یادگیری

ماشین

پروژه

محمد

ابول نژادیان

98103867

دکتر

ابوالفضل مطهری

بهار ۱۴۰۲

فهرست عناوین

مقدمه

ابتدا با عرض پوزش بابت طولانی شدن گزارش؛ امکان آوردن تمامی ابعاد پروژه و پاسخ به سوالات آورده شده در توضیحات پروژه بدون آوردن این توضیحات نبود. به منظور اینکه مصححان محترم راحت‌تر به بخش‌های مورد نظر خود دسترسی داشته باشند، می‌توانند از فهرست عناوین استفاده کنند.

در این پروژه قصد داریم تا با استفاده از چندین روش که در طول ترم با آن‌ها آشنا شده ایم، تحلیل احساسات را روی تعداد عبارت لیبل‌گذاری شده انجام دهیم.

با توجه به این‌که داده‌های ورودی به صورت متن خام هستند، نمی‌توانیم در ابتدا روی همین متن‌ها مدل‌هایی که می‌خواهیم را آموزش دهیم و نیاز داریم تا از آن‌ها ویژگی‌های مدنظرمان را استخراج کنیم[1]. همینطور عباراتی که داده شده‌اند، به اصلاح کثیف هستند و نیاز داریم تا پیش‌پردازش[2]های مدنظر مثل حذف حروف اضافی[3]، حذف علامت‌های نگارشی، ریشه‌یابی از کلمات و... را انجام دهیم تا ناخالصی‌های موجود در داده ورودی از بین برود.

به همین منظور، در بخش‌های آینده به طور کامل هر کدام از مراحل طی شده در انجام پروژه توضیح داده شده اند. همچنین می‌توانید از طریق این لینک به Jupyter Notebook این پروژه در Google Colab دسترسی پیدا کنید.

پکیج‌های استفاده شده

به طور کلی، از کتابخانه‌های زیر برای پیاده سازی پروژه استفاده شدند، اما در هرکدام از این کتابخانه‌ها toolboxهای متعددی وجود دارند که سعی شده است ذیل هر کدام از کتابخانه‌ها، همچنین به toolboxهایی که از هر کدام از آن‌ها استفاده شده است نیز اشاره شود و دلیل استفاده از آن‌ها نیز بیان شود.

- **Numpy**: این کتابخانه یکی از پرکاربردترین کتابخانه‌ها برای موارد استفاده یادگیری ماشین است. در این پروژه به منظور استفاده از آرایه‌ها و operation هایی که می‌توان روی این آرایه‌ها

---

[1] Feature Extraction

[2] Preprocessing

[3] Stopwords

(مثل میانگین‌گیری، جمع و...) با توابع از پیش آماده آن انجام داد، از این کتابخانه استفاده کردیم.

- **Pandas**: این کتابخانه به همراه کتابخانه Numpy، نیز جزو همیشگی پروژه‌های یادگیری ماشین است. در این پروژه از این کتابخانه به منظور وارد کردن داده csv، نمایش[4] داده‌ها و انجام عملیات‌های به خصوص گروهی (groupby) از این کتابخانه استفاده کرده‌ایم.
- **Tqdm**: از این پکیج برای نشان دادن روند پیشرفت هر کدام از مدل‌ها (یا در پیش‌پردازش) استفاده شده است. کاربرد دیگری به جز در زمان آموزش و برای نشان دادن progress bar ندارد.
- **Matplotlib**: از این کتابخانه نیز به منظور نمایش نمودارها به منظور مقایسه ورودی‌ها، نتایج مدل‌ها و ... استفاده می‌شود.
- **Nltk**: این کتابخانه مهم‌ترین کتابخانه مورد استفاده در زمان پیش‌پردازش متن است. از این کتابخانه در جهت موارد پیش‌پردازشی ای که در بخشی به همین نام آمده است استفاده می‌شود. می‌توانید تمامی پیش‌پردازش‌هایی که روی داده انجام می‌دهیم را در بخش پیش پردازش داده‌ها مشاهده کنید. تمامی این مراحل با استفاده از کتابخانه NLTK انجام شده‌است.
- **Imblearn**: یکی دیگر از خواسته‌های پروژه این است تا در آموزش، مشکل نامتعادل بودن دسته‌ها[5] رفع شود. برای این موضوع، از این کتابخانه و کلاس RandomUnderSampler استفاده می‌کنیم. منطق این روش (Under sampling) و همچنین نتایج این بخش را می‌توانید در بخش خواندن و پیش‌پردازش داده‌ها مشاهده نمایید.
- **Gensim**: از این کتابخانه فقط برای استخراج ویژگی‌های کلمات با استفاده از Word2Vec در رویکرد ۲ استفاده می‌کنیم. این کتابخانه قابلیت‌های دیگری نیز دارد، اما ما فقط برای استخراج ویژگی‌های کلمات با روش‌های CBOW و Skip gram استفاده کرده‌ایم که هر کدام از این روش‌ها به تفصیل در ادامه توضیح داده شده‌اند.

---

[4] Representation

[5] Class Imblance

- **Sklearn**: این کتابخانه اصلی‌ترین و پراستفاده‌ترین کتابخانه در طول پروژه است. از این کتابخانه، toolkitهای زیر برای موارد کاربردی که تک به تک توضیح داده شده‌اند استفاده شده است.
  - modelSelection: برای آموزش بعضی از مدل‌ها لازم داریم تا تعدادی ابرپارامتر[6] را تعیین کنیم و سپس با استفاده از آن ابرپارامترها مدل‌ها را آموزش دهیم. این ابرپارامترها با استفاده از داده validation تعیین می‌شوند و برای validation نیاز داریم تا مقادیر ممکن برای این ابرپارامتر ها را در یک ParameterGrid که این toolkit در اختیارمان قرار می‌دهد قرار دهیم. همچنین یکی دیگر از کاربردهای مهم این toolkit، جداسازی داده‌های آموزش از ارزیابی با استفاده از تابع train_test_split است که تقریبا در تمامی کاربردهای یادگیری ماشین استفاده می‌شود.
  - Feature_extraction: از این toolkit در رویکرد ۱ برای استخراج ویژگی‌های متن‌ها استفاده کرده ایم. این toolkit، قابلیت تبدیل لیستی از متن‌ها (Phrase) به Document-Term Matrix با استفاده از روش TF-IDF را فراهم می‌سازد.
  - Decomposition: در بخشی لازم شده است تا برای نمایش بهتر نزدیک بودن بردار کلمات نزدیک به هم از نظر معنایی، از روش PCA استفاده کنیم و کاهش بعد بزنیم. این toolkit به همین دلیل استفاده شده است.
  - Models: در اصل toolkit ای به این نام در این کتابخانه وجود ندارد، اما برای کوتاه‌تر شدن گزارش، تمامی toolkit های مربوط به مدل‌ها را در این بخش جمع‌آوری کرده‌ایم. تمام مدل‌های Logistic Regression, Gaussian Naive Bayes, Random Forest, Adaboost, Support Vector Machine, Multi-layer Perceptron Classifier در این کتابخانه وجود داشته‌اند که در بخش آموزش مدل‌ها استفاده شده‌اند.

