به نام خدا

پروژه نهایی بازیابی هوشمند اطلاعات

Question-Answer Re-ranking

محمد ناصری

۸۱۰۱۰۰۴۸۶

دی ماه ۱۴۰۰

مقدمه

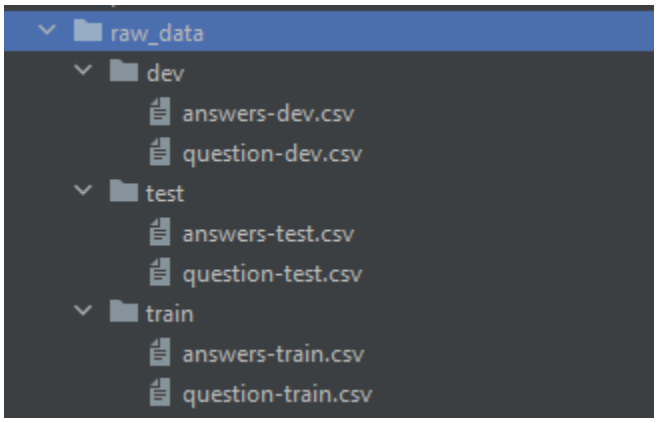
زمینه مطالعاتی پاسخ دهی پرسش به عنوان یکی از زمینه های مشترک بازیابی اطلاعات و پردازش زبان طبیعی به حساب می آید. امروزه و با گسترش منابع اطلاعاتی در دسترس، جستجو و یافتن پاسخ مناسب و صحیح، فرآیندی زمانبر و پرهزینه خواهد بود. در این راستا این امکان وجود دارد که با بهره گیری از سیستم های پرسش و پاسخ، هدف هایی همچون رتبه بندی پاسخ های مرتبط، بازیابی مناسبترین پاسخ، شناسایی پاسخهای مرتبط و موارد مشابه حاصل شوند. در این پروژه شما عمل بازیابی پاسخ را با هدف رتبه بندی پاسخ های مرتبط انجام خواهید داد. مجموعه دادهای شامل 44Kزوج پرسش-پاسخ می باشد. در این مجموعه به ازای هر پرسش ده پاسخ با برچسب مناسب وجود دارد. این مجموعه داده به سه قسمت train, dev, testتقسیم شده است که داده های موجود در مجموعه تست بدون برچسب میباشند. هدف رتبهبندی ده پاسخ متناظر با هر پرسش است به گونهای که پاسخهای با برچسب good در رتبهبندی بالاتر از پاسخهای با برچسب badو potentially usefulقرار گیرند

گام اول: پیش پردازش

برای این منظور از کتابخانه‌های NLTK و xml.Etree استفاده میکنیم.

در قدم اول هر قسمت از دیتاست را به کمک کتابخانه xml و کلاس ElementTree خوانده و parse میکنیم. سپس دیتای لازم (از جمله متون سوال و جواب و شناسه هر عبارت) را از قسمت‌های مختلف xml جمع‌آوری و در یک فایل csv برای استفاده بعدی ذخیره سازی میکنیم. نکته لازم به ذکر در این قسمت این هست که سوال‌ها و جواب‌ها در دو فایل مجزا و جداگانه ذخیره میشوند و همچنین جهت استفاده بهینه‌تر و پاسخ بهتر، برای سوال، متن topic سوال نیز به اول string سوال اضافه میشود.

برای اجرای این بخش کافیست فایل parse_xml_data.py را اجرا کرد. فایل‌های csv متناظر با دیتاست‌ها در دایرکتوری raw_data/. ذخیره میشوند.



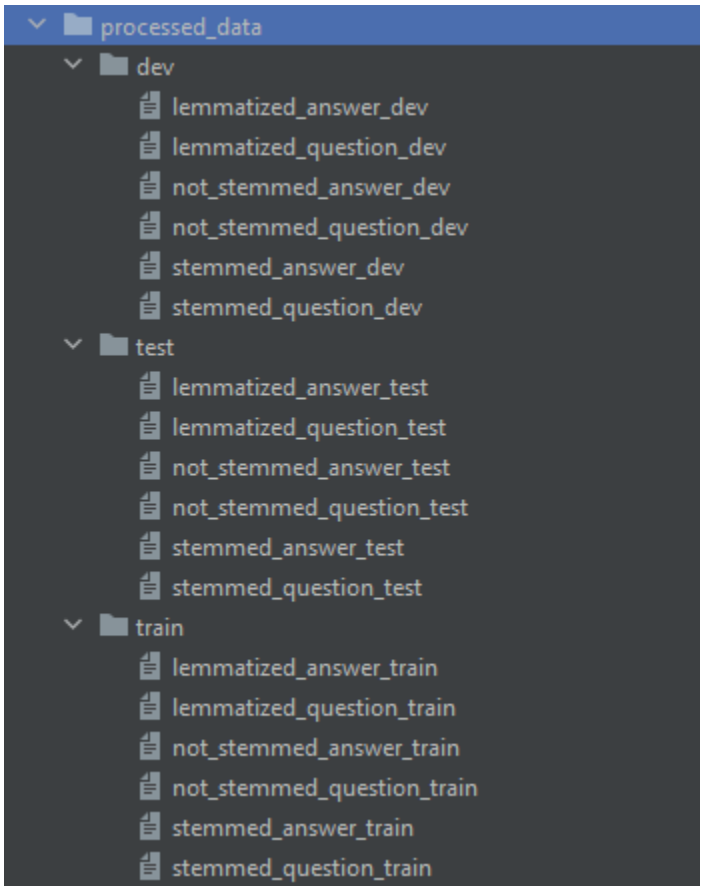
در مرحله بعد برای پیش پردازش داده خام بدست آمده از کتابخانهNLTK و RE استفاده میکنیم تا علاوه بر حذف علائم اضافی، کلمات stopword را نیز از متن داده حذف کنیم.

در این بخش پس از پیش پردازش اولیه(حذف فاصله‌ها، حذف تک کلمه‌هاو...) و حذف stopwordها، به ازای هر فایل سوال و جواب ۳ خروجی متفاوت داده میشود که عبارتند از:

1. Non_stemmed_documents که شامل سندهای stem نشده میباشند و کلمات به صورت اولیه خود حضور دارند
2. Stemmed_documents که شامل سندهای با کلمات stem شده هستند.
3. Lemmatized_documents که شامل سندهای با کلمات lemmatize شده هستند.

تولید این خروجی‌ها برای استفاده در قسمت‌ها و گام‌های مختلف پروژه میباشد.

برای اجرای این بخش کافیست تا فایل pre_process.py را اجرا کرد. فایل‌های csv متناظر با ۳ مدل خروجی هر دیتاست در دایرکتوری processed_data و زیربخش مربوط به خود ذخیره میشوند.



با خروجی‌های بدست آمده مرحله پیش‌پردازش تکمیل شده و به گام بعد میرویم.

