به نام خدا

نام دانشگاه: پردیس فارابی دانشگاه تهران

نام درس: پردازش زبان های طبیعی

نام تمرین: Word2Vec

دانیال فرهنگی ۲۲۰۷۹۸۰۷۸

ورودی ۹۸

تمرین Word2Vec:

آدرس ریپازیتوری گیتهاب: https://github.com/danfarh/word2vec

.....

مسئله word2vec به دو روش skip-gram و cbow قابل حل است که در این تمرین با هر دو روش انجام شده است.

از آنجایی که با روش skip-gram نتوانستم به دقت و لاس مطلوبی برسم، به سراغ روش cbow رفتم و دقت ۹۷ درصد و لاس ۰.۰۵ بدست آوردم. که درباره نتایج و پیش‌بینی‌های مدل‌ها، در ادامه توضیح خواهم داد.

شرح:

در ابتدا دیتاست را با pandas می‌خوانیم و همچنین لیستstopword ها هم میخوانیم و سپس به سراغ preprocess متن می‌رویم.

با کتابخانه hazm دیتا را توکنایز میکنیم و stopword ها را از متن حذف میکنیم.

اکنون لازم است که به هر کلمه یک عدد نسبت دهیم برای این کار از Tokenizer کراس استفاده می‌کنیم با این کار همه کلمات یک آیدی به خود می‌گیرند و همچنین می‌توانیم تعداد تکرار هر کلمه در متن و تعداد کلمات یکتا را بدست آوریم.

روش skip-gram:

در روش skip-gram، به شبکه عصبی کلمه مرکز را می‌دهیم و شبکه، کلمات اطراف یا کلمات مشابه را بر می‌گرداند.

پس در ابتدا لازم است که داده‌های ورودی و خروجی را تولید کنیم و به شبکه بدهیم تا آموزش ببیند. همان‌طور که در تصویر روبرو مشاهده می‌کنید داده‌های X و y را به روشی که توضیح داده شد ایجاد کرده‌ایم.

```
def generate_data(corpus, window_size, num_unique_words):
    maxlen = window_size * 2
    all_inputs = []
    all_outputs = []
    for words in corpus:
      len_words = len(words)
      for index,w in enumerate(words):
        s = index - window_size
        e = index + window_size + 1
        for i in range(s, e):
            if i != index and 0 <= i < len_words:
              all_inputs.append(w)
              all_outputs.append(to_categorical(words[i], num_unique_words))

    return (np.array(all_inputs), np.array(all_outputs))
```

بعد از تولید داده‌های ورودی و خروجی به سراغ ایجاد شبکه‌عصبی می‌رویم.

همان‌طور که در تصویر زیر مشاهده می‌کنید در لایه اول از لایه Embedding استفاده کرده‌ایم که به تعداد کلمات یکتا نورون دارد و embed-size آن ۱۰۰ است و در لایه بعدی از Reshape استفاده کرده‌ایم و در لایه آخر هم به تعداد کلمات یکتا نورون داریم و از تابع فعالسازی softmax استفاده کرده‌ایم برای اینکه نورون‌هایی که بیشترین احتمال را به خود می‌گیرند می‌توان به عنوان کلمات مشابه درنظر گرفت.

```
embed_size=100
model = Sequential()
model.add(Embedding(input_dim=num_unique_words, output_dim=embed_size, input_length=1, embeddings_initializer='glorot_uniform'))
model.add(Reshape((embed_size, )))
model.add(Dense(num_unique_words, activation='softmax', kernel_initializer='glorot_uniform'))
```

در نهایت با ۲۰۰ ایپاک در این روش به دقت حدودا ۱۰ درصد و لاس ۳ رسیدیم.

