به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس پردازش زبان طبیعی

پاسخ تمرین ۲

نام و نام خانودگی: سامان اسلامی نظری

شماره دانشجویی: ۸۱۰۱۹۹۳۷۵

فروردین ماه ۱۴۰۳

فهرست

پاسخ سوال دوم

پیش از انجام بخش‌های این سوال، تمام داده‌ها خوانده شده و در `sarcasm_df` ذخیره شدند. برای اینکه کد قابل اجرا باشد، باید فایل‌های sarcasm.json و glove.6b.50d.txt در کنار آن قرار گرفته باشند.

پاسخ بخش اول – پیش‌پردازش مجموعه داده

برای پیش‌پردازش داده از تابع `tokenize_text` استفاده شد که در بخش اول نوشته شده بود؛ این تابع با استفاده از کتابخانه `nltk` ابتدا تمام حروف را به حروف کوچک تبدیل می‌کند، سپس stop word-ها را حذف کرده و در نهایت کلمات را lemmatize می‌کند. این تابع علاوه بر پیش‌پردازش، متن را توکنایز نیز می‌کند.

در نهایت با استفاده از تابع `train_test_split` در کتابخانه `sklearn` داده‌ها را به دو بخش تمرین و ارزیابی تقسیم می‌کنیم.

پاسخ بخش دوم – بارگذاری GLOVE

هر خط از فایل `glove.6b` شامل یک کلمه و برداری از اعداد جلوی آن می‌باشد. با فرض اینکه کلمات شامل فاصله نمی‌باشند، بردارها را در یک `np.array` ذخیره کرده و آن‌ها را به کلمات متناظرشان نگاشت می‌دهیم. این نگاشت در `glove_embeddings` ذخیره شده است.

در مرحله بعدی مجموعه کلمات[۱] (در متغیر vocabulary) را بدست آورده و نگاشت شاخص[۲] (در متغیر `vocabulary_index`) به کلمات را نیز محاسبه می‌کنیم. پس از این مرحله تابع `create_embedding_matrix` را ایجاد می‌کنیم. وظیفه این تابع این است که هر کلمه از مجموعه کلمات موجود در داده‌ها را در GloVe جست و جو کرده (از نگاشت ساخته شده در مرحله قبل استفاده می‌کنیم) و در صورتی که این کلمه در GloVe موجود بود، بردار متناظرش را در ماتریس قرار می‌دهد؛ در غیر این صورت بردار آن کلمه را صفر در نظر می‌گیرد.

حالا که ماتریس جانمایی آماده است، برای Vectorize کردن مجموعه داده، ابتدا تعداد تکرار کلمات در داده را با استفاده از `CountVectorizer` در کتابخانه `sklearn` محاسبه کرده و این مقدار را متناظرا در بردار هر کلمه ضرب می‌کنیم.

پاسخ بخش سوم – آموزش مدل

---

[۱] vocabulary

[۲] index

با استفاده از نتایج بخش‌های قبل، یک مدل Logistic Regression با استفاده از کتابخانه `sklearn` آموزش داده شد. نتایج این مدل روی داده‌های ارزیابی به صورت زیر می‌باشد:

	precision	recall	f1-score	support
0	0.79	0.68	0.73	3494
1	0.59	0.72	0.65	2230
accuracy			0.70	5724
macro avg	0.69	0.70	0.69	5724
weighted avg	0.71	0.70	0.70	5724

عکس ۱ نتایج ارزیابی مدل رگرسیون برای داده‌های sarcasm

یکی از دلایل اینکه دقت این مدل خیلی بالا نیست می‌تواند به این موضوع برگردد که ادبیات توییتر با ادبیات متونی که GloVe روی آن‌ها آموزش داده شده متفاوت است. مجموعه داده استفاده‌شده در GloVe 6B از ویکی‌پدیا و برخی منابع مشابه بدست آمده که چندان مناسب تشخیص حس شوخ طبعی نیستند. یکی از دلایلی که متون جدی (کلاس صفر) بهتر تشخیص داده‌شده‌اند همین موضوع است.