گام دوم: استخراج ویژگی

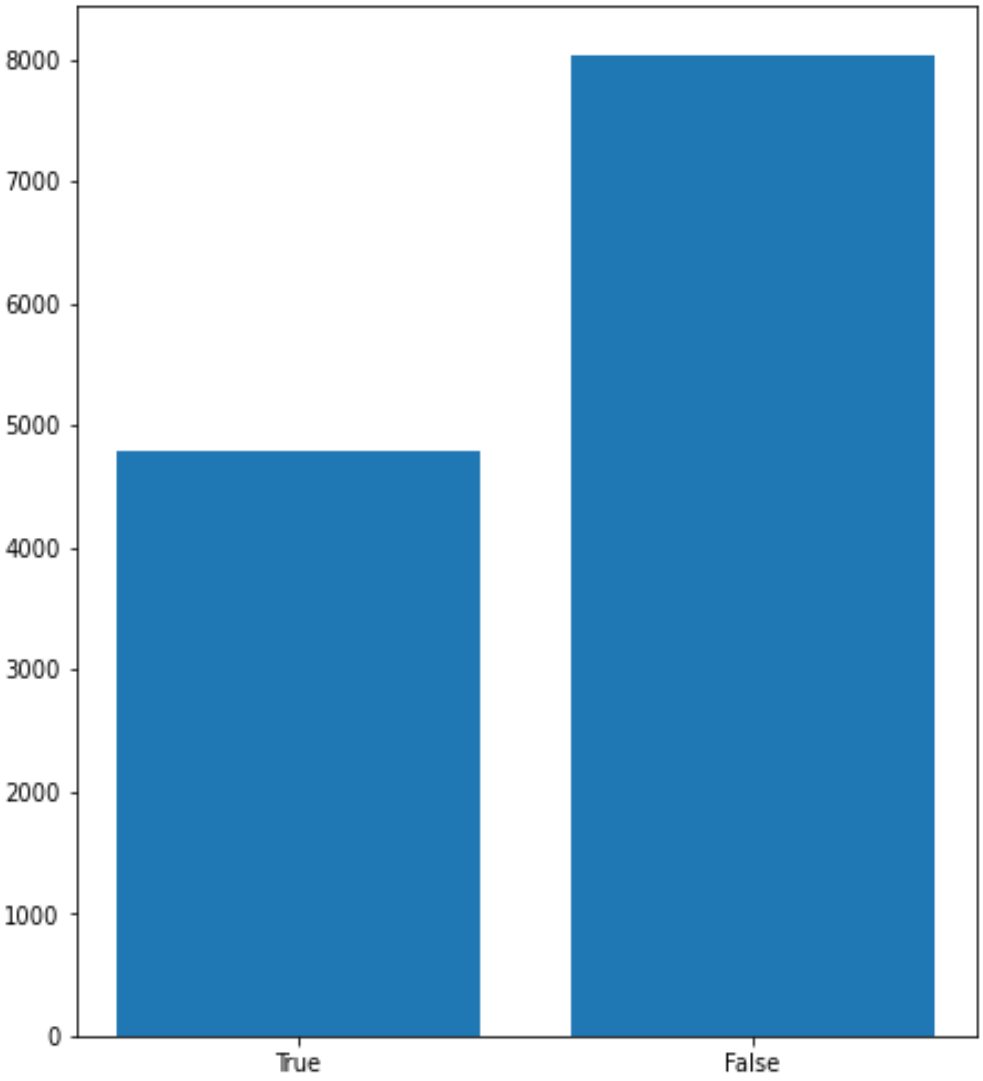
با توجه به آموخته‌های خود در کلاس درس مجموعه‌ای از ویژگیها را استخراج کنید تا به کمک آنها بتوانید یک شبکه عصبی را آموزش دهید. این ویژگیها میتوانند مبتنی بر پرسش، مبتنی بر پاسخ، مبتنی بر ارتباط بین پرسش و پاسخ و یا سایر اطلاعات موجود در مجموعه داده باشند. ویژگی های خود را به همراه دلیل انتخاب هر ویژگی بیان کرده و تاثیر آن را در عملکرد نهایی رتبه‌بندی بررسی نمایید.

برای بررسی و استخراج ویژگی‌های مورد استفاده برای شبکه عصبی، به سراغ داده‌های پردازش شده میرویم و خصوصیات سوال‌ها و پاسخ‌ها را بررسی میکنیم.

از قبل میدانیم از روش‌های بررسی شباهت پرسش و پاسخ (query, document) می‌توان به دو مورد Okapi BM25 و همچنین TFIDF-Similarity (شباهت کسینوسی) اشاره کرد.

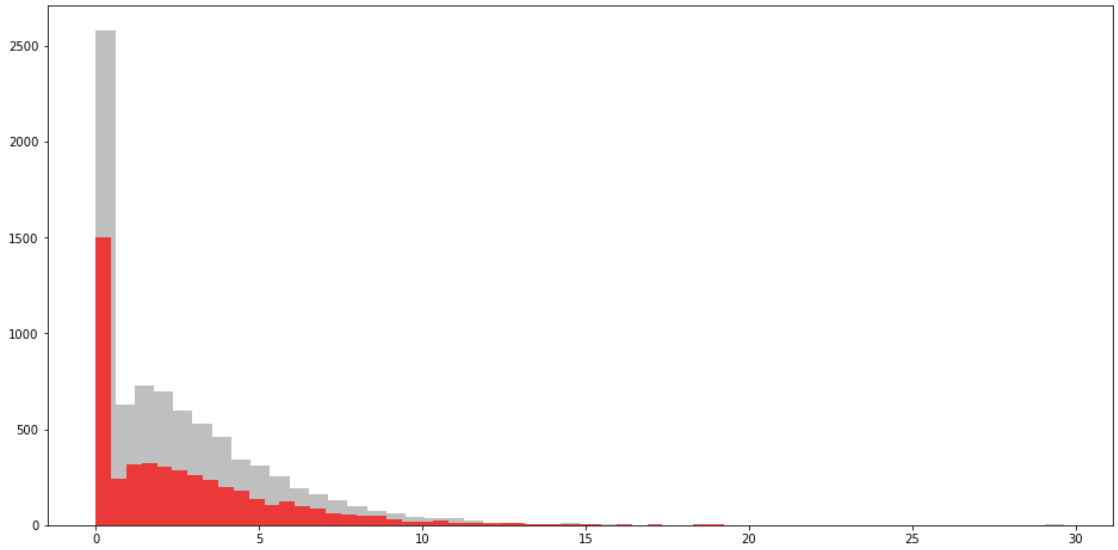
در ادامه به بررسی تعدادی از خصوصیات استخراج شده میپردازیم. این خصوصیت‌ها با توجه به توزیع آنها بر روی پاسخ‌های درست و غلط بررسی و تعریف شده‌اند. (برای این منظور یک دفترچه jupyter نیز ایجاد شده که به همراه فایل‌ها ارائه میشود)

در ابتدا برای داشتن دید کلی نسبت به داده توزیع جواب‌های درست و غلط را بررسی میکنیم(جوابهای Potentially Useful جزئی از پاسخهای غلط در نظر گرفته شده اند)



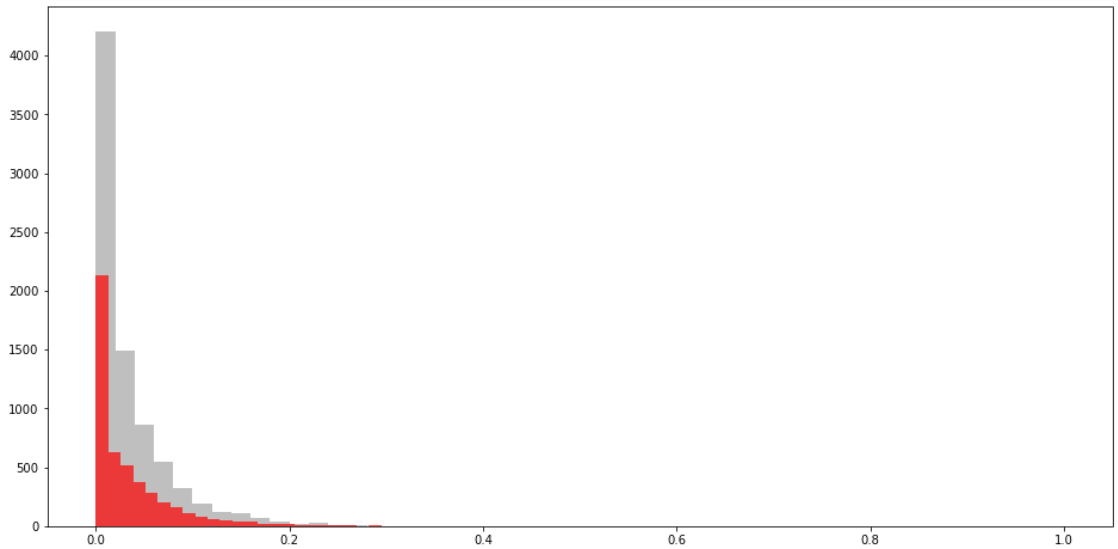
همانطور که واضح است تعداد پاسخ‌های نامربوط بسیار بیشتر از پاسخ‌های درست میباشند که نشان‌دهنده Efficient و کارا نبودن این forum پرسش و پاسخ میباشد و کاربران بیشتر پاسخ‌هایی کاملا به‌ربط به سوال ارائه داده‌اند.