[6] Hyperparameter

- **Keras**: این کتابخانه صرفا در یکی از روش‌های آموزش مدل که دقت خوبی هم به ما نداد استفاده شده است. از این کتابخانه برای آوردن لایه‌های Dense برای یادگیری عمیق استفاده شده است.

مراحل انجام پروژه

این پروژه طبق مراحلی که در توضیحات داک پروژه آورده شده است انجام شده است. در خود Jupyter Notebook ای که در اختیار شما قرار گرفته شده‌است، سعی شده تا توضیحاتی مختصر در هر کدام از مراحل آورده شود، اما در این مستند، توضیحات کامل‌تری ارائه خواهد شد.

خواندن و پیش‌پردازش داده‌ها

در مرحله ابتدایی، مجموعه دادگانی که در اختیار قرار داده شده بود را خوانده و روی آن EDA[7] انجام داده‌ایم. این بخش در توضیحات پروژه گنجانده نشده بود و صرفا این بخش را انجام داده‌ایم تا حس بهتری نسبت به داده‌ای که با آن کار می‌کنیم به دست بیاوریم.

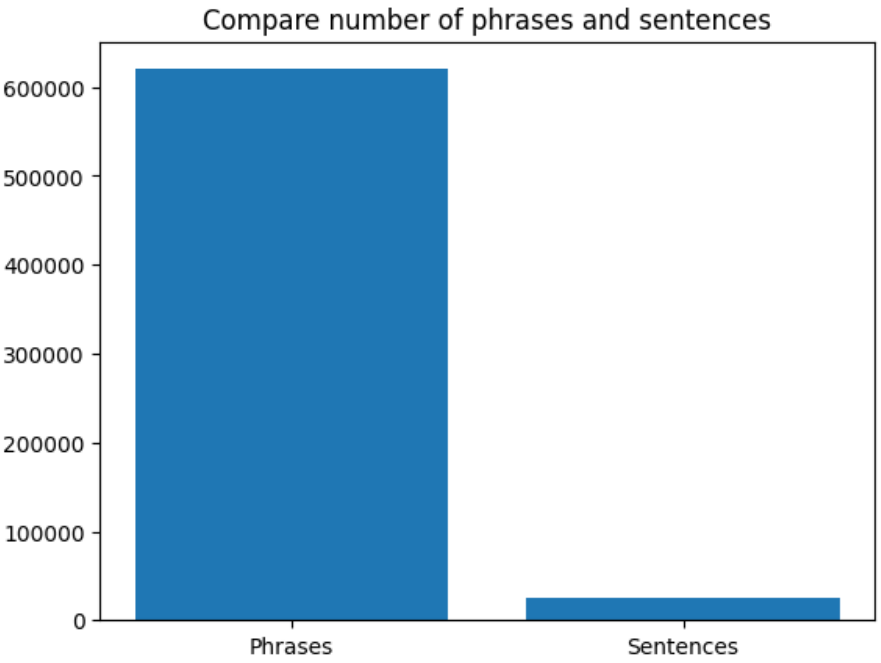
EDA

با صدا زدن تابع info روی dataframe حاوی کل دادگان، این اطلاع را به دست می‌آوریم که داده بی‌نقص است و مقدار null در دادگان موجود نیست؛ به همین علت نیازی به نگرانی درباره پرکردن این خانه‌ها نخواهیم داشت. همچنین به طور کل 155048 داده داریم.

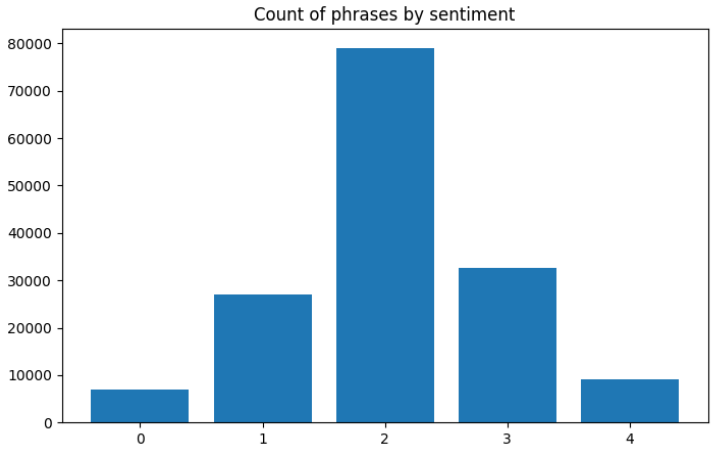
در این دادگان، دو ستون PhraseId و SenetenceId آمده است که اگر بخواهیم تعداد اعداد یکتای این دو ستون را با هم مقایسه کنیم، در اصل تعداد عبارات و جملات در دادگان را با هم مقایسه کرده‌ایم.

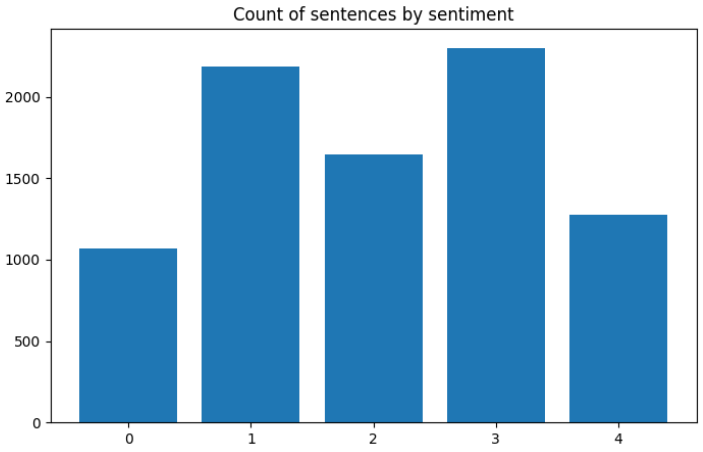
این مقایسه در جدول زیر قابل مشاهده است.

