مجید ادیبیان 4001131078 تمرین دوم درس پردازش زبان طبیعی

بخش اول:

گام اول) در ابتدا داده های آموزشی از فایل train.csv خوانده شده اند و یک پیش پردازش ساده بر روی خبرهای آن انجام شده تا علائم نگارشی حذف شود و فاصله‌ها نرمال باشد. در ادامه توکن ها جدا شده و به مدل word2vec داده شده اند تا آموزش ببینند.

گام دوم) مدل TF-IDF با استفاده از کتابخانه genism و مجموعه داده های توکن‌بندی شده در قسمت قبل ساخته شده اند و سپس به دو روش بردار اسناد محاسبه می‌شود. در ابتدا از میانگین وزن دار بردار هر کلمه سند استفاده شده که وزن ها همان مقدار TF-IDF هستند و سپس از مدل Doc2vec در کتابخانه genism استفاده شده است و بر روی داده ها آموزش داده شده است.

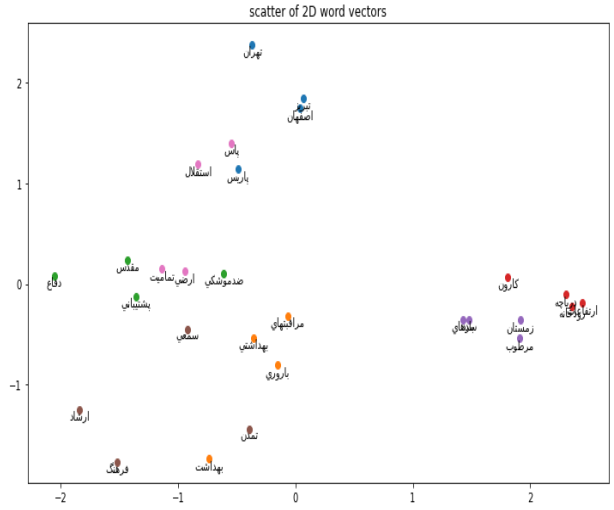
گام سوم) در این قسمت می‌خواهیم شبیه‌ترین سندها به تعدادی سند داده شده را بیابیم. برای این منظور مجموعه داده تست را می خوانیم و توابع قبلی که پیاده سازی شده بود را برای آن ها اجرا می‌کنیم تا بردار هر یک از سندها به دو روش به دست آید. سپس با استفاده از معیار کسینوسی شبیه‌ترین سند از مجموعه آموزشی را به دست می‌آوریم که نتیجه به صورت زیر است:

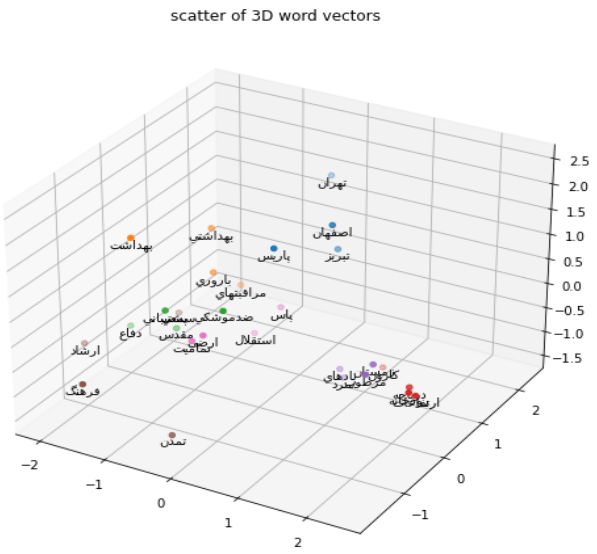
	با استفاده از TF-IDF		با استفاده از Doc2vec	
	شبیه ترین سند	فاصله کسینوسی	شبیه ترین سند	فاصله کسینوسی
Doc1	Doc165	۰.۹۸۸۵	Doc33	۰.۶۹۶۲
Doc3	Doc19	۰.۹۹۵۰	Doc19	۰.۸۷۹۸
Doc5	Doc26	۰.۹۸۸۴	Doc7	۰.۷۰۱۶
Doc25	Doc679	۰.۹۹۹۸	Doc679	۰.۹۹۱۲
Doc36	Doc667	۰.۹۸۳۷	Doc550	۰.۷۰۴۷

