

به نام خدا



دانشگاه تهران

دانشکده فنی

دانشکده مهندسی برق و کامپیوتر



## درس پردازش زبان طبیعی

پاسخ تمرین ۲

نام و نام خانودگی: زهرا ریحانیان

شماره دانشجویی: ۸۱۰۱۰۱۱۷۷

فروردین ماه ۱۴۰۲

|   |  |
|---|--|
| ۳ | پاسخ سوال دوم.....                         |
| ۳ | پاسخ بخش اول - پیش پردازش مجموعه داده..... |
| ۳ | پاسخ بخش دوم - بار گذاری GLOVE.....        |
| ۳ | پاسخ بخش سوم - آموزش مدل.....              |
| ۵ | پاسخ سوال سوم.....                         |
| ۵ | پاسخ بخش اول.....                          |
| ۵ | پاسخ بخش دوم.....                          |
| ۶ | پاسخ بخش سوم.....                          |

## پاسخ سوال دوم

کد مربوط به این بخش در مسیر `codes/Q2.ipynb` موجود است.

### پاسخ بخش اول – پیش پردازش مجموعه داده

در این مرحله ابتدا فایل داده شده را لود و در یک دیتافریم ذخیره کردم.

در مرحله بعد GLOVE را دانلود و آنزپ کردم و به خاطر محدودیتی که در حافظه داشتم از بردارهای ۲۰۰ بعدی آن استفاده کردم و در یک دیکشنری آن ها را ذخیره نمودم.

در قدم بعدی ابتدا X که عناوین خبری و y که کنایه بودن یا نبودن را مشخص می کنند، جدا کردم. برای پیش پردازش همه ی حروف را به حروف کوچک تغییر دادم و نشان گذاری ها را حذف کردم.

ابتدا با استفاده از متد `train_test_split` داده ها را به دو دسته آموزش و ارزیابی را به نسبت ۸۰ به ۲۰ جدا کردم. سپس برای هر یک از سطر ها در `X_train` و `X_test` این عملیات ها را انجام دادم:


با استفاده از `lemmatizer` و `word_tokenize` عملیات توکن سازی و یافتن بن واژه ها را انجام دادم و توکن هایی را که در دیکشنری مربوط به GLOVE نبودند، حذف کردم و بقیه را در یک لیست ذخیره کردم. (ادامه در پاسخ بخش دوم)

### پاسخ بخش دوم – بار گذاری GLOVE

در مرحله بعد برای هر توکن، بردار متناظر با آن در GLOVE را پیدا کرده و در یک لیست ذخیره کردم. در نهایت برای این که تعداد بردار های GLOVE در هر سطر یکسان باشد، تابع `pad_X` را تعریف کردم که کار آن این است که تعداد بردار ها در هر سطر را به تعداد دلخواه برساند. من در اینجا تعداد را ۵۰ گذاشتم. برای بعضی که طول کمتر دارند بردار صفر اضافه کردم و آن هایی که طول بیشتر دارند تا همان طول مورد نظر را در نظر می گیرم و بقیه را حذف کردم. در واقع از حداکثر تعداد بردار های یک سطر استفاده نکردم، چون وقتی این کار را کردم برنامه به علت پر شدن رم، کرش می کرد. در نهایت برای این که بتوانم از این داده ها در مدل `logistic regression` استفاده کنم، آن ها به ۲ بعد `reshape` کردم.

### پاسخ بخش سوم – آموزش مدل

همان طور که در صورت تمرین خواسته شده بود، مدل `Logistic regression` را از `sklearn` دریافت کردم و آن را با استفاده از جانیمایی های بدست آمده آموزش دادم. در نهایت با استفاده از داده های ارزیابی، مدل را ارزیابی کردم و نتایج زیر بدست آمد:



|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.78      | 0.78   | 0.78     | 2995    |
| 1            | 0.76      | 0.76   | 0.76     | 2729    |
| accuracy     |           |        | 0.77     | 5724    |
| macro avg    | 0.77      | 0.77   | 0.77     | 5724    |
| weighted avg | 0.77      | 0.77   | 0.77     | 5724    |

شکل ۱ ارزیابی مدل بر اساس معیار های مختلف

همان طور که مشاهده می کنید، مدل به طور میانگین برای هر ۳ معیار خواسته شده، مقدار ۷۷ درصد را کسب کرده است. یعنی ۷۷ درصد از کلاس های درست را پیش بینی کرده است و در ۷۷ درصد مواقع درست پیش بینی می کند.