Query-Document Similarity



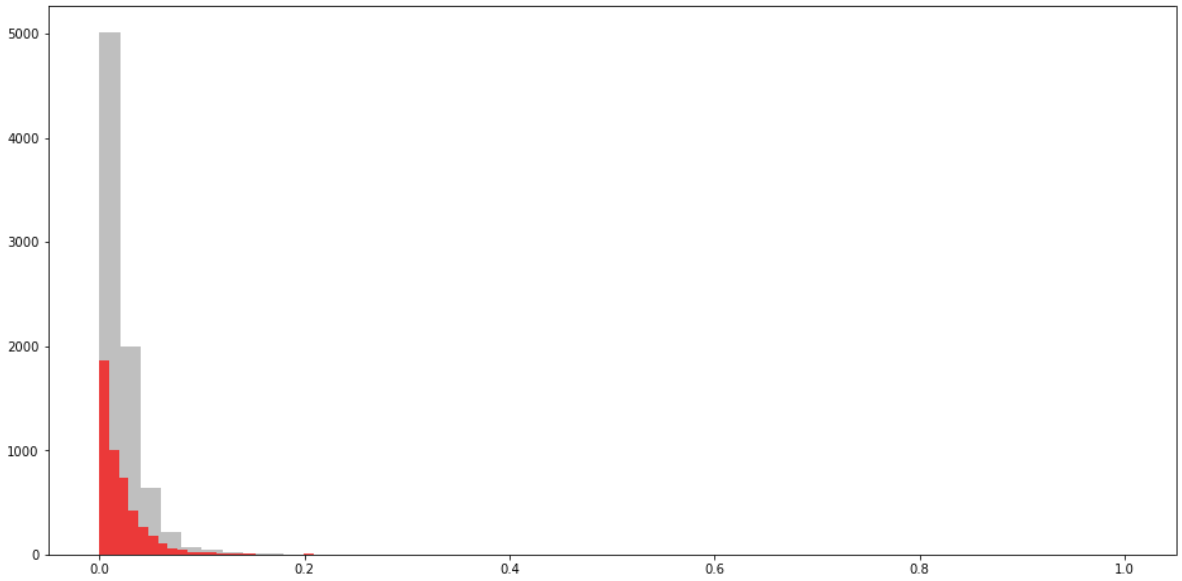
bm25-distribution1 Figure

از توزیع BM25(تصویر بالا) روی داده‌ها این نتیجه میشود که پاسخ‌های درست(رنگ قرمز) نسبت به پاسخ‌های غلط(رنگ خاکستری) شباهت و امتیاز bm25کمتری با پرسش مربوط به خود دارند. همچنین هر دو مورد توزیعی شبیه به (نه کاملا) Powerlaw دارند. این مورد برای شباهت TFIDF به وضوح بیشتری قابل مشاهده است که در زیر میبینیم:

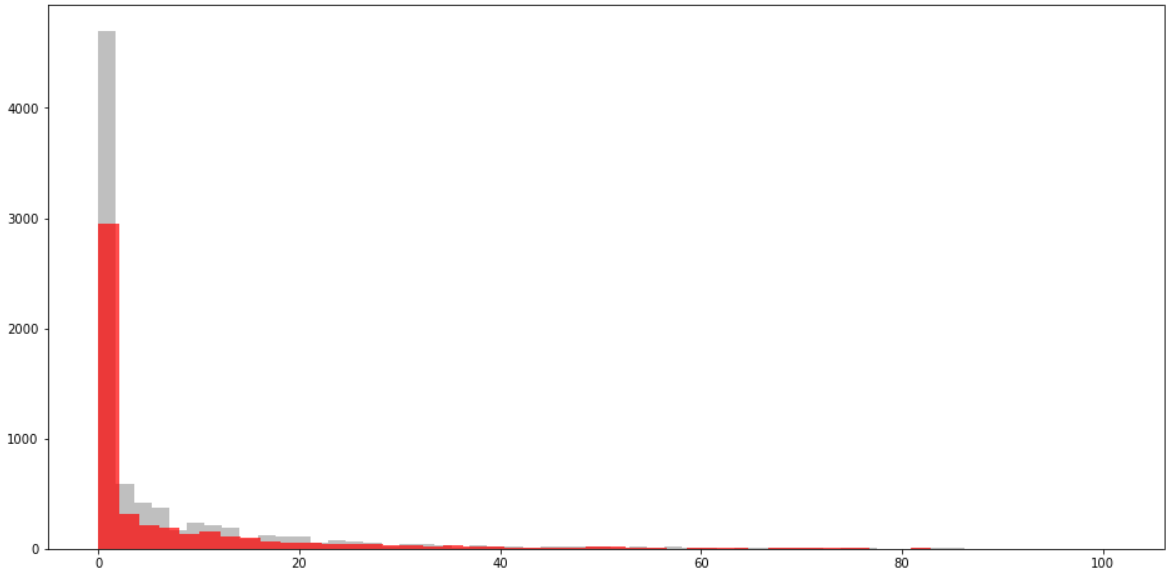


TfIDF-Cosine-Similarity2 Figure

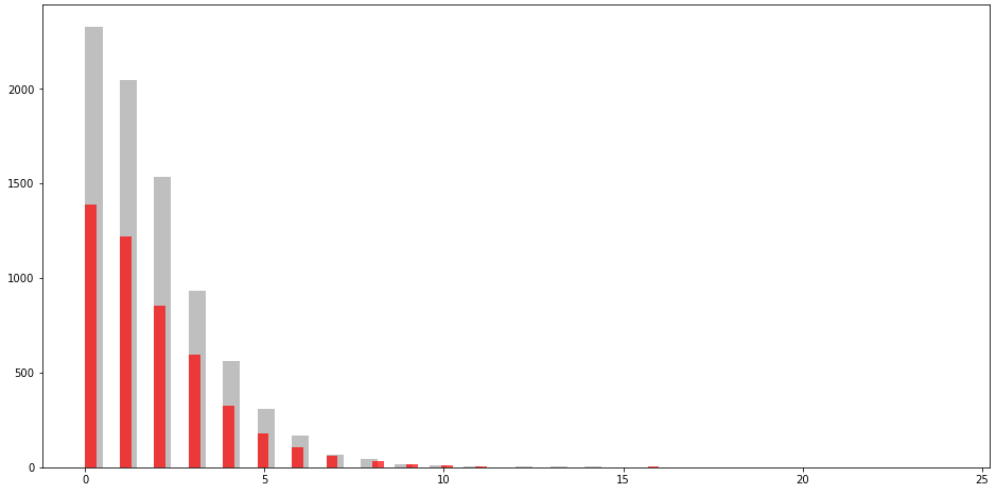
برای مقایسه پرسش و پاسخ میتوان چند نمونه ویژگی دیگر که مبتنی بر پرسش و پاسخ هستند معرفی کرد:



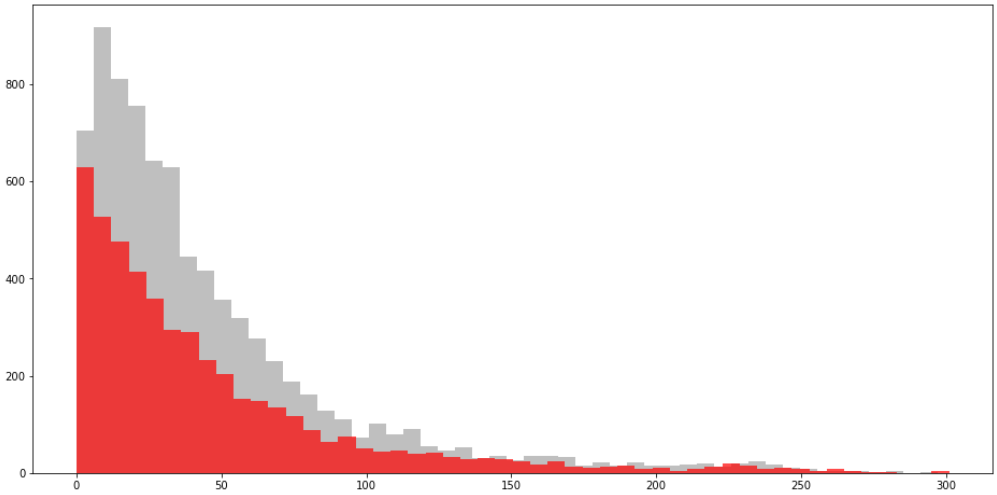
Overlap-Ratio3 Figure



Answer Span Figure



Same Word Sequence4 Figure



Informativeness5 Figure

- Overlap-ratio: بعنوان نسبت تعداد ترم‌های مشترک به اندازه دو سند پرسش و پاسخ و به صورت زیر تعریف میشود:

$$f(q,d) = \frac{|A \cap B|}{|A \cup B|}$$

- Answer Span: بیشترین فاصله بین دو کلمه سوال که در جواب ظاهر شده است.
- Same word sequence: اندازه طولانی ترین سری کلمات مشترک بین صورت و مخرج. بعنوان مثال:

A = [1,2,3,4,5]

B = [2,9,8,3,12,5]

F(A, B) = [2,3,5]

- Informative: تعداد ترم‌هایی از پاسخ که در پرسش دیده نشده و احتمالا اطلاعات اضافه‌ای به ما میدهد.

ویژگی‌های مبتنی بر پرسش و مبتنی بر پاسخ

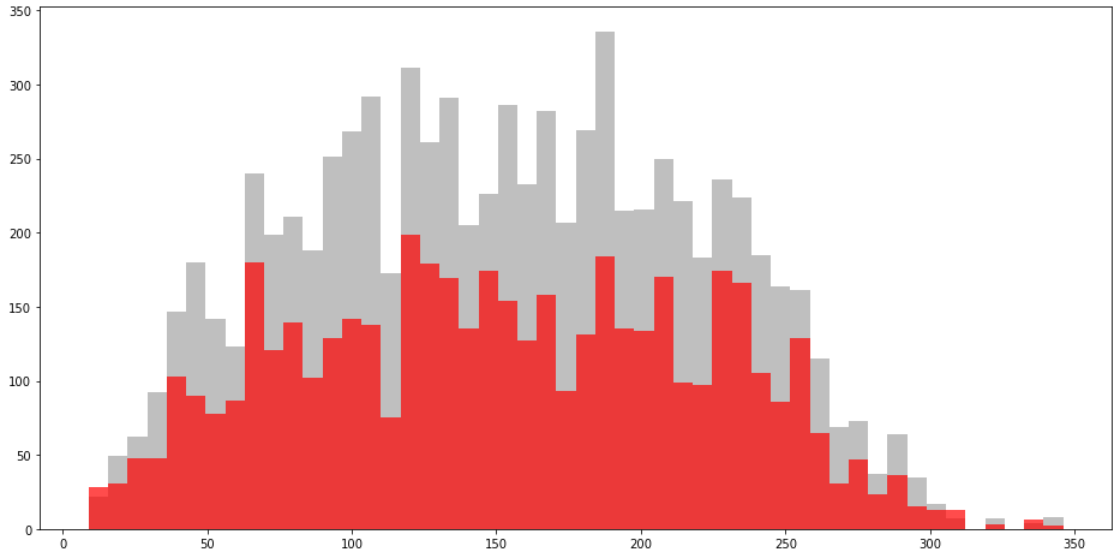
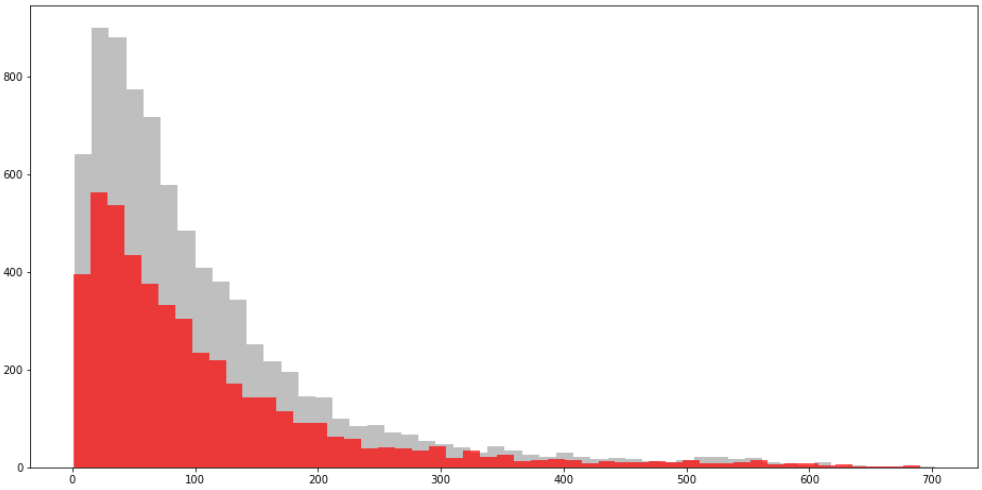