گام چهارم) در این قسمت باید سه شبیه‌ترین کلمه به وکتور کلمه‌های داده شده را بیابیم که برای این کار از تابع آماده خود genism استفاده می‌کنیم. که نتایج به صورت زیر است:

	اولین شبیه ترین		دومین اولین شبیه ترین		سومین شبیه ترین	
	کلمه	مقدار شباهت	کلمه	مقدار شباهت	کلمه	مقدار شباهت
استقلال	تمامیت	۰.۶۸	پاس	۰.۶۷۱	ارضی	۰.۶۶
فرهنگ	ارشاد	۰.۷۶۷	سمعی	۰.۷۰۱	تمدن	۰.۶۹۲
سرد	زمستان	۰.۷۹۳	مرطوب	۰.۷۷۸	بادهای	۰.۷۷۵
رودخانه	دریاچه	۰.۸۳۹	کارون	۰.۸۱۱	ارتفاعات	۰.۸۰۴
دفاع	مقدس	۰.۶۸۳	ضدموشکی	۰.۶۵۵	پشتیبانی	۰.۶۵۱
بهداشت	باروری	۰.۷۷۸	بهداشتی	۰.۷۳۸	مراقبتهای	۰.۷۲۶
تهران	اصفهان	۰.۶۱۵	تبریز	۰.۵۸۵	پاریس	۰.۵۸

همچنین بردار این کلمات را با استفاده از PCA به فضای دو بعدی و سه بعدی می‌بریم و سپس آن ها را به صورت نقاطی در آن فضا نشان می‌دهیم که نتیجه به صورت زیر است:





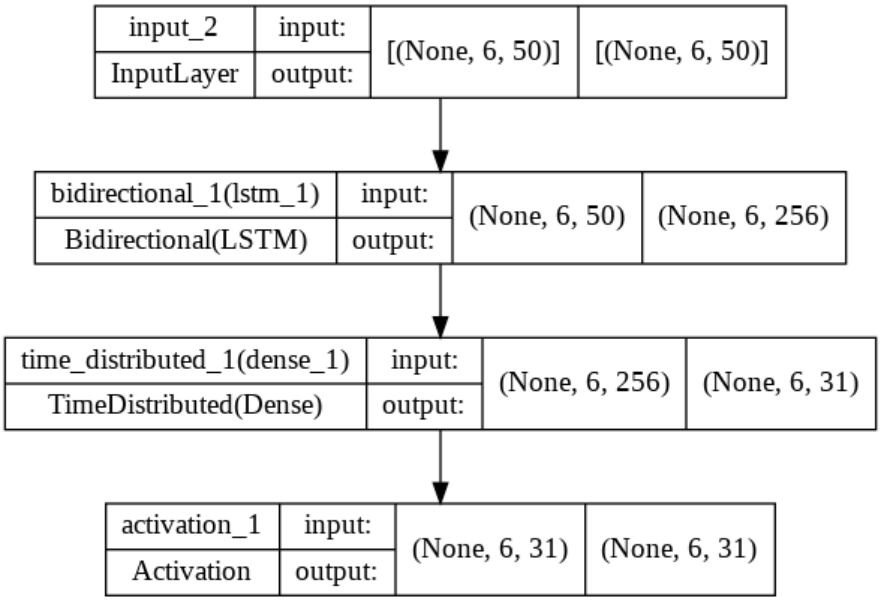
همان‌طور که دیده می‌شود نقاط مربوط به هر موضوع تقریبا به هم نزدیک هستند. به عنوان مثال نقاط آبی رنگ که کلمات مربوط به کلمه «تهران» هستند و همگی شهر هستند در نزدیک یکدیگر و دورتر از سایر نقاط قرار گرفته‌اند. علاوه بر این موضوع می‌بینیم که کلماتی از دو گروه که شباهت بیشتری دارند به هم نزدیک‌ترند. به عنوان مثال مجموعه کلمات صورتی رنگ که کلمات شبیه به "استقلال" هستند و مجموعه کلمات سبز رنگ که کلمات شبیه به "دفاع" هستند به یکدیگر نزدیک‌اند که این نزدیکی در مفهوم و معنای این کلمات نیز طبیعی به نظر می‌رسد. این رابطه در مورد کلمات قرمز که شبیه به "رودخانه" هستند و کلمات بنفش رنگ که شبیه به "سرد" هستند نیز به وضوح دیده می‌شود.