خروجی روش skip-gram:

```
get_most_similarity('عشق')
['عشق',
 'جان',
 'دل',
 'عقل',
 'آتش',
 'شاه',
 'جمله',
 'شمس',
 'عاشقان',
 'دست',
 'مست',
 'جهان',
 'گشت',
 'ملک',
 'روح']
```

```
get_most_similarity('شمس')
['تبریزی',
 'تبریز',
 'دین',
 'الدین',
 'مفخر',
 'الحق',
 'جان',
 'عشق',
 'خداوند',
 'نور',
 'الضحی',
 'دل',
 'ذره',
 'مخدوم',
 'خورشید']
```

```
get_most_similarity('گنج')
['عاشق',
 'درآرد',
 'زندان',
 'ای',
 'نرسم',
 'نشین',
 'گنجی',
 'خلا',
 'گنج',
 'مشین',
 'دمی',
 'بیدار',
 'کرده',
 'مطلق',
 'نه']
```

```
get_most_similarity('گل')
```

```
['آب',
 'خار',
 'گل',
 'بلبل',
 'دل',
 'باغ',
 'یار',
 'ریحان',
 'آتش',
 'گلزار',
 'نسرین',
 'رعنا',
 'چمن',
 'شکر',
 'رود']
```

```
get_most_similarity('مست')
```

```
['مست',
 'چشم',
 'جان',
 'آمدست',
 'خاک',
 'عشق',
 'عقل',
 'خراب',
 'ره',
 'دست',
 'خواجه',
 'جام',
 'یار',
 'جمله',
 'مخمور']
```

```
get_most_similarity('دست')
```

```
['دست',
 'جان',
 'دل',
 'پا',
 'پای',
 'عشق',
 'جام',
 'دهان',
 'مست',
 'شب',
 'شمس',
 'کف',
 'دلم',
 'باده',
 'یار']
```

روش CBOW:

روش cbow دقیقا برعکس skip-gram است. به شبکه عصبی کلمات اطراف را می‌دهیم و کلمه مرکز را به عنوان خروجی به ما می‌دهد.

پس در ابتدا لازم است که داده‌های ورودی و خروجی را تولید کنیم و به شبکه بدهیم تا آموزش ببیند. در تصویر روبرو نحوه ایجاد داده‌های X و y را مشاهده می‌کنید

.

```python
# CBOW
def generate_data(corpus, window_size, num_unique_words):
    all_inputs = []
    all_outputs = []

    for sentence in corpus:
        L = len(sentence)
        for index, word in enumerate(sentence):
            start = index - window_size
            end = index + window_size + 1

            context_words = []
            for i in range(start, end):
                if i != index:
                    if 0 <= i < L:
                        context_words.append(sentence[i])
                    else:
                        context_words.append(0)
            all_inputs.append(context_words)
            all_outputs.append(to_categorical(word, num_unique_words))

    return (np.array(all_inputs), np.array(all_outputs))
```

بعد از تولید داده‌های ورودی و خروجی به سراغ ایجاد شبکه‌عصبی می‌رویم.

همان‌طور که در تصویر زیر مشاهده می‌کنید در لایه اول از لایه Embedding استفاده کرده‌ایم که به تعداد کلمات یکتا نورون دارد و embed-size آن ۵۰ است و input_length به اندازه دو برابر window_size است زیرا همان‌طور که توضیح داده شد در این روش کلمات اطراف را به عنوان ورودی به شبکه می‌دهیم. در لایه بعدی از Lambda استفاده کرده‌ایم و در لایه آخر هم به تعداد کلمات یکتا نورون داریم و از تابع فعالسازی softmax استفاده کرده‌ایم برای اینکه نورونی که بیشترین احتمال را به خود می‌گیرند می‌توان به عنوان کلمه‌ی مرکز پیش‌بینی شود.

بعد از این این‌که مدل به خوبی یاد گرفت وزن‌های لایه embedding را می‌گیریم. اکنون بردار هر کلمه را داریم خیلی راحت می‌توانیم فاصله کسینوسی کلمات را بدست آوریم و کلماتی که نزدیک هم هستند، کلمات مشابه خواهند بود.

```
embed_size=50
model = Sequential()
model.add(Embedding(input_dim=num_unique_words,
                    output_dim=embed_size,
                    input_length=window_size*2,
                    name="embedding_layer",
                    embeddings_initializer='glorot_uniform'))
model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size, )))
model.add(Dense(num_unique_words, activation='softmax', kernel_initializer='glorot_uniform'))
```

در نهایت با ۳۰۰ ایپاک در این روش به دقت ۹۷ درصد و لاس ۰.۰۵ رسیدیم.

سپس مدل آموزش دیده را ذخیره کرده و برای پیدا کردن کلمات مشابه مدل را لود میکنیم و با وزن‌های لایه embedding بردار هر کلمه را داریم سپس تابعی می‌نویسیم که فاصله کسینوسی کلمات نسبت به هم را بدست آورد و n کلمه مشابه اول را به عنوان خروجی برگرداند.