Figure 6 Query Length



Answer Length 7 Figure

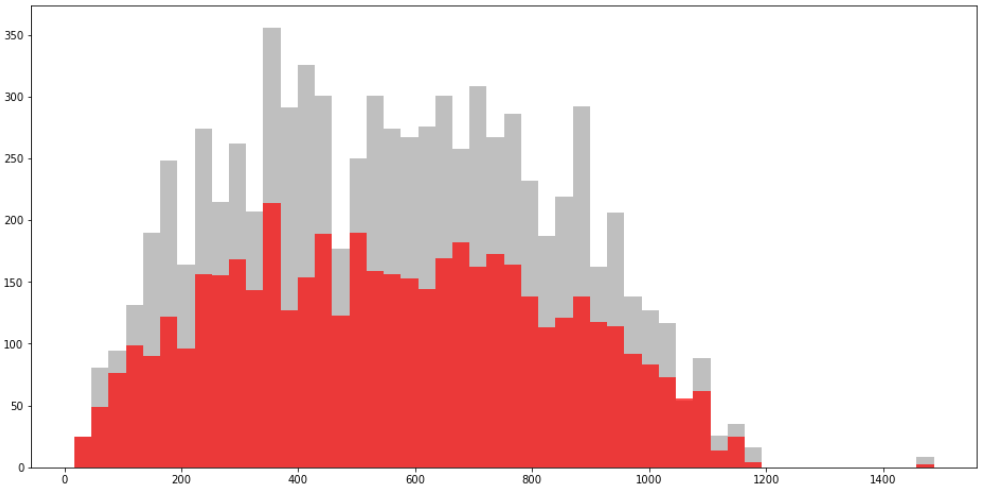


Figure 8 Question-IDF-sum

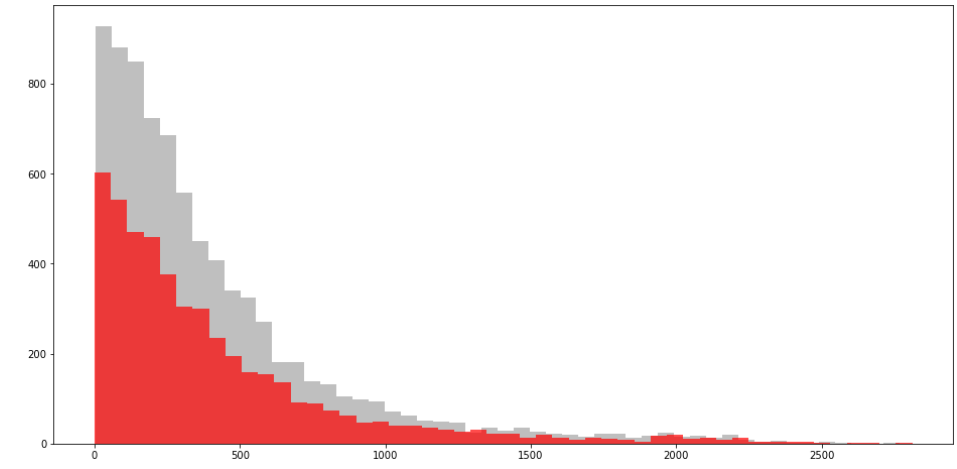


Figure 9 Answer-IDF-Sum

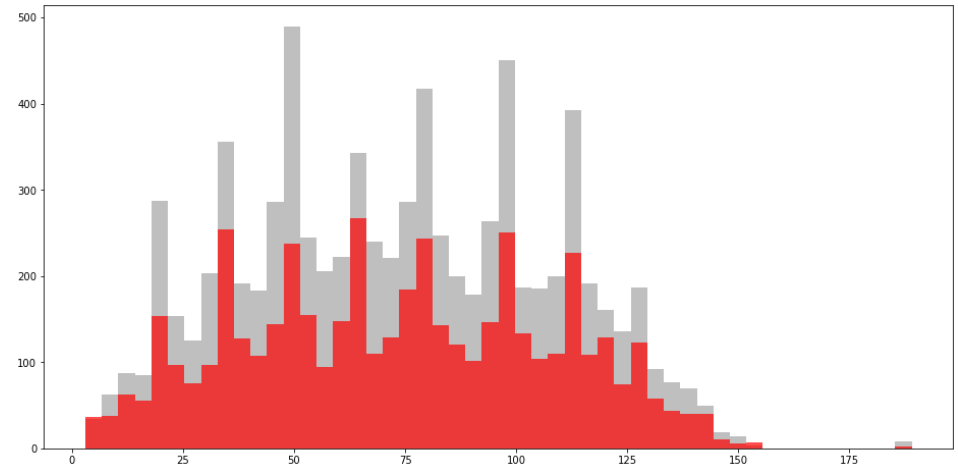


Figure 10 Question-TF-Sum

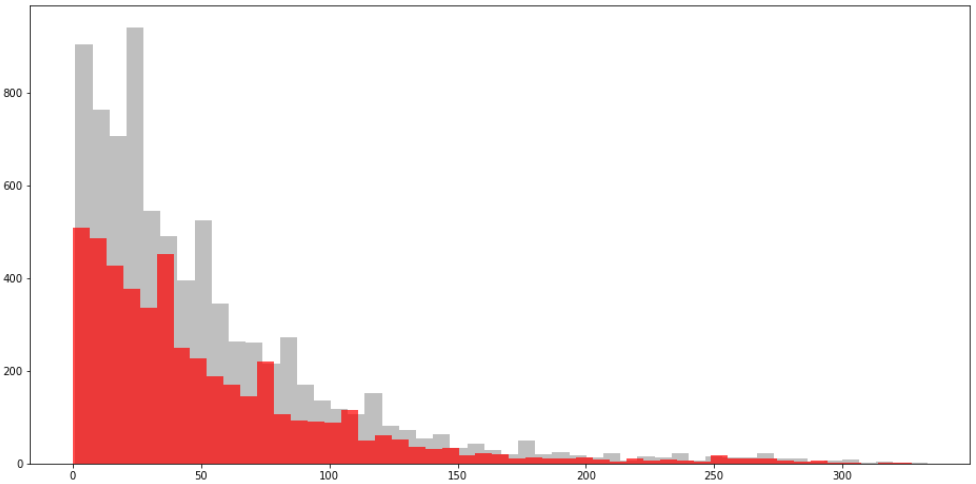


Figure 11 Answer-TF-Sum

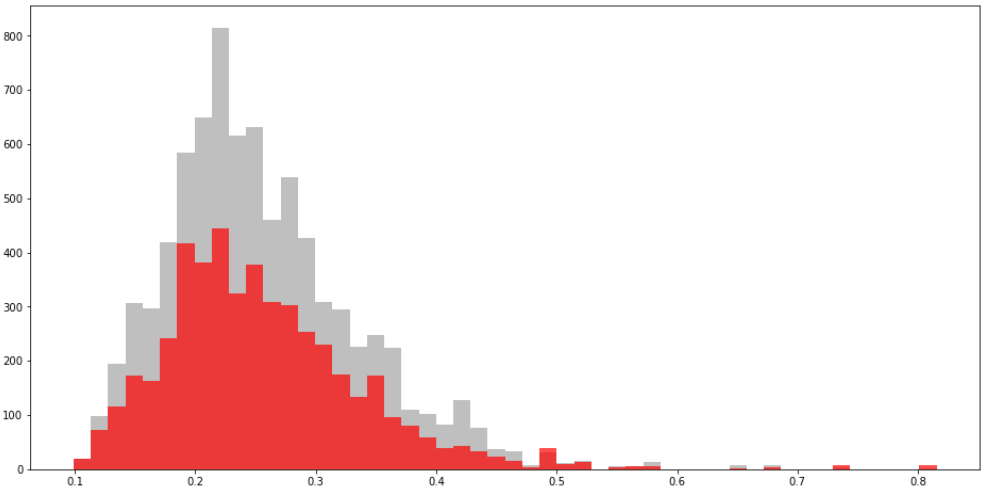


Figure 12 Question-TFIDF-Max

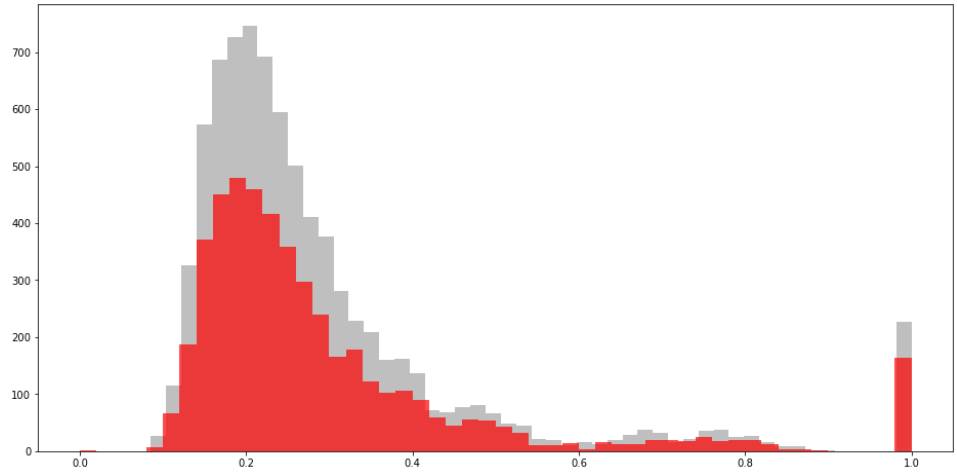


Figure 13 Answer-TFIDF-Max

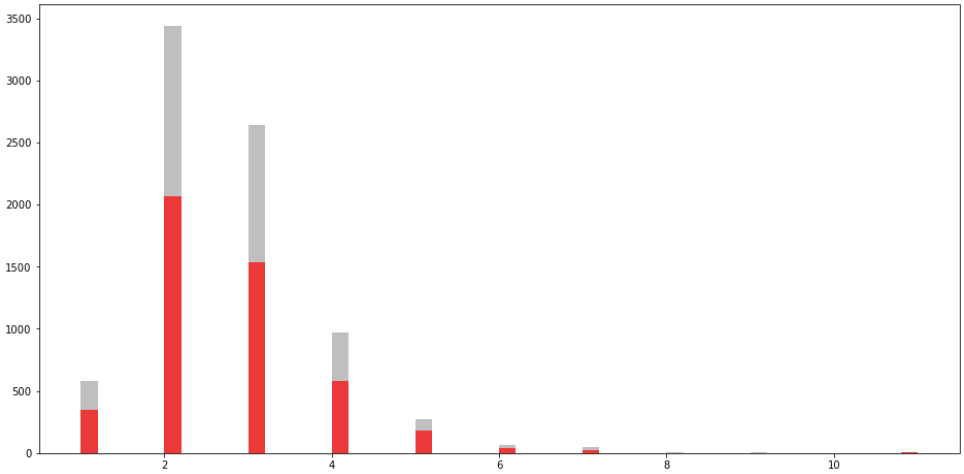


Figure 14 Query-TF-Max

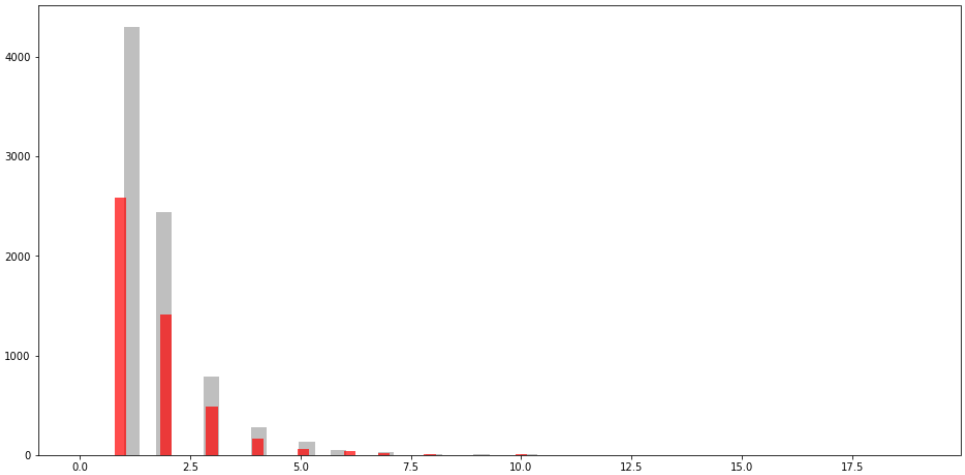


Figure 15 Answer-TF-Max

