توضیح داده شده میسازد و دنباله کلمات و دنباله برچسب ها را به دست می‌آورد. سپس بر روی هر دنباله کلمه یک پنجره ۱۰ کلمه‌ای را در نظر گرفته‌ایم و با گام‌های ۱۰ کلمه‌ای به جلو میلغزانیم و هر بار دنباله کلمه و دنباله برچسب آن را جدا می‌کنیم.

سپس بردار هر کلمه را به دست می‌آوریم. برای این کار از روش word2vec در کتابخانه genism استفاده کرده‌ایم. در این مرحله اگر کلمه مورد نظر در مجموعه کلمات word2vec وجود نداشت با یکسری تغییرات مانند حذف برخی علائم در آن‌ها سعی میکنیم باز هم بردار آن کلمه را بیابیم و در صورتی که نباشد برداری که برای OOV در نظر گرفته‌ایم را برای آن کلمه ذخیره می‌کنیم.

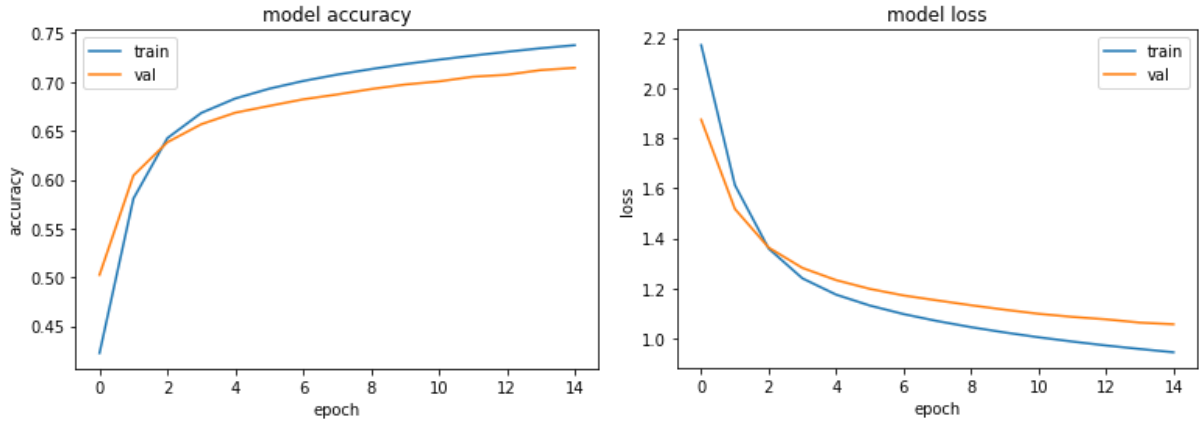
در گام بعد داده‌ها را به ماتریس‌های عددی تبدیل می‌کنیم. که برای این کار در تابع مورد نظر به جای هر کلمه بردار word2vec آن را قرار می دهیم و برای هر برچسب هم یک عدد اختصاص می دهیم.

مدل استفاده شده با استفاده از یک لایه BiLSTM است که در شکل زیر نمودار لایه‌های این مدل را می‌بینیم.



برای آموزش مدل آن را در ۱۵ ایپاک آموزش می‌دهیم که در نهایت بر روی داده‌های آموزشی و ارزیابی به ترتیب به مقدار صحت ۷۳.۶ و ۷۱.۳ رسیده‌ایم.

نمودار تغییرات loss و accuracy در طی ایپاک‌ها به صورت زیر است:



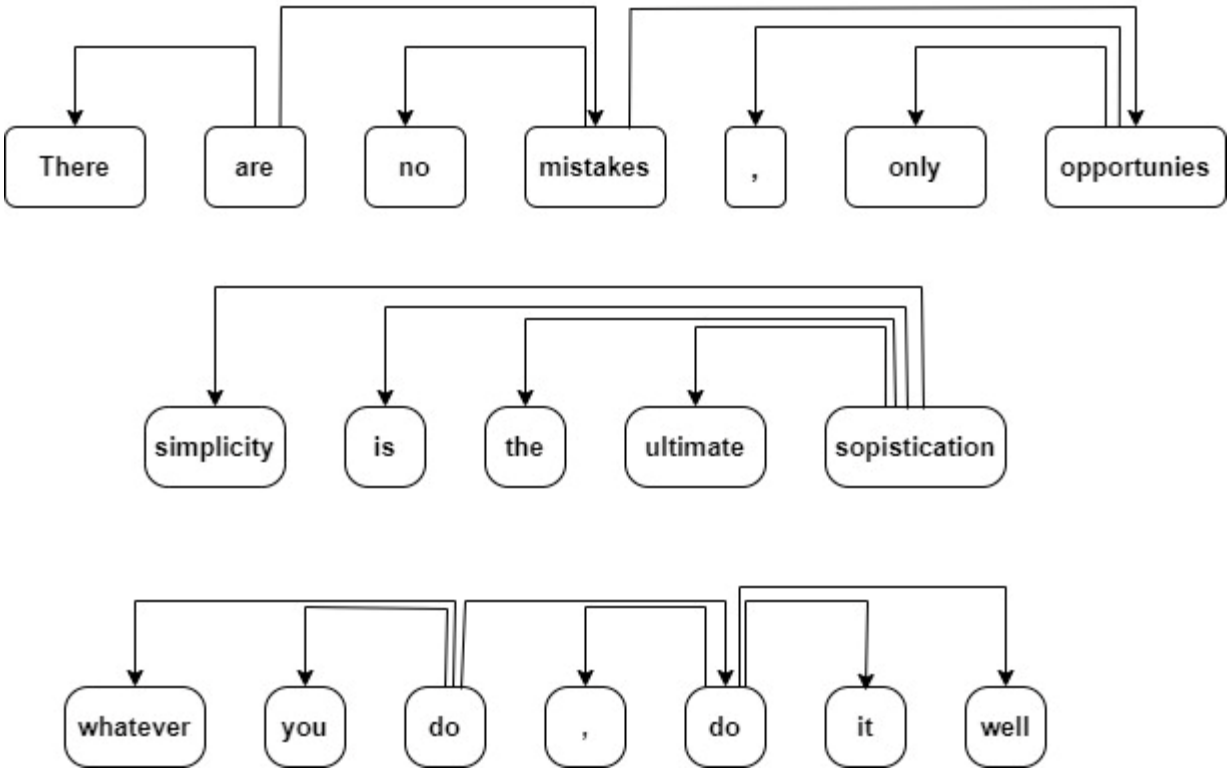
در ادامه داده‌های تست را به مدل می‌دهیم و معیارهای خواسته شده را به دست می‌آوریم. در این معیارها محاسبه precision و recall و f1-score به استفاده از روش میانگین‌گیری macro غیر منطقی است که دلیل آن وجود تعداد بسیار زیاد کلاس‌هاست که باعث می‌شود مثلا اگر یک برچسب R100 داشته باشیم و آن را تشخیص ندهیم برای آن مقدار recall برابر صفر شود که به شدت میانگین را کم می‌کند. به همین دلیل در این معیارها از روش weighted در میانگین‌گیری precision و recall و f1-score های هر کلاس استفاده می‌کنیم.
نتیجه به صورت زیر است:

```
recall: 0.7262
precision: 0.6864
f1_score: 0.7025
accuracy: 0.7262
```

نمونه جملات داده شده را به مدل داده‌ایم تا برچسب‌های آن را تشخیص دهد که خروجی به صورت زیر است:

```
There are no mistakes , only opportunities:
['1R', 'Root', '1R', '2L', '2R', '1R', '3L']
Simplicity is the ultimate sophistication:
['4R', '3R', '2R', '1R', 'Root']
Whatever you do , do it well:
['2R', '1R', 'Root', '1R', '2L', '1L', '2L']
This is a test sentence:
['4R', '3R', '2R', '1R', 'Root']
```

با رسم شکل ارتباط‌های تشخیص داده شده از روی این برچسب‌ها به شکل‌های زیر می‌رسیم:

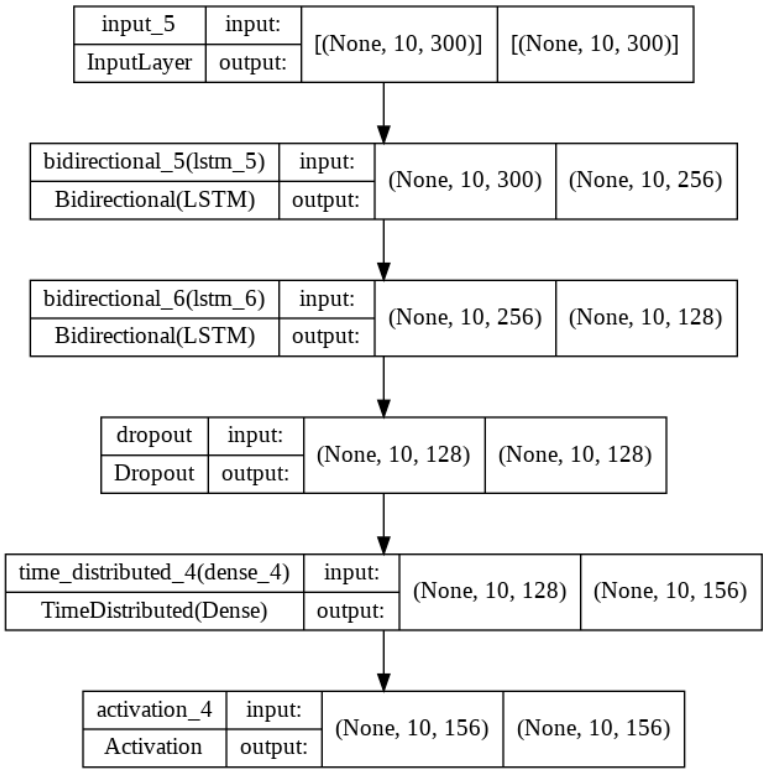


قسمت امتیازی:

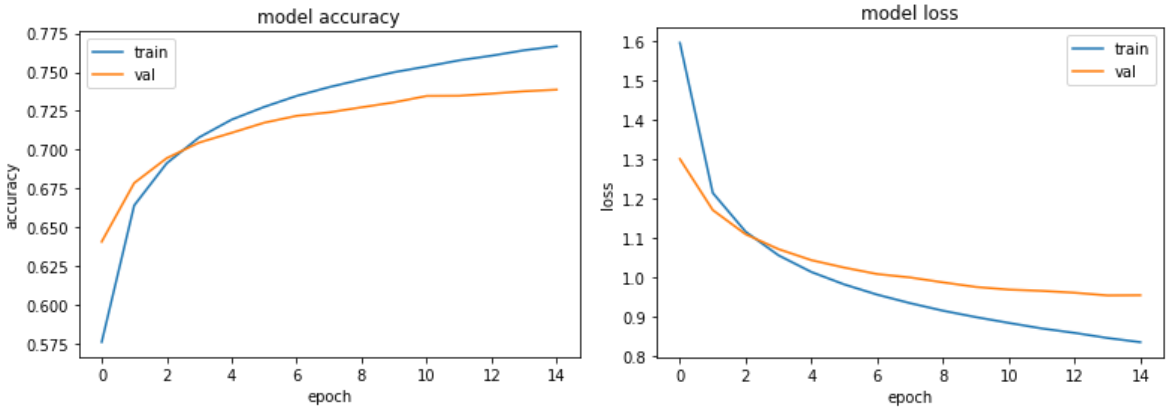
برای بهبود نتیجه مدل پیاده‌سازی شده را پیچیده‌تر می‌کنیم و برای بردارهر کلمه نیز از Glove استفاده می‌کنیم.

در ابتدا بردار هر کلمه را با استفاده از Glove موجود در کتابخانه genism به دست می‌آوریم و ذخیره می‌کنیم.

سپس مدل پیاده‌سازی شده را پیچیده‌تر می‌کنیم. که برای این کار یک لایه دیگر BiLSTM با ۶۴ بعد و همچنین یک لایه Dropout به شبکه اضافه می‌کنیم که شبکه به صورت زیر خواهد بود:



سپس مدل را بر روی داده‌های آموزشی تا ۱۵ ایپاک آموزش می‌دهیم که تغییرات loss و accuracy به صورت زیر است:



در نتیجه این تغییرات معیارهای قبلی را بر روی داده‌های تست به دست می‌وریم که به صورت زیر است:

```
recall: 0.7358
precision: 0.6977
f1_score: 0.7138
accuracy: 0.7358
```

همان طور که دیده می‌شود تمام معیارها مقداری بهبود داشته اند.