بدست آوردن بردار هر کلمه:

برای بدست آوردن بردار هر کلمه تابعی نوشته شده است که یک کلمه به عنوان ورودی می‌گیرد و بردار آن کلمه را به عنوان خروجی برمی‌گرداند.

```
get_word_vector('مه')
```

```
array([-1.2564193 ,  0.9754958 , -0.0413386 ,  0.6978016 , -0.3388974 ,
       -1.0896406 ,  0.79556346, -0.0146776 ,  0.24062763,  0.5892331 ,
        0.19248633,  0.13684535, -0.69566256, -0.89324903,  0.72135955,
        0.5938318 ,  0.8202653 , -0.8655508 ,  0.8860461 ,  0.61623704,
        1.1615644 ,  0.69691944, -0.06205862,  0.6632749 ,  0.57734853,
       -0.12747145,  0.38678074,  0.2045722 ,  1.0815064 ,  0.55405074,
       -0.5492463 , -0.29469597,  0.36800307, -0.04071516,  0.04787865,
        1.5306802 ,  0.21228653,  0.73807913, -1.1649163 ,  0.13621949,
       -0.9588356 ,  0.69721293, -1.0329196 , -1.1614479 ,  0.39543965,
       -0.7699773 ,  0.8095429 ,  0.4698367 , -0.46544254,  0.51259494],
      dtype=float32)
```

خروجی روش CBOW :

```
get_most_similarity('گل')
```

```
['گل',
 'خار',
 'سرو',
 'بلبل',
 'باغ',
 'بهار',
 'سبزه',
 'چمن',
 'درخت',
 'گلستان',
 'شاخ',
 'سلام',
 'لاله',
 'گلزار',
 'ریحان',
 'بوی',
 'نفسی',
 'گلشن',
 'بام',
 'زمین']
```

```
get_most_similarity('مست')
```

```
['مست',
 'شراب',
 'خمار',
 'مستان',
 'جام',
 'خواجه',
 'آمدست',
 'قضا',
 'شیوه',
 'اسرار',
 'پیر',
 'دستار',
 'اینست',
 'مخمور',
 'ساقیا',
 'بند',
 'توام',
 'گردان',
 'درده',
 'مغز']
```

```
get_most_similarity('دست')
```

```
['دست',
 'پای',
 'کف',
 'دهان',
 'مشین',
 'قدح',
 'گم',
 'بزن',
 'یا',
 'نظر',
 'مسیح',
 'دزد',
 'زن',
 'دستی',
 'سفر',
 'باقی',
 'قضا',
 'هزاران',
 'ساغر',
 'عصا']
```

```
get_most_similarity('مه')
```

```
['مه',
 'ماه',
 'رخ',
 'آفتاب',
 'فلک',
 'خورشید',
 'لقا',
 'جمال',
 'سجده',
 'نهان',
 'تابان',
 'سایه',
 'نقاب',
 'پیشت',
 'سحر',
 'خوبان',
 'شمع',
 'علیک',
 'خوبی',
 'رنگ']
```

```
get_most_similarity('نور')
```

```
['نور',
 'سایه',
 'شمع',
 'فلک',
 'علیک',
 'آفتاب',
 'خورشید',
 'لقا',
 'پروانه',
 'قد',
 'اصل',
 'عروسی',
 'بفعل',
 'سور',
 'نقاب',
 'لحظه',
 'خوبان',
 'الله',
 'خوبی',
 'جمع']
```

```
get_most_similarity('غم')
```

```
['غم',
 'شادی',
 'رود',
 'تست',
 'نیم',
 'خلق',
 'طلب',
 'جانی',
 'شهر',
 'اندوه',
 'خرد',
 'سوز',
 'جنگ',
 'تنگ',
 'شاد',
 'زاده',
 'آبی',
 'بستان',
 'گره',
 'طرب']
```

```
get_most_similarity('آسمان')
```

```
['آسمان',
 'زمین',
 'چرخ',
 'گلستان',
 'سما',
 'زنان',
 'قمر',
 'باغ',
 'بستو',
 'کای',
 'دود',
 'ندا',
 'طرب',
 'قرین',
 'بنگر',
 'مشک',
 'درخت',
 'سیه',
 'برگ',
 'توبه']
```

```
get_most_similarity('آتش')
```

```
['آتش',
 'دود',
 'سما',
 'آتشی',
 'مرده',
 'عیسی',
 'شوق',
 'تنگ',
 'موسی',
 'دان',
 'کوه',
 'شعله',
 'اندیشه',
 'صافی',
 'خاموش',
 'طلب',
 'مفرش',
 'زنده',
 'سوز',
 'تک']
```

```
get_most_similarity('آب')
```

```
['آب',
 'چشمه',
 'جوی',
 'جو',
 'روان',
 'حیات',
 'سنگ',
 'حیوان',
 'صفت',
 'آسیا',
 'مرده',
 'خضر',
 'صفا',
 'زمین',
 'جانی',
 'تشنه',
 'است',
 'رود',
 'آهن',
 'جگر']
```