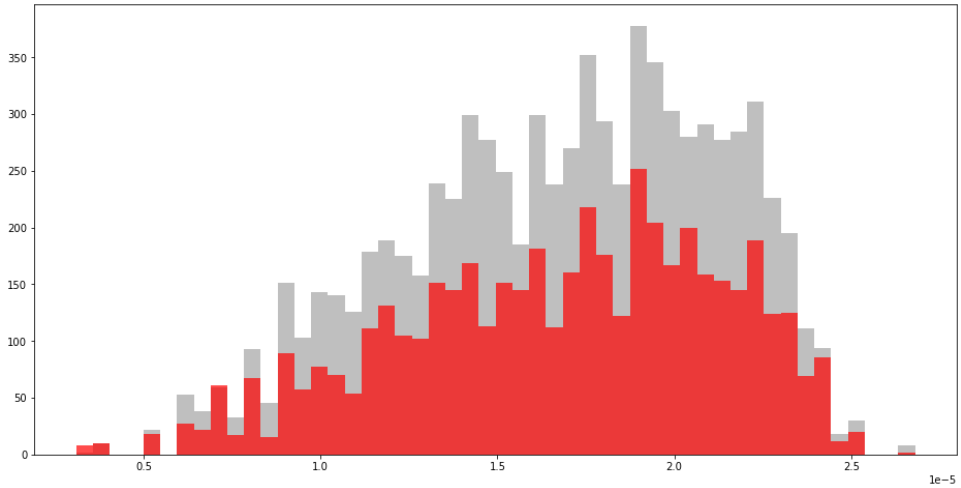


Figure 16 Query-TFIDF-Mean

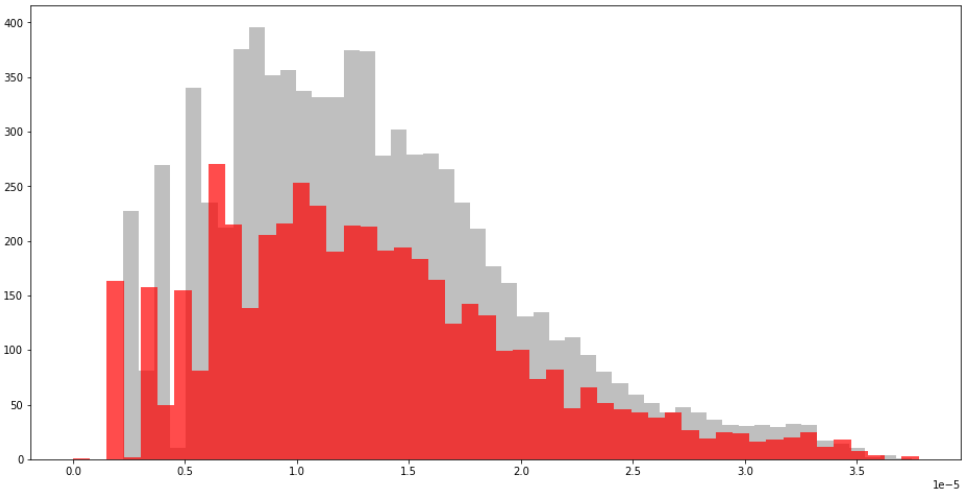


Figure 17 Answer-TFIDF-Mean

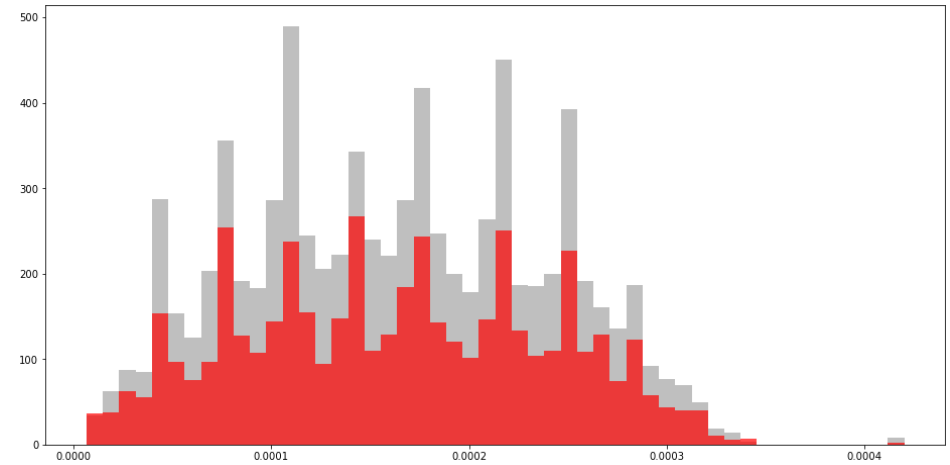


Figure 18 Query-TF-Mean

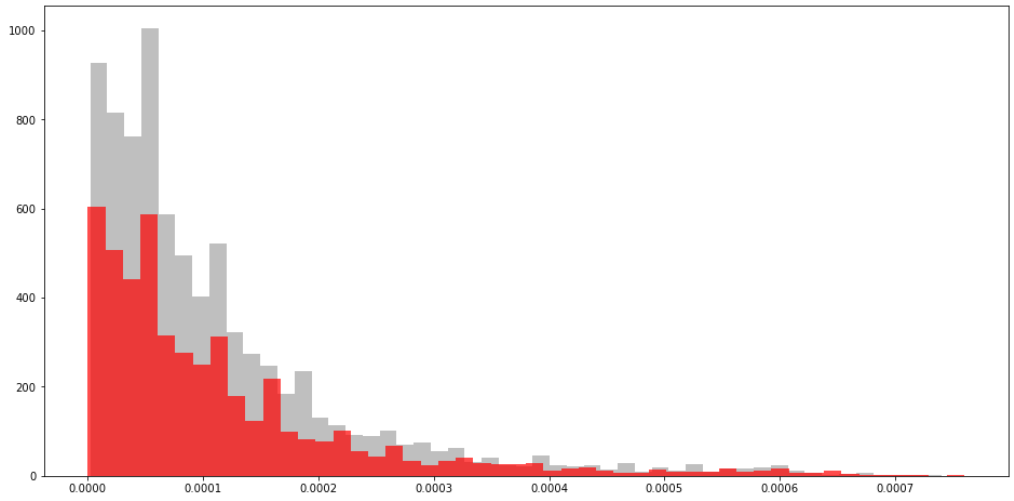


Figure 19 Answer-TF-Mean

موارد بالا با توجه به توزیع‌های متفاوتی که برای پاسخ‌های درست و غلط دارند بعنوان ویژگی برای جفت (پرسش-پاسخ) در نظر گرفته شدند. تعداد ویژگیهای در نظر گرفته شده از این قبیل بیشتر از لیست ذکر شده بودند که با انجام ارزیابی‌های دستی(اجرای الگوریتم بر روی داده dev) و همینطور الگوریتم‌های ماشینی مانند GridSearch تعدادی از آنها حذف شدند.