[7] Exploratory data analysis



همچنین یکی دیگر از مشکلاتی که با آن به احتمال زیاد در بخش آموزش مدل‌ها دست و پنجره نرم خواهیم کرد، نامتعادل بودن دسته‌هاست. در این پروژه، دسته‌ها در اصل همان احساسات هر کدام از عبارات است. برای مقایسه تعداد عبارات و جملات درون هر کلاس(احساس) جداول زیر را مشاهده کنید.(سمت راست نمودار مربوط به کلاس‌های جملات و سمت چپ برای عبارات)





این به ما نشان می‌دهد که برای عبارات، نامتعادل بودن در بین کلاس‌های احساسات مشهود و قابل توجه است و می‌تواند مشکل برای آموزش مدل‌ها ایجاد کند؛ به همین علت در ادامه و در مرحله پیش‌پردازش داده‌ها لازم است تا این نامتعادل بودن را حل کنیم.

جداسازی دادگان

در این بخش، داده‌ها را به سه بخش آموزش، اعتبارسنجی و ارزیابی تقسیم می‌کنیم. این کار با استفاده از تابع train_test_split انجام شده‌است.

پیش‌پردازش داده‌ها

همانطور که در ابتدا نیز مشاهده شد، دادگان کثیف هستند و نیاز داریم تا این دادگان را مرتب کنیم. کارهای زیادی برای این امر می‌توان انجام داد که تمامی کارهایی که انجام شده است در زیر آورده شده:

1. **تبدیل حروف بزرگ انگلیسی به حروف کوچک**: این کار به سادگی با تابع lower پیش‌فرض خود پایتون پیاده‌سازی شده‌است.
2. **حذف فاصله‌های بیش از یکی**: گاهی اوقات بیش از یک فاصله بین کلمات می‌افتد که چون بعدا در tokenize کردن جملات ممکن است باعث دردسر شود، این فاصله‌های اضافی را حذف کرده‌ایم.
3. **حذف stopwordها**: کتابخانه nltk با فراهم آوردن لیستی از پیش تعیین شده از stopwords، این امکان را به ما می‌دهد تا بتوانیم این حروف اضافی را از دادگان ورودیمان حذف کنیم.
4. **حذف علائم نگارشی**: با استفاده از `RegexpTokenizer` که کتابخانه nltk در اختیارمان می‌گذارد، می‌توانیم فقط کلمات را نگه داریم و دیگر قواعد نگارشی و حتی اعداد را که در تصمیم‌گیری در تحلیل احساسات به کارمان نمی‌آید را دور بریزیم.
5. **ریشه‌یابی کلمات**[8]: کتابخانه nltk با استفاده از یک دیکشنری جامع به نام wordNet که درون خود دارد، می‌تواند کلماتی که به آن داده می‌شود را به یک ریشه یکسان ببرد. برای مثال کلمات memorize, memorization, memory همگی از یک ریشه هستند که با استفاده از `WordNetLemmatizer` این کتابخانه همه این کلمات را به یک ریشه یکسان می‌برد که طبیعتا نتیجه تحلیل احساسات را می‌تواند دقیق‌تر کند. یکی از نکاتی که در اینجا حائز اهمیت است، این است که در `WordNetLemmatizer` می‌توانیم برای دقیق‌تر کردن این ریشه‌یابی،

[8] Lemmatization

نقش کلمه در جمله[9] آن را هم به این کلاس پاس بدهیم. به همین منظور این قطعه از کد در پروژه، با استفاده از pos_tagger، نقش هر کلمه را هم مشخص می‌کند.

```
6. if pos not in ['a', 'r', 'n', 'v']:
   pos='n' #for better lemmatization
   result.append(wordnet.lemmatize(token,pos))
```

Tokenization: به دلیل اینکه می‌خواهیم تا استخراج ویژگی‌ها روی کلمات انجام شود و قصد داریم تا کلمات را به embedder بدهیم، از این کتابخانه استفاده می‌کنیم تا بتوانیم این کار را انجام دهیم.

تمامی این کارها در قالب یک خطلوله[10] در تابع `process_texts_pipeline(texts, labels` آورده شده است.

متعادل‌کردن کلاس‌ها

روش under sampling یکی از ساده‌ترین راهکارهای درست کردن این مشکل است. این روش به این صورت عمل می‌کند که ابتدا کلاس با کمترین دادگان را تشخیص می‌دهد و سپس از باقی کلاس‌ها به همان تعداد sample انتخاب می‌کند. مشکل این روش این است که از تمامی داده‌ها استفاده نخواهیم کرد و عموما accuracy کمتری نسبت به دادگان بیشتر خواهیم داشت.

روش دیگر که در اینجا از آن استفاده نکرده‌ایم، over sampling است که در این روش برعکس روش قبل، کلاس با بیشترین دادگان را تشخیص می‌دهیم و سپس کلاس‌های دیگر را sampling with replacement انجام می‌دهیم. در این روش امکان overfitting وجود دارد.

حال در ادامه، با استفاده از دو رویکرد، سعی خواهیم کرد تا ویژگی‌های متن‌ها را استخراج کنیم و متن را آماده آموزش کنیم.

رویکرد ۱: روش‌های اولیه

در این رویکرد، به سادگی از کلاس TfIdfVectorizer تعبیه شده در sklearn استفاده می‌کنیم و یک document-term matrix تحویل می‌گیریم. این روش صرفا به کلمات خاص‌تر، وزن بیشتری می‌دهد و

---

[9] Part of Speech

[10] Pipeline

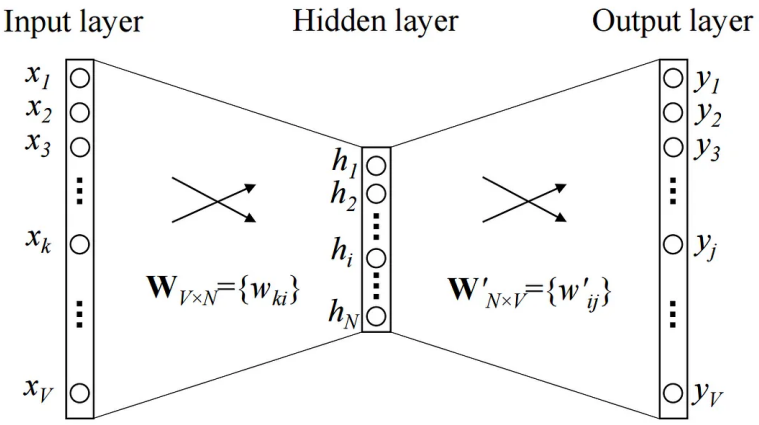
به کلماتی که بیش‌تر تکرار می‌شوند وزنی کمتر. در این روش سیاق[11] در نظر گرفته نمی‌شود و به همین دلیل کلماتی که از لحاظ معنایی به هم نزدیک‌تر هستند، عموما بردارهای شبیه به هم نخواهند داشت.