پاسخ سوال سوم

برای اینکه این کد قابل اجرا باشد، باید فایل `advs.txt` که در لینک صورت پروژه به آن اشاره شده، در کنار آن قرار گرفته باشد. ابتدا داده‌ها را دانلود و بارگذاری می‌کنیم. برای انجام پیش‌پردازش، صرفا به تبدیل حروف به حروف کوچک و از بین بردن علائم نگارشی بسنده می‌کنیم.سپس با استفاده از `TextVectorization` تمام توکن‌ها را به شاخص متناظرشان در مجموعه کلمات نگاشت می‌دهیم. در نهایت نیز مجموعه داده را به دنباله‌هایی از توکن‌ها به طول ۱۰ تبدیل می‌کنیم. مجموعه داده در این مرحله به شکل زیر می‌باشد:

```
length = 9566
[   2 1739    6  130   34    0    0    0    0    0] >>> ['the', 'adventures', 'of', 'sherlock', 'holmes', '', '', '', '', '']
[ 631 7676 7304    0    0    0    0    0    0    0] >>> ['arthur', 'conan', 'doyle', '', '', '', '', '', '', '']
[ 258    6 1699    0    0    0    0    0    0    0] >>> ['table', 'of', 'contents', '', '', '', '', '', '', '']
[  7 809   8 911   0   0   0   0   0   0] >>> ['a', 'scandal', 'in', 'bohemia', '', '', '', '', '', '']
[  2 538 643   0   0   0   0   0   0   0] >>> ['the', 'redheaded', 'league', '', '', '', '', '', '', '']
```