بخش دوم:

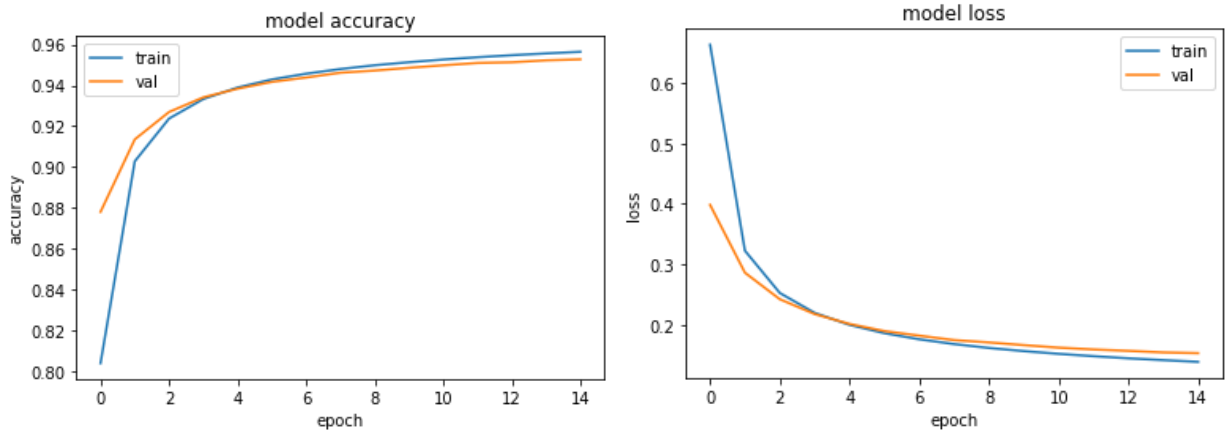
گام اول) در ابتدا داده ها در کولب دانلود شده اند و تابعی برای ساخت وکتور کلمات از روی داده متنی داده شده ساخته شده است که فایل را خوانده و وکتور را جدا می‌کند و به صورت numpy به ازای آن کلمه ذخیره می کند. سپس تابعی برای ساختن ماتریس داده ها پیاده سازی شده است که فایل های متنی را می خواند و دنباله کلمات و دنباله تگ ها را می خواند و سپس ماتریس متناسب با هر یک را می سازد. در این ماتریس ها برای کلمات از وکتوری که از فایل vectors.txt خوانده شده است استفاده شده و برای تگ ها هر تگ به یک عدد تناظر یافته است.

با توجه به آن که اندازه جملات ورودی می تواند بسیار متفاوت باشد و حتی در تست دنباله هایی با اندازه های بزرگتر ممکن است وجود داشته باشد، بهتر است دنباله کلمات و تگ مربوط به آن ها را به سگمنت‌های کوچکی بشکنیم که برای این کار با آزمایش‌های بسیار، بهترین دنباله کلمه و مقدار گام را یافته ایم که به صورت دنباله‌های ۶ کلمه‌ای و گام‌های ۲ کلمه‌ای است. به این صورت که بر روی داده های آموزش پیمایش می کنیم و دنباله های ۶ کلمه‌ای را جدا می‌کنیم و هر بار دو کلمه به جلو حرکت می‌کنیم. سپس از داده هایی که به این روش به دست آمد در آموزش مدل استفاده می‌کنیم.