تشخیص کنایه در متن از جمله تسک های چالش برانگیز در حوزه پردازش زبان طبیعی است بنابراین می توان ۷۷٪ را برای precision، recall و f1-score معقول دانست.

## پاسخ سوال سوم

کد مربوط به این بخش در مسیر `codes/Q3.ipynb` موجود است.

### پاسخ بخش اول

برای حل این مساله، ابتدا داده ها را لود و به فرمت لیستی از `string` ذخیره کردم.

برای پیش پردازش، تمام حروف را به حروف کوچک تبدیل کردم و نشان گذاری ها را حذف کردم. در نهایت همه ی توکن ها را با استفاده از `split()` از هم جدا کردم و همه را در یک لیست به اسم `corpus` ذخیره کردم.

برای تولید نمونه های `skipgram`، ابتدا لازم بود به هر کلمه یک شناسه یکتا اختصاص بدهم. به همین خاطر از `Tokenizer` کتابخانه `keras` ماژول `preprocessing.text` استفاده کردم و آن را روی `corpus` فیت کردم و با استفاده از فیچر `word_index` کلمات را به شناسه هایی یکتا، تبدیل کردم. سپس در لیست `wids` شناسه متناظر با کلمات داخل `corpus` را ذخیره کردم.

در ادامه با استفاده از متد `skipgrams` از ماژول `tf.keras.preprocessing.sequence` جفت `target` و `context` ها به همراه لیبل های متناظر را بدست آوردم و همان طور که خواسته شده بود، به ازای هر نمونه مثبت ۴ نمونه منفی تولید شد که در نهایت ۴۱۸۰۷۴۰ نمونه تولید شد.

برای استفاده از این نمونه ها در مدل، ابتدا `target` ها و `context` ها را جدا کردم و فرمت لیست را به فرمت آرایه `numpy` تغییر دادم و در نهایت در یک دیتاست با `batch_size = 1024` و `buffer_size = 10000` ذخیره کردم.

مدل `skipgram` را به این صورت تعریف کردم که شامل دو لایه جانمایی برای `target` و `context` و یک لایه `dense` با `activation = "sigmoid"` است. طول `embedding` را برابر ۱۰۰ در نظر گرفتم و مدل را با `optimizer='nadam'` و `loss='binary_crossentropy'` کامپایل و به تعداد ایپاک ۳۰ تا آن را آموزش دادم. با این ساختار توانستم بهترین نتیجه را بگیرم و به دقت حدود ۹۴.۶ درصد رسیدم. در نهایت وزن های مربوط به دو لایه جانمایی `target` و `context` را استخراج و جمع کردم و بردار ویژگی کلمات را بدست آوردم که آن ها را در ماتریس `weights` ذخیره کردم.

منبع من برای حل این بخش، این لینک<sup>۱</sup> بود که متناسب با این مساله و داده هایی که در اختیارمان قرار داده شد و دانشی که داشتم، آن را تغییر دادم.

### پاسخ بخش دوم

<sup>۱</sup> <https://medium.com/@corymaklin/word2vec-skip-gram-904775613b4c>

برای این بخش ابتدا عملیات خواسته شده را روی بردار ها به صورت زیر انجام دادم:

```
result = weights[word2id['king']] - weights[word2id['man']] + weights[word2id['woman']]
```