رویکرد ۲: روش‌های نوین

در روش‌های نوین، از شبکه‌های عصبی برای استخراج ویژگی‌های کلمات استفاده می‌شود. این شبکه‌های عصبی این امکان را می‌دهند که سیاقی که هر کلمه در آن استفاده شده‌است نیز در نظر گرفته‌شود. دو روش زیر را با هم بررسی می‌کنیم.

Continous Bag of Words: CBOW

در این روش، سیاق کلمه به عنوان ورودی به شبکه عصبی داده می‌شود و کلمه مقصد به عنوان خروجی درخواست می‌شود. خروجی کلی مدل وابسته به یک لایه پنهان[12] در شبکه عصبی است. وزن‌های این لایه به عنوان بردار نهایی آن کلمه داده می‌شود. شکل کلی این شبکه عصبی به صورت زیر است. (منبع)

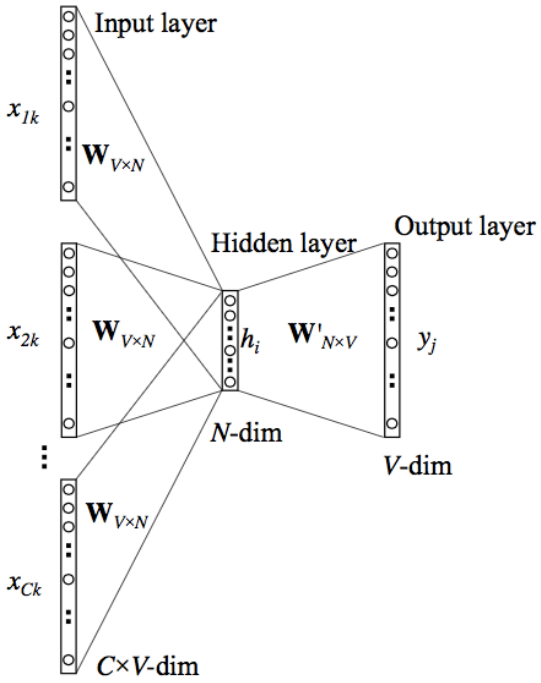


همانظور که مشاهده می‌شود، می‌توانیم یک پنجره روی جملات بگردانیم و در هر کدام از این پنجره‌ها، یک کلمه را به عنوان کلمه مقصد انتخاب کنیم و باقی کلمات را به عنوان سیاق به شبکه ورودی بدهیم. در این شبکه عصبی، طول هر بردار N خواهد بود؛ از آنجایی که لایه پنهان N گره دارد. در این شبکه صرفا یک کلمه (سیاق) به کار برده شده است و می‌توان به صورت زیر چندین سیاق را هم داشت. (منبع)

---

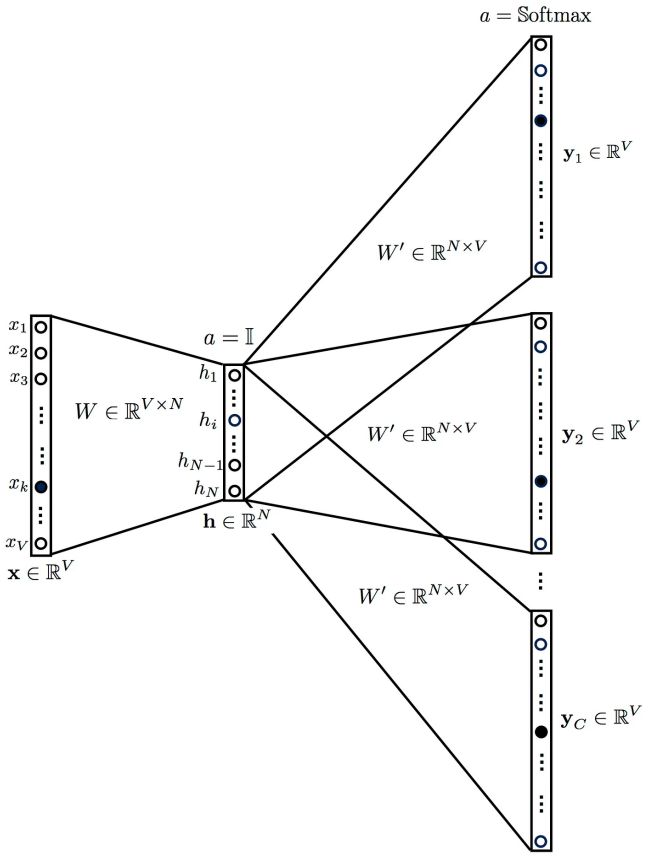
[11] Context

[12] Hidden layer



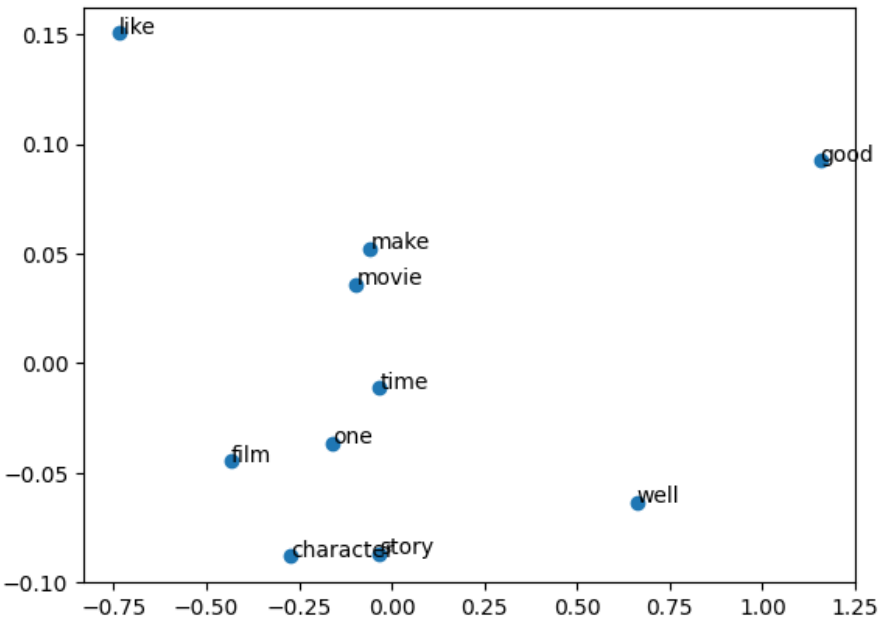
Skip gram

در این روش به صورت برعکس عمل می‌کنیم و با ورودی دادن کلمه مقصد، سعی می‌کنیم تا سیاق را خروجی بگیریم. در این روش نیز باز هم بردارهای کلمات بر اساس وزن لایه پنهان تعیین می‌شود. برای مثال می‌توان شکل زیر را نگاه کرد که شبکه عصبی‌ای را نشان می‌دهد که برای روش skip-gram طراحی شده‌است و V سیاق را بر اساس یک کلمه مقصد خروجی می‌دهد. (منبع)