به منظور استخراج ویژگی‌ها و ذخیره بردارهای تولیدشده، کافیست فایل feature_extraction.py اجرا شود. نتایج بردارهای ویژگی در دایرکتوری feature_vectors/. و در زیربخش‌های مربوط در دو قالب csv و pickle (جهت بارگیری سریعتر برای مراحل بعد) ذخیره میشوند.

برای این منظور از کتابخانه Sklearn استفاده میکنیم و در ابتدا با استفاده از دو کلاس TFIDFVectorizer و CountVectorizer کلمات را به مقادیر عددی بازنشانی میکنیم.

پس از این مرحله برای بدست آوردن ویژگی‌های دیگر از این دو ماتریس استفاده میشود.

گام سوم: بازیابی پاسخ با کمک یک شبکه پرسپترون چندلایه

برای این گام مجدد از کتابخانه Sklearn و کلاس .neural_network.MPLClassifire استفاده میکنیم.

در ابتدا پس از بازخوانی اطلاعات بدست آمده از مرحله قبل، ویژگی‌های استخراج شده را توسط کلاس VarianceThreshold از لحاظ واریانس مورد بررسی قرار داده شده و تعدادی از آنها در نظر گرفته نمیشوند.

ابتدا الگوریتم را برای داده dev اعمال میکنیم و سپس بر اساس نتایج بدست آمده، همانطور که در قسمت قبل ذکر شد، ویژگی‌های منفی را حذف میکنیم.

پس از اعمال تنظیمات با توجه به تست‌های متوالی، نتایج نهایی بدست آمده برای این روش بر روی دیتاست dev به صورت زیر است:

```
Results on the test set:
              precision    recall  f1-score   support

       False       0.67      0.91      0.77      1476
        True       0.34      0.09      0.15       743

    accuracy                           0.64      2219
   macro avg       0.50      0.50      0.46      2219
weighted avg       0.56      0.64      0.56      2219
```

با این روش به دقت accuracy تقریبی 64% دست پیدا میکنیم.

حال الگوریتم را برای دیتاست تست اجرا کرده و مقادیر را ذخیره سازی میکنیم. مقادیر اولیه این اجرا در فایل mlp_test.csv ذخیره میشود و در آینده این نتایج را مرتب سازی میکنیم

گام چهارم: بازنمایی طیفی کلمات

از آنجایی که ماشین ها نمی توانند متن را درک کنند، باید از اعداد برای نمایش پرس و جوها و اسناد استفاده کنیم. و منظور از اعداد بردارها است. روش‌های متعددی برای تولید بردارها برای نمایش اسناد و پرس‌وجوها مانند Bag of Words (BoW)، فرکانس اصطلاح (TF)، فرکانس مدت و فرکانس معکوس سند (TF-IDF) و غیره وجود دارد که آنها را در قسمت قبل مورد بررسی قرار دادیم.

در اینجا، از word2vec استفاده خواهیم کرد. همانطور که از نام آن پیداست، word2vec به معنای "کلمه به بردار" است و این دقیقاً همان کاری است که انجام می دهد - کلمات را به بردار تبدیل می کند. یک چیز جالب در مورد word2vec این است که می تواند context را ضمیمه داده کرده و آن را با استفاده از بردارها نمایش دهد. به همین دلیل قادر است رابطه معنایی و نحوی بین کلمات را حفظ کند.

مدل فضای برداری بر روی مفهوم شباهت کار می کند. این فرض را بررسی می کند که ارتباط بین یک سند و پرس و جو مستقیماً با شباهت بین نمایش های برداری آنها مرتبط است."

به این معنی که یک سند (D1) با نمره شباهت بالاتر با پرس و جو (Q) نسبت به سند (D2) با نمره شباهت پایین‌تر مرتبط‌تر خواهد بود. این امتیاز شباهت بین بردارهای سند و پرس و جو به عنوان نمره تشابه کسینوس شناخته می شود و توسط:

$$Cosine\ Similarity(D,Q) = \frac{\vec{D} \cdot \vec{Q}}{\left\|\vec{D}\right\| \left\|\vec{Q}\right\|}$$

محسابه میشود. در این بخش، مدل word2vec خود را آموزش می‌دهیم و بردارهایی را برای اسناد و پرس و جوها در مجموعه آزمایشی برای بازیابی اطلاعات ایجاد می‌کنیم. اما قبل از آن، مجموعه داده را برای آموزش مدل word2vec آماده می کنیم. لطفاً توجه داشته باشید، ما قبلاً مجموعه آموزشی را ایجاد کرده‌ایم، اما می‌خواهیم از همان مدل word2vec برای تولید بردارها برای اسناد و پرس و جوها استفاده کنیم. به همین دلیل است که ما اسناد و پرس و جوها را برای ایجاد یک فایل واحد ترکیب می کنیم.

برای این منظور ۲ روش پیاده سازی و تست شد. در هر دو روش از کتابخانه genism برای ساخت بردارهای کلمات استفاده شده و همچنین در هر دو روش از داده‌های lemmatized شده استفاده میشود.

روش اول: پرسش و پاسخ جدا و محاسبه شباهت کسینوسی و استفاده از mlp

در روش اول در ابتدا داده‌های پرسش و پاسخ با یکدیگر ترکیب شده تا یک corpus بزرگ ایجاد شود و سپس بر روی این مجموعه داده عمل آموزش یک شبکه برای بازنمایی طیفی کلمات با کمک genism میکنیم و سپس با بردارهای بدست آمده برای هر عبارت یک بردار (به کمک mean گرفتن از بردارهای کلمات) متناظر ایجاد میکنیم. در مرحله بعد به ازای هر جفت پرسش و پاسخ مقدار شباهت کسینوسی بردار پرسش و پاسخ متناظر را بدست می آوریم و بعنوان امتیاز این جفت قرار میدهیم. سپس این امتیازات بدست آمده را به یک شبکه عصبی چندلایه perceptron برای محاسبه میدهیم. همانطور که در تصویر زیر مشاهده میشود، این روش خروجی ضعیف تری نسبت به روش گام قبل دارد که به علت عدم انتخاب درست روش‌های محاسبه است که در روش بعد تصحیح میشوند.

Results on the test set:

	precision	recall	f1-score	support
False	0.63	0.55	0.58	152
True	0.38	0.47	0.42	92
accuracy			0.52	244
macro avg	0.51	0.51	0.50	244
weighted avg	0.54	0.52	0.52	244

همانطور که مشاهده میشود، دقت این روش به صورت تقریبی برابر 52% است که از روش قبل بسیار ضعیف تر عمل میکند.

روش دوم: جفت پرسش و پاسخ و در نظر گرفتن پرسش و پاسخ بعنوان یک جمله و استفاده از LSTM

برای اصلاح روش قبل این بار برای بدست آوردن corpus بزرگتر و جامع تر از اجتماع جملات داده آموزش و داده تست در کنار هم استفاده میشود که هر جمله بیانگر یک پرسش و پاسخ متناظر آن به صورت join شده است.