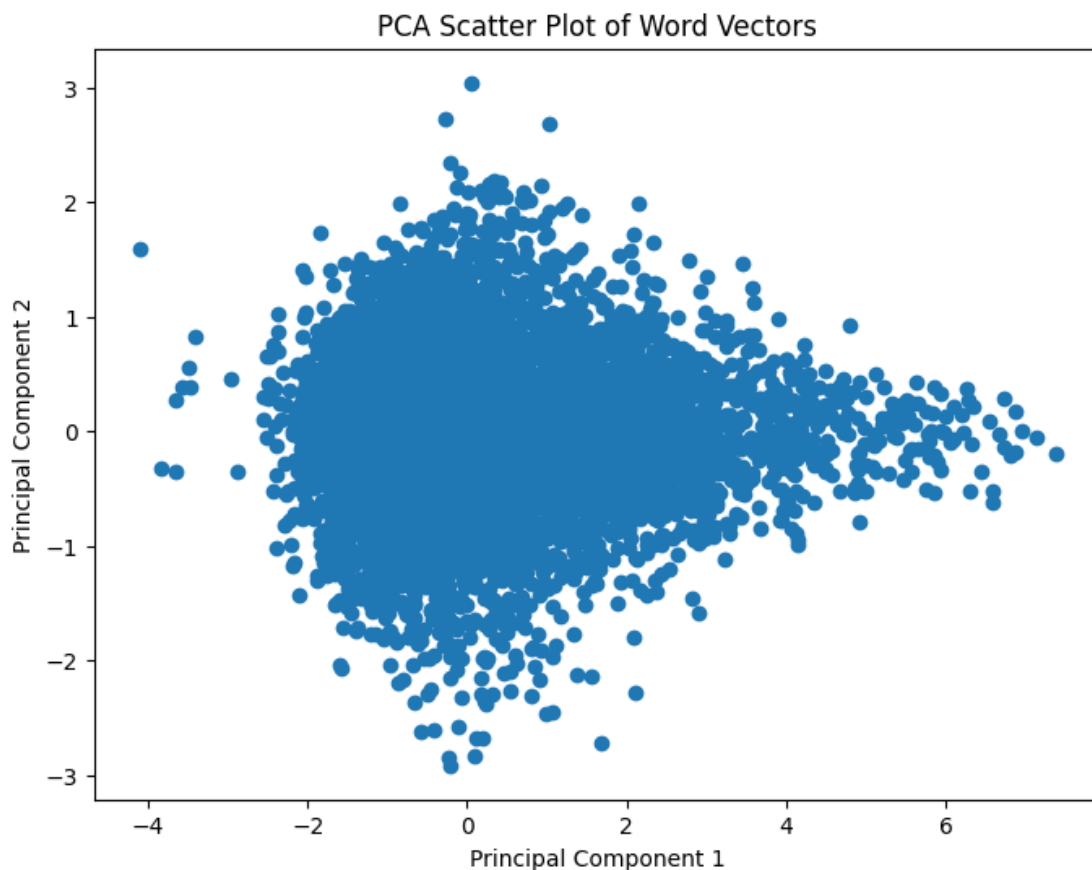
بردار queen هم به این صورت گرفتم:

```
queen_vector = weights[word2id['queen']]
```

و سپس ضرب داخلی این دو بردار را محاسبه کردم که متناسب با مقدار similarity یا تشابه بین دو بردار است. برای این که احتمال را محاسبه کنم از sigmoid استفاده کردم و در نهایت احتمال ۹۹ درصد را بدست آوردم. این نشان می دهد که مدل خوب آموزش دیده است که توانسته بردار هایی با معنی برای کلمات تولید کند. در واقع یکی از ویژگی های معنایی جانمایی ها همین تشابه رابطه ای است که در این بخش یک نمونه از آن روی این جانمایی ها امتحان شد و نتیجه داد.