مقایسه این دو رویکرد

در رویکرد اول، سیاق کلمات و در نتیجه معنی هر کدام از کلمات در نظر گرفته نمی‌شوند. اما در رویکرد دوم به دلیل اینکه سیاق کلمات در وزن‌دهی شبکه عصبی موثراند، معانی کلمات بهتر در بردار هر کدام از کلامات نهفته‌اند و انتظار داریم تا بردار کلمات با معانی نزدیک به هم، در بردار هم نزدیک به هم باشند. برای بررسی این موضوع، بعد از تمرین شبکه عصبی CBOW روی دادگان تمرین، ۱۰ کلمه اول vocabulary این مدل را با استفاده از روش PCA، کاهش بعد می‌دهیم و روی یک نمودار دو بعدی بردار این کلمات را رسم می‌کنیم. نتیجه به صورت زیر خواهد بود:



همانطور که مشاهده می‌شود، کلمات نزدیک به فیلم در نزدیکی هم هستند و کلماتی که بیشتر مثبت هستند (good, well) به سمت راست نمودار متمایل هستند.

در ادامه و در بخش مقایسه مدل‌ها، به ارزیابی نتایج تمرین یک مدل موفق روی هر دوی این روش‌ها خواهیم پرداخت.

آموزش مدل‌ها

در این مرحله، بعد از استخراج ویژگی‌های تمامی عبارات، آماده انتخاب مدل‌ها برای آموزش آن‌ها روی دادگان آموزشی هستیم. از مدل‌هایی که در طول ترم در کلاس تدریس شده‌اند استفاده شده‌است و اگر جایی نیاز به توضیح بیشتر بود، سعی می‌کنیم که توضیح دهیم. تمامی این مدل‌ها با استفاده از امبدینگ skip-gram تمرین داده‌شده‌اند. البته در انتها یکی از مدل‌های خوب (adaboost) را روی امبدینگ tf-idf هم تمرین می‌دهیم تا بتوانیم نتایج این دو را با هم مقایسه کنیم.

تعیین ابرپارامترها

برای این بخش، در هرکدام از مدل‌ها ابرپارامترهای مهم آن را در قالب یک دیکشنری قرار داده ایم و سپس تمام حالات ممکن بین این ابرپارامترها را با استفاده از دادگان تمرین، آموزش می‌دهیم و با استفاده از دادگان اعتبارسنجی تست می‌کنیم. هر کدام از این ترکیب‌ها که نتیجه بهتری بدهند، از همان ابرپارامترها استفاده می‌کنیم. یک تابع به نام

`model_hyperparameter_tuning_with_validation_set` تعریف شده‌است که با دریافت یک مدل، ترکیب ابرپارامترهای پیشنهادی و دادگان آموزش و اعتبارسنجی، کار validation را انجام می‌دهد. نحوه کار این تابع نیز به این صورت است که با پیمایش تمام ترکیب‌های بین ابرپارامترها و آموزش مدل با توجه به این ترکیب، سپس با استفاده از دادگان اعتبارسنجی آن را تست می‌کند و اگر دقت افزایش یافته بود، آن را به عنوان مجموعه ابرپارامترهای منتخب، انتخاب می‌کند.

مدل‌های استفاده شده

Logisitic Regression

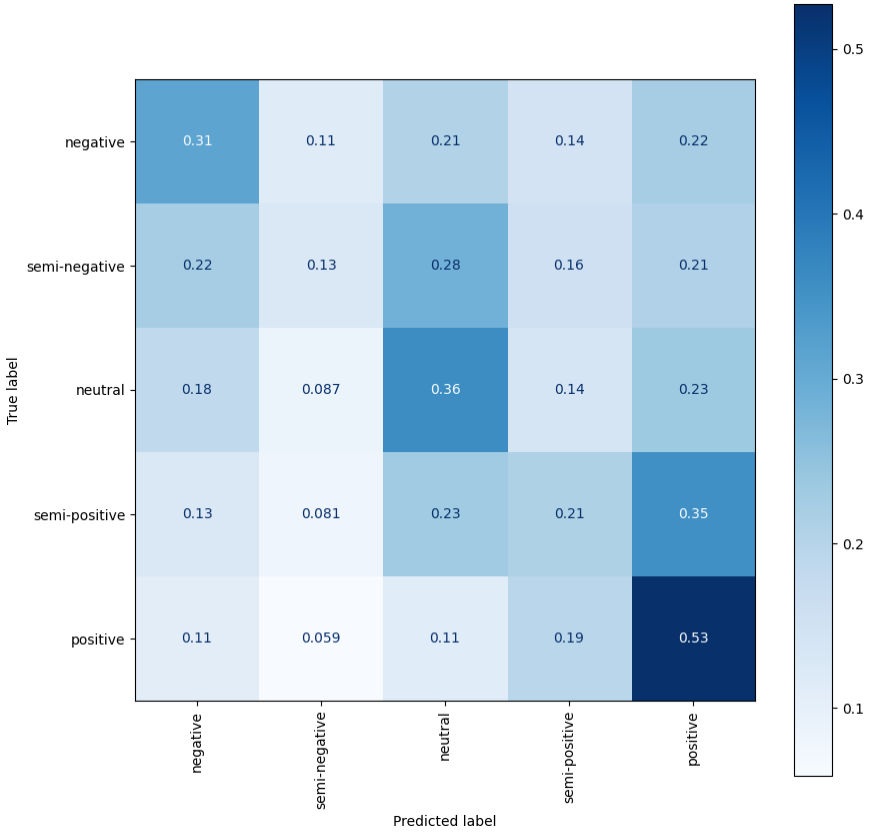
این مدل به عنوان یکی از ساده‌ترین مدل‌ها برای کلاسه بندی دادگان استفاده می‌شود. نقطه قوت این مدل سادگی آن و نقطه ضعف آن کم بودن دقت آن است.