هزاران نوع شبکه عصبی خاص وجود دارد که توسط محققین به عنوان اصلاحات یا تغییراتی در مدل های موجود پیشنهاد شده است. گاهی اوقات رویکردهای کاملاً جدید نیز بوجود آمده‌اند.

سه دسته از شبکه های عصبی مصنوعی وجود دارد که توصیه می شود به طور کلی روی آنها تمرکز شود. آن ها به شرح زیر هستند:

- پرسپترون های چندلایه (MLP)
- شبکه های عصبی کانولوشن (CNN)
- شبکه های عصبی مکرر (RNN)

این سه دسته از شبکه‌ها انعطاف‌پذیری زیادی را ارائه می‌کنند و طی دهه‌ها ثابت کرده‌اند که در طیف گسترده‌ای از مشکلات مفید و قابل اعتماد هستند. آنها همچنین دارای انواع فرعی زیادی هستند که به آنها کمک می کند تا آنها را در مورد ویژگی های چارچوب های مختلف مسائل پیش بینی و مجموعه داده های مختلف تخصصی کنند.

اکنون که می دانیم روی چه شبکه هایی تمرکز کنیم، باید ببینیم چه زمانی می توانیم از هر کلاس شبکه عصبی استفاده کنیم

چه زمانی از پرسپترون های چندلایه استفاده کنیم؟

پرسپترون های چندلایه یا به اختصار MLP، نوع کلاسیک شبکه عصبی هستند.

آنها از یک یا چند لایه نورون تشکیل شده اند. داده ها به لایه ورودی داده می شود، ممکن است یک یا چند لایه مخفی وجود داشته باشد که سطوح انتزاع را ارائه می دهد، و پیش بینی هایی روی لایه خروجی انجام میشود که لایه مرئی نیز نامیده می شود.

MLPها برای مسائل پیش‌بینی طبقه‌بندی مناسب هستند که در آن به ورودی‌ها یک کلاس یا برچسب تخصیص داده می‌شود.

آنها همچنین برای مسائل پیش بینی رگرسیون مناسب هستند که در آن یک کمیت با ارزش واقعی با توجه به مجموعه ای از ورودی ها پیش بینی می شود. داده ها اغلب در قالب جدولی ارائه می شوند، مانند آنچه در یک فایل CSV یا یک spreadshit مشاهده می کنید.

از MLP برای موارد زیر استفاده میشود:

- مجموعه داده های جدولی
- مشکلات پیش‌بینی طبقه‌بندی
- مشکلات پیش بینی رگرسیون

چه زمانی از شبکه های عصبی کانولوشنال استفاده کنیم؟

شبکه‌های عصبی کانولوشن یا CNN برای نگاشت داده‌های تصویر به یک متغیر خروجی طراحی شده‌اند.

آنها به قدری موثر ثابت شده‌اند که برای هر نوع مشکل پیش‌بینی که شامل داده‌های تصویر به‌عنوان ورودی می‌شود، بهترین روش هستند.

چه زمانی از شبکه های عصبی RNN استفاده کنیم؟

شبکه های عصبی RNN ها برای کار با مشکلات پیش بینی توالی(Sequence) طراحی شده اند.

مشکلات پیش‌بینی توالی اشکال مختلفی دارند و با انواع ورودی‌ها و خروجی‌های پشتیبانی شده توصیف می‌شوند.

چند نمونه از مشکلات پیش بینی توالی عبارتند از:

- One-to-Many: مشاهده ای به عنوان ورودی که به دنباله ای با چندین مرحله به عنوان خروجی نگاشت شده است.
- Multi-to-One: دنباله ای از مراحل متعدد به عنوان ورودی نگاشت شده به کلاس یا پیش بینی کمیت.
- Many-to-Many: دنباله ای از چندین مرحله به عنوان ورودی که به دنباله ای با چندین مرحله به عنوان خروجی نگاشت شده است.

مشکل Many-to-Many اغلب به عنوان sequence-to-sequence یا به اختصار seq2seq نامیده می شود.

RNN ها به طور کلی و LSTM ها به طور خاص بیشترین موفقیت را در هنگام کار با دنباله ای از کلمات و پاراگراف ها به دست آورده اند که به طور کلی پردازش زبان طبیعی نامیده می شود.

این شامل هم دنباله‌های متن و هم دنباله‌های زبان گفتاری است که به‌عنوان یک سری زمانی نمایش داده می‌شوند. آنها همچنین به عنوان مدل های تولیدی که نیاز به خروجی دنباله دارند، نه تنها با متن، بلکه در برنامه هایی مانند تولید دست خط استفاده می شوند.

از RNN برای موارد زیر استفاده میشود:

- داده های متنی
- داده های گفتاری
- مشکلات پیش‌بینی طبقه‌بندی
- مشکلات پیش بینی رگرسیون
- مدل های مولد

شبکه های عصبی مکرر برای مجموعه داده های جدولی مناسب نیستند، همانطور که در یک فایل CSV یا صفحه گسترده مشاهده می کنید. آنها همچنین برای ورودی داده های تصویری مناسب نیستند.

با توجه به توضیحات ذکر شده برای این task از شبکه عصبی LSTM استفاده میکنیم. همانطور که ذکر شد برای corpus ترکیبی از داده آموزش و تست را در نظر میگیریم تا word2vec اموزش دیده شده، دقت بهتری داشته باشد و سپس توسط این مدل کلمات corpus را بازنمایی میکنیم.

داده‌های مورد بررسی را به صورت جمله‌ای جفت پرسش و پاسخ تعریف میکنیم. تا با توجه به نمایش‌های برداری و این امر که قسمت‌های ابتدایی sequence هر ۱۰ جفت پرسش-پاسخ مرتبط با یک پرسش با یکدیگر برابر هستند، این جملات بررسی و رده بندی بشوند

در مرحله بعد dictionary هایی برای دسترسی راحت تر و سریع تر به داده و بازنمایی‌ها تعریف میکنیم و سپس توسط کتابخانه kerass از tensorflow شبکه عصبی مورد نظر را با داده‌های به دست آمده تعریف و در ابتدا برای ۱۰۰ ایپاک اجرا میکنیم.

با مشاهده نتایج این قسمت میبینیم که تا ایپاک ۸ مقدار loss بدست آمده در هر مرحله در حال کاهش میباشد ولی از این مرحله به بعد مقدار loss همواره دچار افزایش میشود و متوجه میشویم که این مقدار برابر مقدار بهینه تعداد epoch برای اجرای الگوریتم است. پس الگوریتم را با این مقدار بر روی داده dev اجرا میکنیم که نتایج نشان میدهد دقت این روش به صورت تقریبی برابر 70% میباشد که نسبت به هر دو روش قبل دقت بهتر و بالاتری را دارد.

```
- loss: 0.5368 - accuracy: 0.7308 - val_loss: 0.6010 - val_accuracy: 0.6992
```

در نهایت با مقادیر بدست آمده الگوریتم را برای داده تست اجرا و ذخیره میکنیم.

پ.ن:

- فایل‌های بلا استفاده و یا کدهای کامنت‌شده در پیاده‌سازی برای تست کردن ورودی‌های متفاوت و حالت‌های متفاوت بوده و به دلیل کمبود زمان برای refactor کردن متاسفانه حذف نشدند.
- برای ساده‌سازی فرآیند sort کردن امتیازهای روش mlp ، امتیاز تعلق گرفته به هر پاسخ، مقدار احتمال صحیح بودن آن منهای مقدار احتمال غلط بودن آن است.
- ترتیب اجرای فایلها بصورت زیر است
  1. Parse_xml_data
  2. Pre_process
  3. Feature_extraction
  4. Mlp
  5. Word2Vec_Lstm
  6. Sort_output
- خروجی نهایی در پوشه کد در دو فایل tsv با نامهای output_mlp و output_word2vec با فرمت خواسته شده ذخیره شده‌اند

پایان.