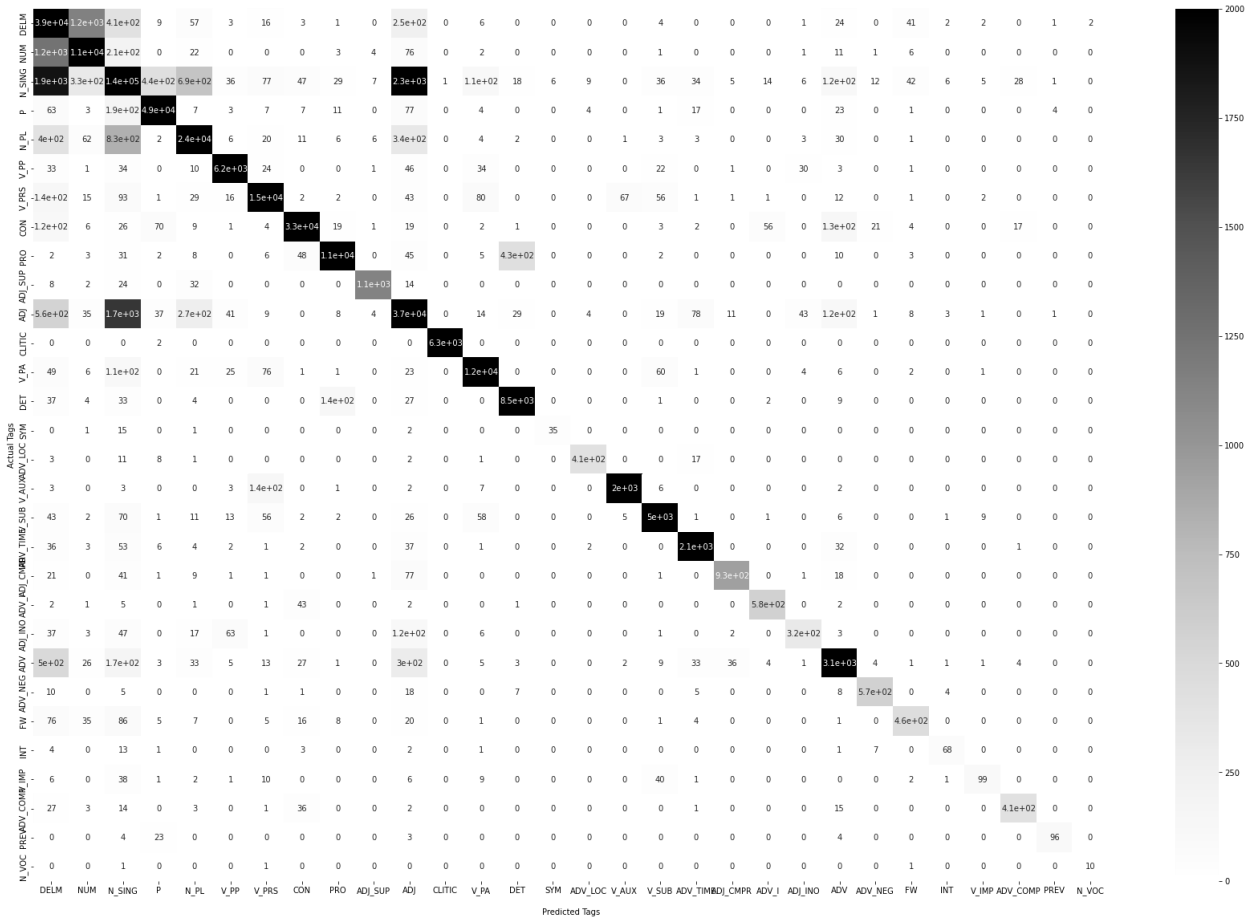
مدلی ساخته شده است که یک لایه ورودی و یک لایه BiLSTM و یک لایه dense دارد که در نهایت با تابع فعال سازی softmax کلاس هر کلمه را مشخص میکند. برای این پیاده سازی از کتابخانه Keras استفاده شده است. سپس مدل ساخته شده را بر روی داده های train آموزش می دهیم. جزئیات مدل پیاده سازی شده به صورت زیر است:



گام دوم) با استفاده از history به دست آمده از آموزش، نمودار تغییرات loss و accuracy را برای داده train و validation رسم کرده ایم که نتیجه به صورت زیر است:



جهت ارزیابی مدل هر جمله از داده های تست را به دنباله های ۶ کلمه ای جدا می کنیم و آنها را به مدل می‌دهیم تا برای هر ۶ کلمه برچسب را پیش بینی کند که برای این کار تابعی پیاده سازی شده که با گرفتن مدل و داده تست تگ‌ها را روی داده تست به دست می آورد و سپس ماتریس درهم ریختگی را به دست می آورد و سپس آن را plot می کند که در تصویر زیر نتیجه آن را می بینیم. توجه شود که در ارزیابی روی داده‌های تست برچسب مربوط به padding را در نظر نگرفته‌ایم.