عکس ۲ مجموعه داده Vectorize-شده در سوال سوم

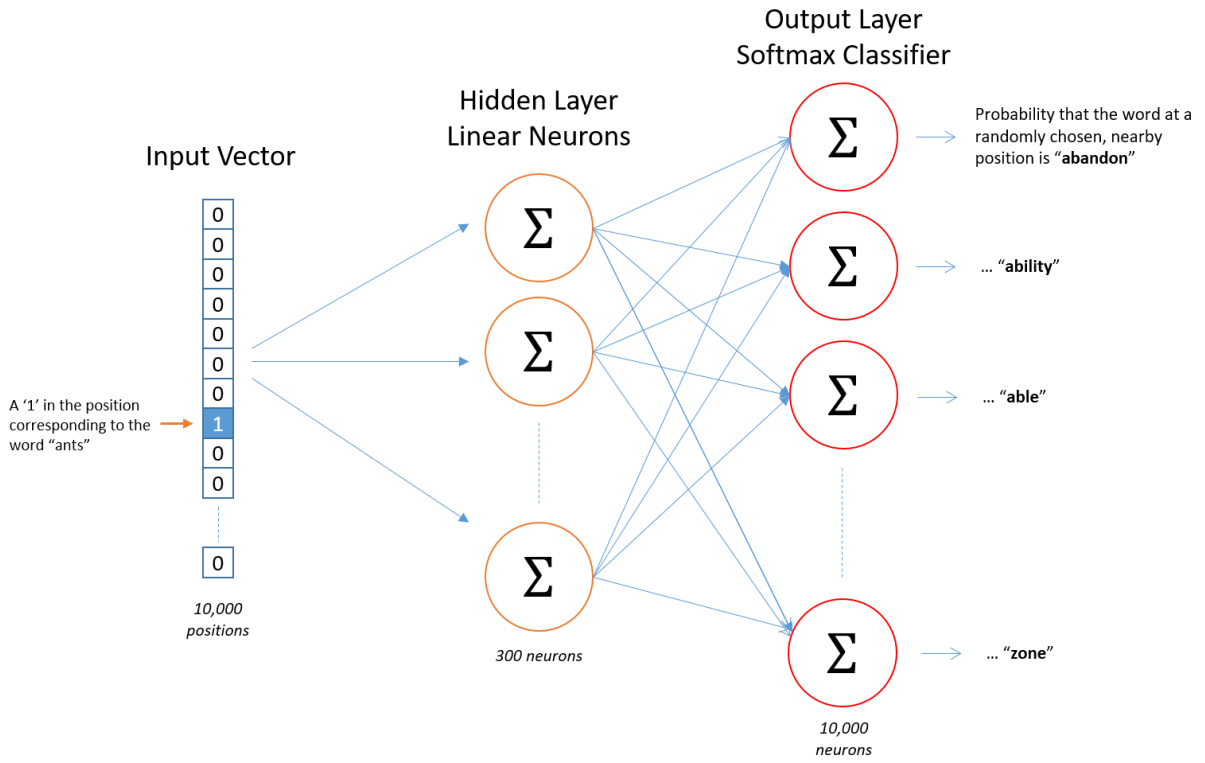
پاسخ بخش اول – پیاده‌سازی مدل SKIP-GRAM

قبل از آنکه این مدل را پیاده‌سازی کنیم، تابع نمونه‌برداری را به صورت Negative Sampling پیاده‌سازی می‌کنیم. برای انتخاب نمونه‌های مثبت از تابع `skipgrams` استفاده می‌کنیم که وظیفه آن انتخاب جفت توکن‌هایی است

که با توجه به یک کلمه هدف، کلمه جفت آن در Context Window مد نظر ظاهر می‌شود. برای انتخاب جفت‌های نمونه‌های منفی از تابع `log_uniform_candidate_sampler` استفاده می‌کنیم. این تابع با استفاده از توزیع Zipf نمونه‌ها را انتخاب می‌کند؛ این توزیع مقادیر موجود در لیست را طوری نسبت می‌دهد که احتمال انتخاب مقادیر آخر لیست کم‌تر می‌باشد. این توزیع در جداول توزیع کلمات در زبان طبیعی مورد استفاده قرار می‌گیرد.

در نهایت پنج نمونه که چهارتای آن‌ها منفی و یکی از آن‌ها مثبت است را به هم چسبانده و برچسب متناظرشان را نیز ایجاد می‌کنیم.

در مرحله بعدی مدل شبکه عصبی Skip-Gram را پیاده‌سازی می‌کنیم. مدل این شبکه عصبی در تصویر زیر قابل مشاهده است که از این لینک دریافت شده:



عکس ۳ مدل شبکه عصبی Skip-Gram

در ابتدا ورودی را به صورت یک بردار به روش One-Hot می‌سازیم. سپس یک لایه مخفی داریم که به تعداد کلمات در مجموعه داده سطر داشته و تعداد ستون‌های آن برابر تعداد ویژگی‌های مد نظرمان است. در این جا این مقدار برابر ۱۰۰ می‌باشد. هنگامی که شبکه عصبی را آموزش می‌دهیم، این لایه بروزرسانی شده و در نهایت هر سطر که متناظر با یک کلمه است، مشابه سطرهایی می‌شود که بیش‌تر به آن‌ها نزدیک است؛ نزدیک بودن یعنی احتمال اینکه این دو کلمه با همدیگر در یک Context Window ظاهر شوند بیش‌تر است.

لایه‌های مذکور را با استفاده از روش subclassing در keras پیاده‌سازی کرده و مدل را آموزش می‌دهیم.

پاسخ بخش دوم – میزان شباهت بردارها

نتیجه عملیات مذکور به صورت زیر می‌باشد:

```
queen/ king - man + woman: -0.034237589687108994
queen/ king: -0.06889821588993073
queen/ woman: 0.13184642791748047
man/ woman: 0.30305355578727722
```