هدف از استفاده	از این مدل به دلیل ساده بودن و خطی بودن مرزهای تصمیم استفاده شده‌است تا بررسی کنیم که این مدل ساده تا چه حد می‌تواند روی این دادگان دقیق عمل کند.
تعداد پارامترها	۲: C و tolerance - از l2 regularization به علت استفاده از lbfgs solver استفاده شده است

نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

Precision	0.27
Recall	0.31
F1-Score	0.24



همانطور که مشاهده می‌شود، نتایج این مدل رضایت‌بخش نیست.

Gaussian Naive Bayes

این مدل نیز یکی دیگر از مدل‌های ساده است که صرفا بدون در نظر گرفتن رابطه بین کلمات، با محاسبه احتمالات Prior و Posterior، دادگان را کلاسه‌بندی می‌کند. نقطه قوت این مدل ساده بودن و نقطه ضعف آن کم بودن دقت است.

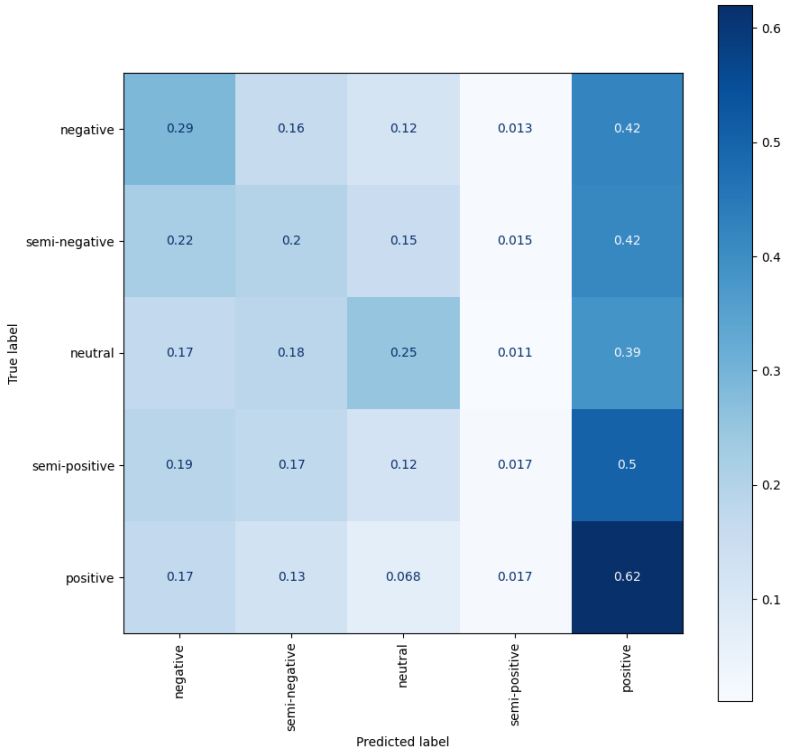
هدف از استفاده	به دلیل سادگی و سرعت در آموزش (به دلیل عدم داشتن ابرپارامتر)
تعداد پارامترها	۱: var_smoothing

نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

Precision	0.26
Recall	0.27

F1-Score	0.17



Random Forest

این مدل یکی از قوی‌ترین مدل‌های کلاسه بندی است که با استفاده از ensembling، از مجموع نظراتی تعداد درخت تصمیم ساده استفاده می‌کند. نقطه قوت این مدل در دقت بالای آن و نقطه ضعف آن حجیم بودن و کند بودن آن است.

هدف از استفاده	به دلیل قدرت مدل و تعداد زیاد ابرپارامتر برای بازی و تنظیم.
تعداد پارامترها	4: n_estimators, max_depth, min_samples_split, bootstrap البته تعداد بیشتری پارامتر در این مدل بود که به علت سنگین شدن محاسبات و عدم توان اجرایی، صرف نظر شد.

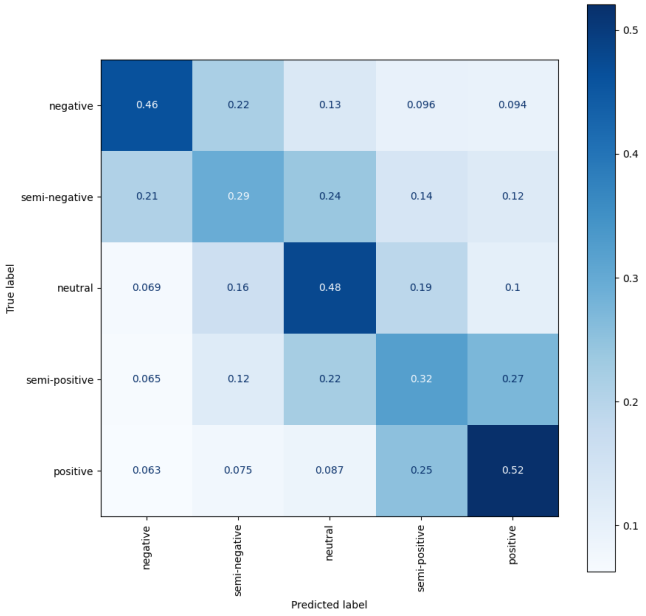
نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

Precision	0.34
Recall	0.41

F1-Score	0.35

همانطور که مشاهده می‌شود، این مدل از دو مدل دیگر بسیار قوی‌تر است.



Adaboost

این روش نیز یکی دیگر از روش‌های قدرتمند برای کلاسه بندی با استفاده از تعداد درخت تصمیم ساده است. هر کدام از درخت های تصمیم که به شکل stump هستند، به درخت تصمیم بعد کمک می‌کنند.

نقطه قوت این مدل دقت بالای آن و نقطه ضعف آن کند بودن در اعتبارسنجی است.