حال مقدار دقت مدل با استفاده از فرمول داده شده برای داده های تست به سادگی به دست می آید که برابر مجموع اعداد روی قطر اصلی ماتریس در هم ریختگی بر کل کلمات تست است که نتیجه آن برای ماتریس درهم ریختگی بالا که از روی داده های تست به دست آمده است به صورت زیر است:

accuracy: 0.9518

همچنین محاسبه accuracy برای هر تگ معیار مناسبی به ما نمی‌دهد چرا که اگر مثلا یک تگ فقط ۱۰ مورد داشته باشه و در کل کلمات تست ما هیچ وقت این تگ را تشخیص ندهیم به دلیل عدم تشخیص آن در مواردی که آن تگ را نداشته مقدار accuracy نزدیک به ۱۰۰ خواهد بود که مشخصا اشتباه است. به همین دلیل معیارهای precision و recall را برای هر برچسب محاسبه می‌کنیم که از روی ماتریس درهم ریختگی به دست می‌آید:

	tag	precision	recall		tag	precision	recall
1	DELM	0.881	0.952	2	NUM	0.87	0.877
3	N_SING	0.97	0.956	4	P	0.988	0.991
5	N_PL	0.951	0.934	6	V_PP	0.966	0.963
7	V_PRS	0.971	0.965	8	CON	0.993	0.985
9	PRO	0.979	0.947	10	ADJ_SUP	0.979	0.934
11	ADJ	0.906	0.926	12	CLITIC	1.0	1.0
13	V_PA	0.97	0.968	14	DET	0.945	0.971
15	SYM	0.854	0.648	16	ADV_LOC	0.956	0.905
17	V_AUX	0.965	0.926	18	V_SUB	0.949	0.942
19	ADV_TIME	0.915	0.922	20	ADJ_CMPR	0.943	0.844
21	ADV_I	0.881	0.908	22	ADJ_INO	0.781	0.521
23	ADV	0.842	0.726	24	ADV_NEG	0.926	0.906
25	FW	0.801	0.634	26	INT	0.791	0.68
27	V_IMP	0.825	0.458	28	ADV_COMP	0.891	0.8
29	PREV	0.932	0.738	30	N_VOC	0.833	0.769

با دقت در ماتریس در هم ریختگی می‌بینیم برخی از داده‌ها به تعداد بیشتری اشتباه تشخیص داده شده اند برای بررسی این اشتباهات کدی زده شده است که نسبت مقادیری که اشتباه دسته بندی شده اند را بر کل نمونه های آن تگ به دست می آورد و اگر بیشتر از ۰.۱۵ باشد آن ها را چاپ می‌کند که از این طریق سه مورد زیر به دست آمده‌اند که نشان می دهد این سه مورد بیشتر از سایر موارد دچار اشتباه شده اند:

```
true tag: SYM     ,   predicted tag: N_SING
true tag: ADJ_INO ,   predicted tag: ADJ
true tag: V_IMP   ,   predicted tag: N_SING
true tag: V_IMP   ,   predicted tag: V_SUB
true tag: PREV    ,   predicted tag: P
```

همان طور که دیده می‌شود مواردی که دچار خطا شده اند بیشتر مورادی هستند که مشابه هم هستند. به عنوان مثال دو تگ V_IMP و V_SUB را در این لیست می‌بینیم که هر دو فعل هستند یا هر دوی ADJ و ADJ_INO هر دو صفت هستند. تگ PREV و تگ P نیز کلمات مشترک زیادی مانند «در، سر» دارند که باعث خطا در آن شده است. دلیل اشتباه تگ SYM آن است که این تگ برای کلمات انگلیسی استفاده شده است که تعداد بسیار کمی در داده ها داشته اند و این امر باعث شده تعداد کمی خطا در آن باعث درصد خطای بالایی باشد.