عکس ۴

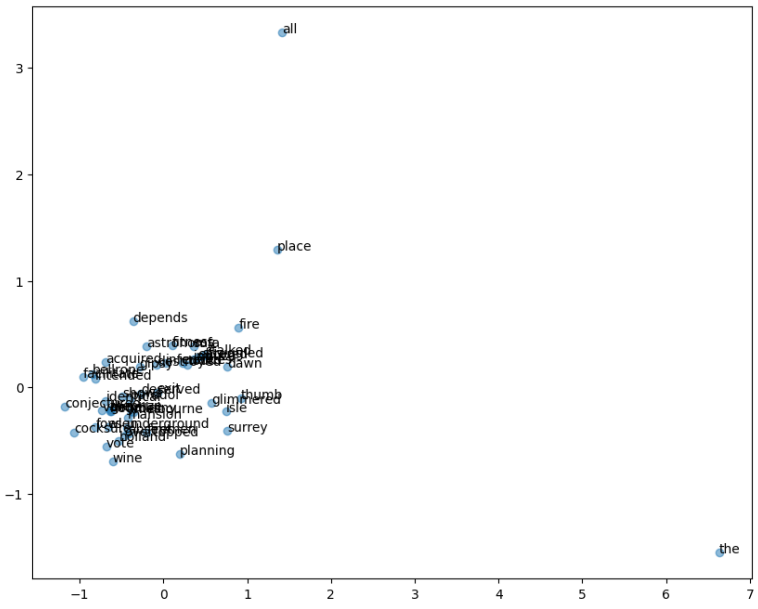
متاسفانه به دلیل آنکه تعداد کمی از کلمات queen و king در مجموعه داده حضور داشت، نتایج چندان مشهود نمی‌باشند؛ با این حال با مقایسه اختلاف بردارهای queen و king با queen و king – man + woman می‌توان مشاهده کرد که اختلاف دومی کم‌تر می‌باشد که منطقی است. کلمه king می‌تواند متشکل از مفاهیم پایه‌ای‌تری باشد که بخشی از آن را با کلمه man به اشتراک می‌گذارد؛ این موضوع برای queen و woman نیز صادق است. بنابراین با کم کردن مفهوم man و اضافه کردن مفهوم woman به کلمه king می‌توانیم به مفهوم کلمه queen نزدیک‌تر شویم.

برای سنجش اختلاف‌ها از روش مشابهت کسینوسی استفاده کردیم. در این روش ابتدا روی دو بردار عملیات ضرب داخلی انجام داده و سپس مقدار نهایی را تقسیم بر ضرب اندازه دو بردار می‌کنیم:

$$Cosin\ similarity = \frac{A.B}{A \times B}$$

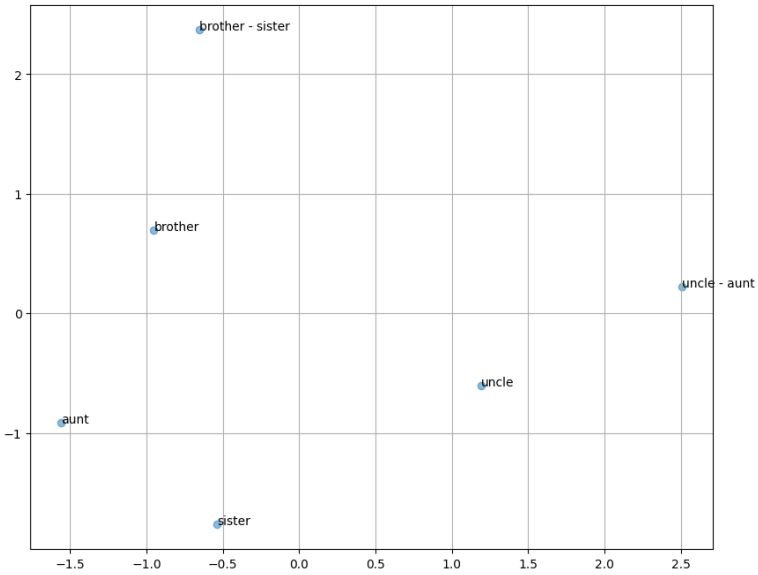
بخش سوم – نمودار داده‌ها

برای کشیدن نمودار کلمات، از آنجایی که تعداد کلمات در مجموعه داده بسیار زیاد بود، یک زیر مجموعه از آن‌ها به صورت رندوم انتخاب شد که نمودار آن به صورت زیر شد:



عکس ۵ نمودار بردارهای Skip-Gram برای یک زیرمجموعه از کلمات اصلی

همچنین بردار تفاصلات خواسته‌شده به صورت زیر بود:



عکس ۶ بردار تفاضلات Skip-Gram

همانطور که انتظار می‌رفت دو بردار brother و sister در مقابل هم و دو بردار uncle و aunt نیز در مقابل یکدیگر قرار دارند. بردارهای بدست آمده به خوبی مخالف بودن این کلمات را نشان می‌دهند.