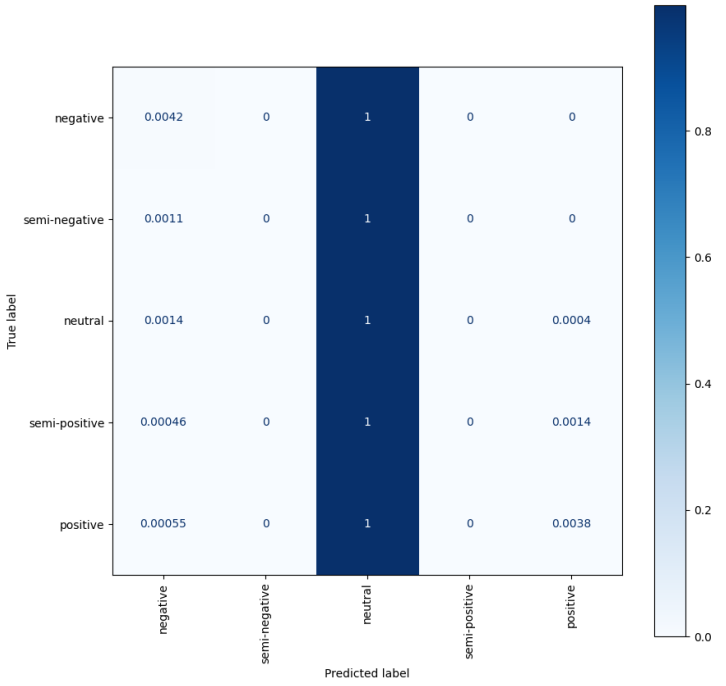
هدف از استفاده	به دلیل قدرت مدل و بیشترین دقتی که می‌دهد.
تعداد پارامترها	2: n_estimators, learning_rate

نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

Precision	0.2
Recall	0.2
F1-Score	0.14

این قوی‌ترین مدلی است که تمرین داده‌ایم.



SVM

این روش نیز یکی دیگر از روش‌های پیچیده در کلاسه بندی است. نقطه قوت این مدل دقت و کرنل‌های دقیق آن است اما نقطه ضعف آن این است که به علت حجیم بودن، نیاز به رم زیادی برای آموزش دارد و سخت‌افزارهای پایه که ما از آن استفاده می‌کنیم، امکان تمرین این مدل روی دادگان حجیم را ندارد.

هدف از استفاده	این مدل قدرت زیادی دارد اما به علت داشتن ابرپارامترهای متعدد، تمرین آن کند است. به این علت از این مدل استفاده شده است تا بتوانیم دقت خوبی بگیریم(با توجه به اینکه کلاسه بندی خوبی است) اما روی این دادگان نتیجه خوبی حاصل نشد.
تعداد پارامترها	3: C, gamma, tolerance البته می‌شد ابرپارامترهای بیشتری را تمرین داد اما رم و زمان اجازه نمیداد.

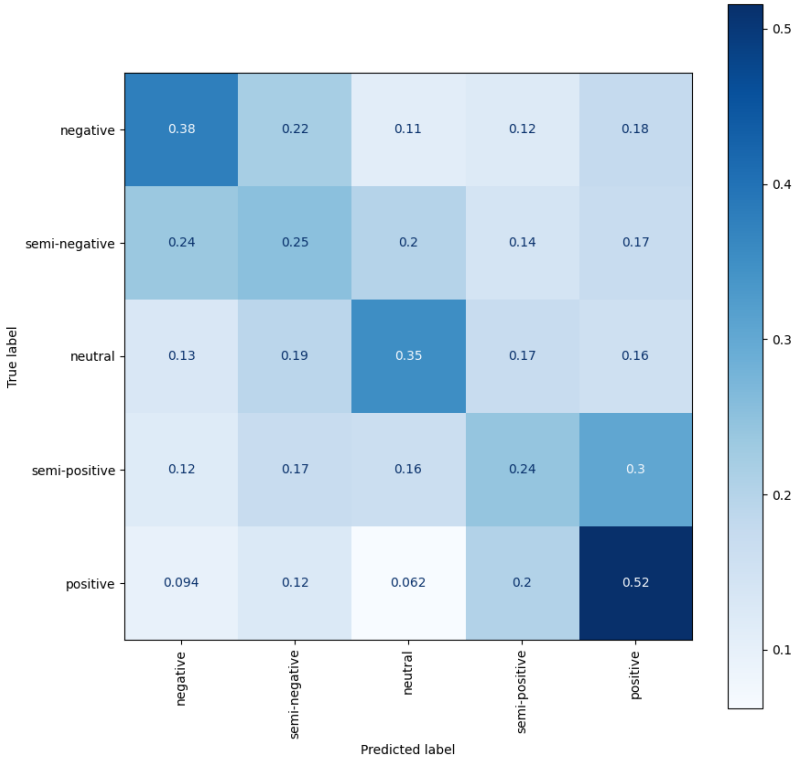
نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

Precision	0.29
Recall	0.35

F1-Score	0.27

این مدل نیز نتایج آنچنان بدی را حاصل نمی‌کند.



MLP

این روش از یادگیری عمیق و شبکه‌های عصبی استفاده می‌کند. نقطه قوت آن استفاده از یادگیری عصبی و در نظر گرفتن سیاق و به یادسپاری کلمات است. نقطه ضعف آن نیز نیاز به بررسی ابرپارامترهای زیاد و دستکاری زیاد لایه‌های نهان آن است که زمان اعتبارسنجی را زیاد می‌کند.

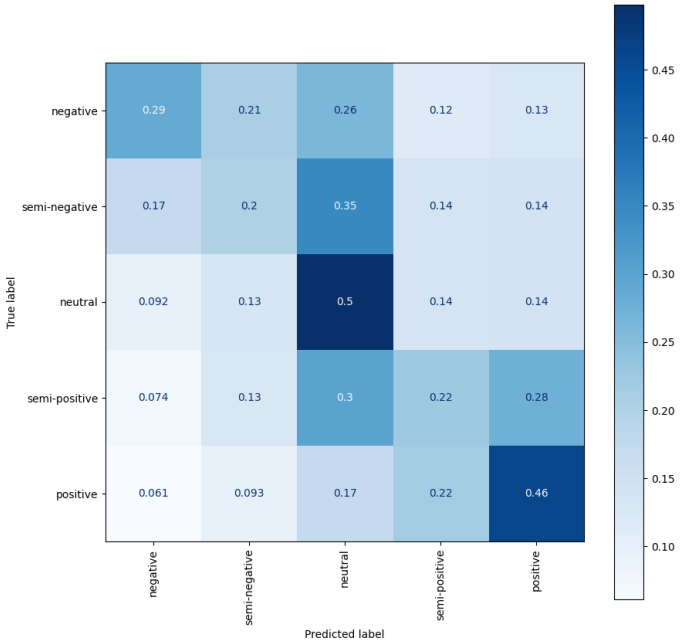
هدف از استفاده	به این دلیل از این مدل استفاده کردیم که از روش‌های نوین تر(شبکه های عصبی) هم در پروژه بهره برده باشیم.
تعداد پارامترها	3: hidden_layers_size, activation, alpha