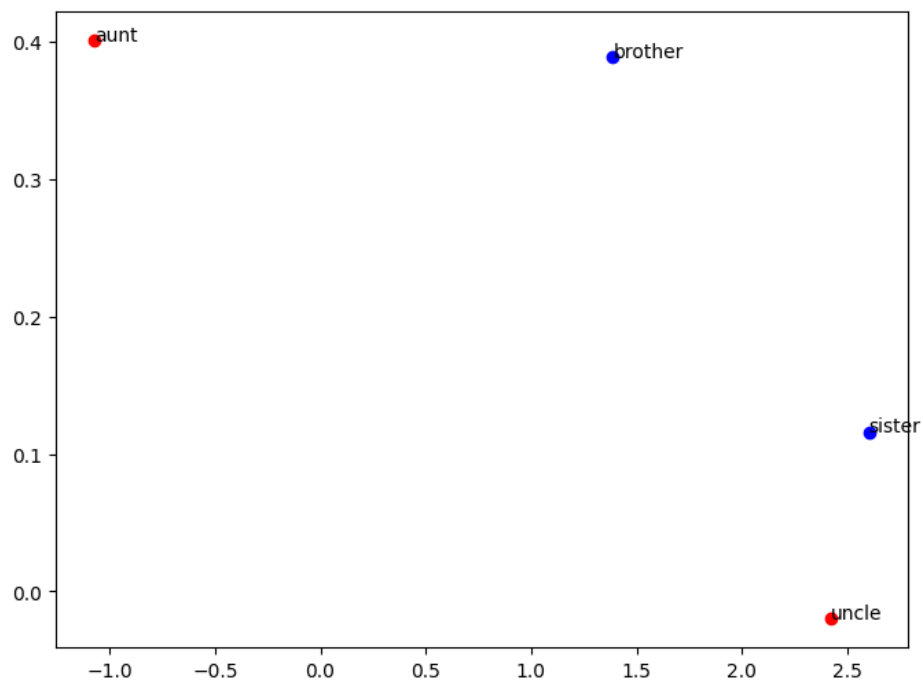
#### پاسخ بخش سوم

برای این بخش ابتدا با استفاده از PCA بردار های کلمات را به فضای دو بعدی کاهش دادم. تصویر زیر همه ی کلمات را در یک نمودار نقطه ای نشان می دهد.



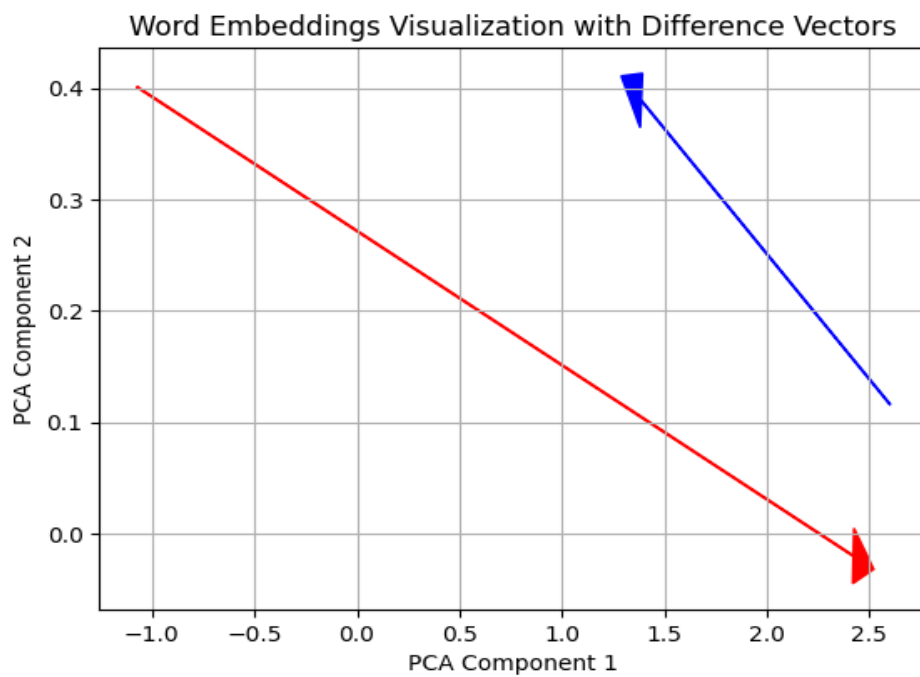
شکل ۲ نمودار نقطه ای همه ی بردار های کلمات در دو بعد

همان طور که خواسته شد نقاط مربوط به brother, sister, uncle, aunt پیدا شد:



شکل ۳ نقاط مربوط به کلمات خواسته شده

و سپس بردار تفاضل آن ها را بدست آوردم:



شکل ۴ نمودار بردار های تفاضل خواسته شده

همان طور که مشاهده می کنید، بردار های تفاضل بدست آمده تقریبا به موازات هم هستند. این موازی بودن هم به علت تشابه رابطه ای رخ داده است و نشان می دهد بردار های کلمات، با معنی هستند و مدل تقریبا خوب آموزش دیده است. این ویژگی در بردار های کلمات باعث می شود که کاربرد های زیادی در تسک های مختلف پردازش زبان طبیعی مانند جستجوی معنایی، ماشین ترجمه و پرسش و پاسخ ها داشته باشند.

البته بردار های تفاضل بدست آمده، اختلاف جهت دارند که ممکن است با بالا بردن سائز امبدینگ و تعداد ایپاک ها برطرف شود اما در تلاش هایی که داشتم، این بهترین نتیجه ای بود که گرفتم.