نتایج ارزیابی مدل نیز در زیر قابل مشاهده است:

*all in macro average

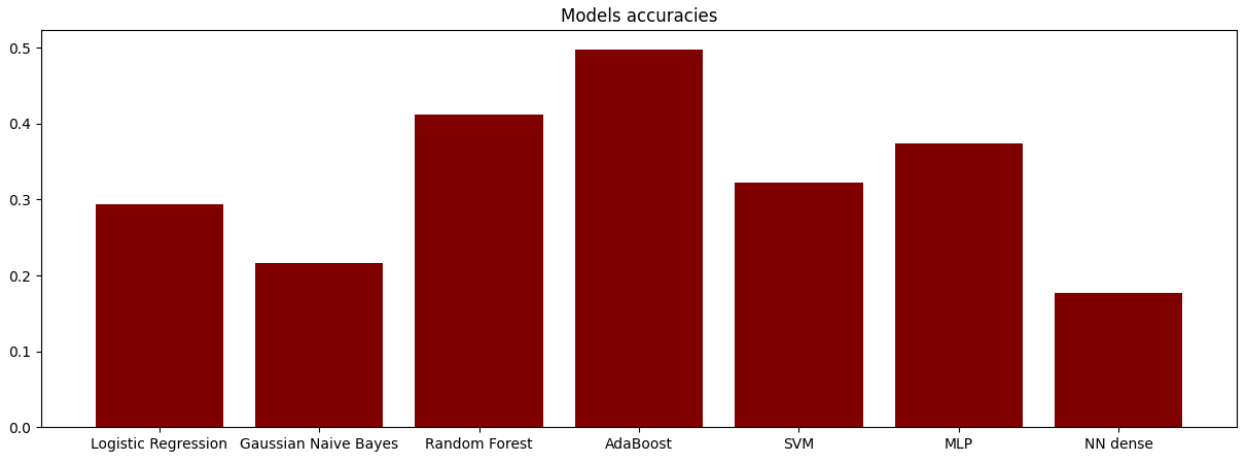
Precision	0.29

Recall	0.33
F1-Score	0.29



مقایسه مدل‌ها

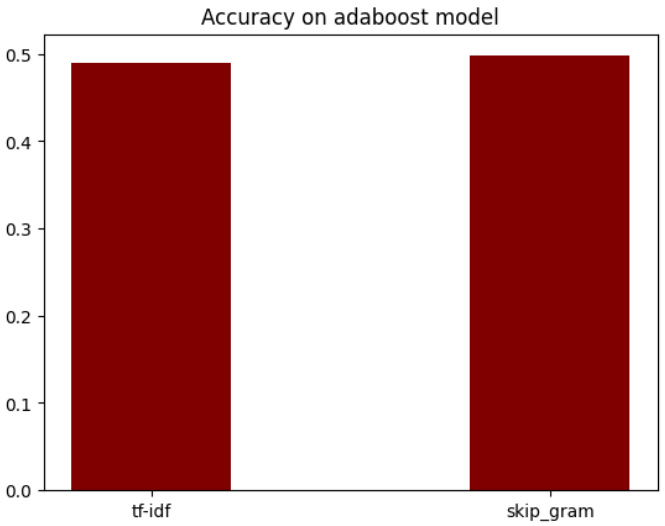
تمامی نتایج این مدل‌ها به صورت جداجدا بررسی شده اند، اما خوب است تا امتیاز دقت[13] این مدل‌ها در کنار هم نیز یک بار بررسی شود.



همانطور که مشاهده می‌شود، مدل adaboost از تمامی مدل‌های دیگر بهتر عمل کرده است.

[13] Accuracy score

همچنین یک مدل adaboost را روی امبدینگ tf-idf نیز تمرین دادیم و میتوانیم مقایسه نتایج این امبدینگ با skip-gram را مشاهده کنیم:



همانطور که مشاهده می‌شود، مدل روی امبدینگ skip-gram مقداری بهتر کار کرده است. انتظار داشتیم که این فاصله بیشتر باشد؛ اما علت اینکه این فاصله کم است در عبارات دادگان داده شده است. بسیاری از این عبارات بخشی از یک جمله که بی معنی هستند، بودند. به همین دلیل skip-gram قدرت خود را در تشخیص سیاق به خوبی نتوانست نشان دهد.

اگر از جملات کامل برای تمرین و ارزیابی این مدل استفاده می‌شد، نتایج بهترین حاصل می‌شد؛ چرا که معانی و سیاق هر جمله کامل بود و skip-gram می‌توانست بردارهای دقیق‌تری تولید کند.

تلاش‌های شکست خورده

همانطور که در مقایسه مدل‌ها نیز اشاره شد، می‌توان به مدل‌های Logistic regression و Gaussian Naive Bayes به عنوان تلاش‌های شکست خورده نگاه کرد. علت آن هم این است که در این مدل‌ها نگاهی به سیاق نمی‌شود (به خصوص در Naive Bayes) و به همین علت چون تحلیل احساسات حساس به این موضوع است، نتایج خوبی روی این دو مدل کسب نمی‌کنیم.

همچنین مدل logisitc regression نیز یک کلاسه بند ساده است که به صورت خطی کلاسه بندی را انجام می‌دهد؛ اما دادگان ما به علت پیچیدگی ای که در زبان‌های طبیعی و امدینیگی که از آن استفاده کردیم، قطعا به صورت خطی از هم جدا نمی‌شوند و نیاز است تا از روش‌های پیچیده تر برای کلاسه‌بندی آن‌ها استفاده کنیم.

نتیجه‌گیری

در این پروژه به این نتیجه رسیدیم که روش‌های امبدینگ نوین در جملات کامل و دارای سیاق، بهتر عمل خواهند کرد و می‌توانند بهتر بردارهایی برای کلمات بیرون بکشند.

همچنین مشاهده کردیم که روش‌های ساده برای کلاسه‌بندی روی زبان‌های طبیعی به خوبی کار نمی‌کنند و نیاز داریم تا از روش‌های پیشرفته‌تر مثل adaboost و SVM برای گرفتن نتایج بهتر استفاده کنیم.

در این پروژه، تحلیل احساسات به صورت label دار انجام شد اما اگر می‌خواستیم بدون label این کار را انجام دهیم، استفاده از شبکه‌های عصبی (به خصوص transformerها) می‌توانست بهترین گزینه باشد.