وبسایت پروژه:

برای وبسایت پروژه از کتابخانه streamlit استفاده کردم که وزن‌های مدل train شده‌ام را لود کردم و همچین با استفاده از PCA ابعاد آن را به ۳ کاهش دادم و نمودار ۳ بعدی آن را رسم کرده‌ام.

سپس با توجه به وزن‌ها، فاصله آنها را با استفاده از تابعی که در کد مشاهده می‌کنید، محاسبه کرده‌ام و در یک نمودار ترسیم شده‌اند.

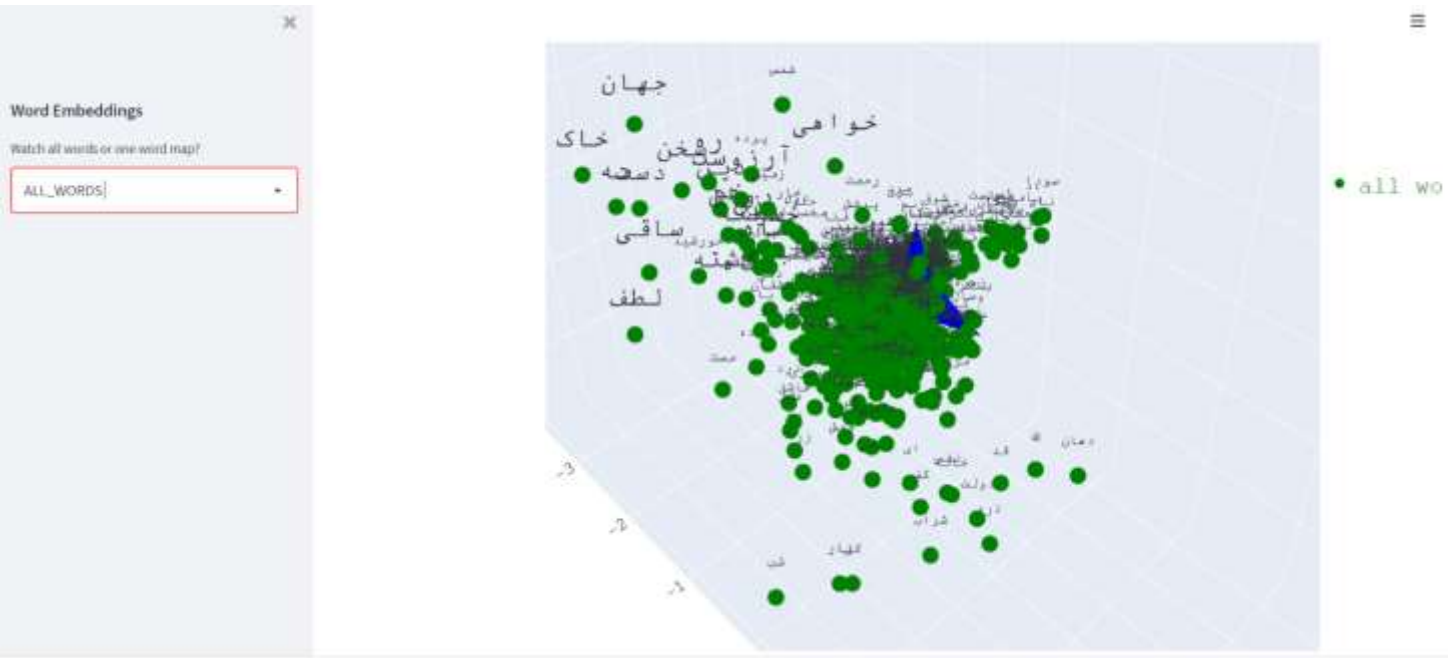
مراحل اجرای برنامه:

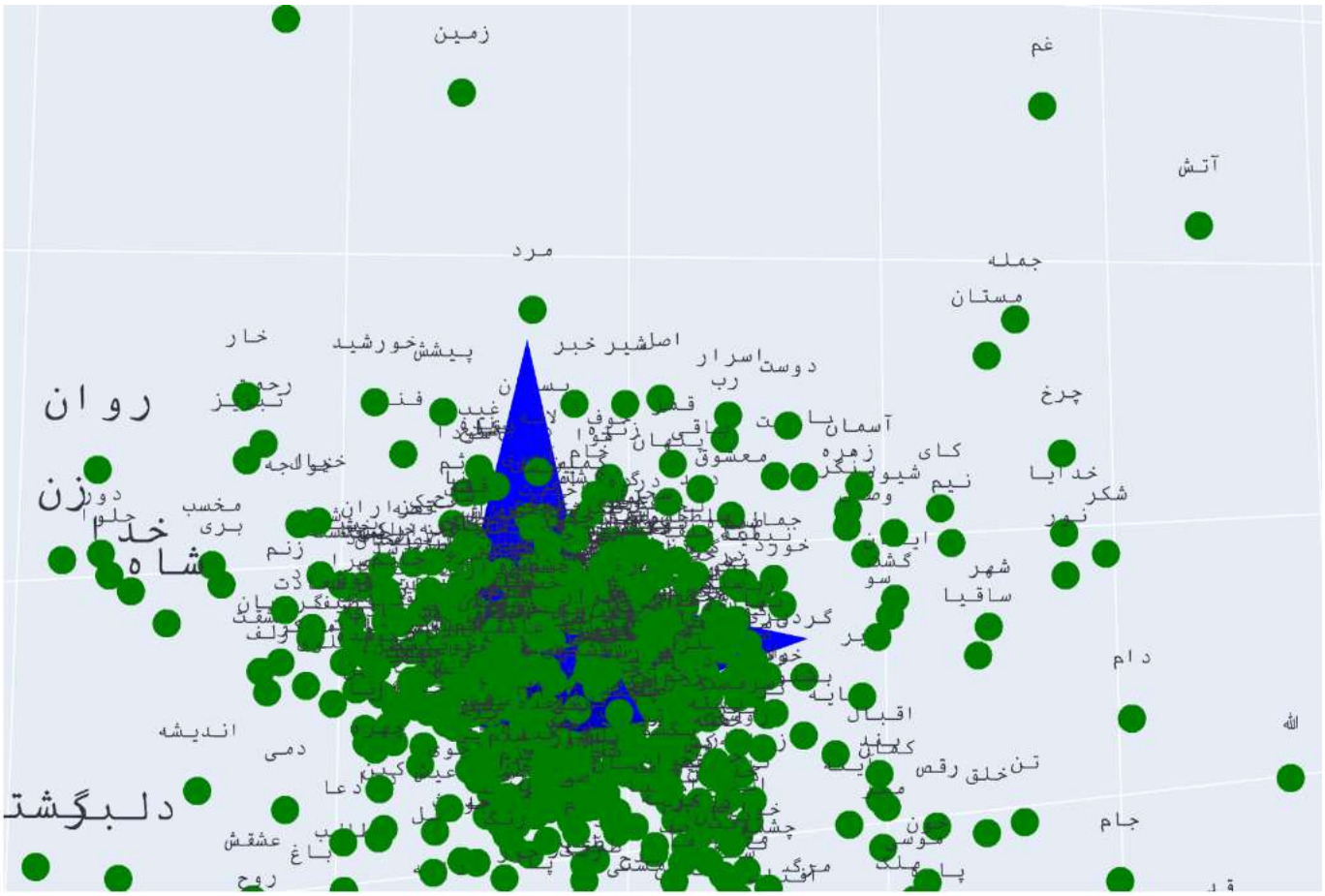
برای اجرا کردن کد کافی است در ابتدا یک محیط مجازی بسازید و سپس تمام کتابخانه‌هایی که در requirements.txt قرار دارند را نصب کنید و برای ران کردن کد هم دستور زیر را بزنید:

```
streamlit run main.py
```

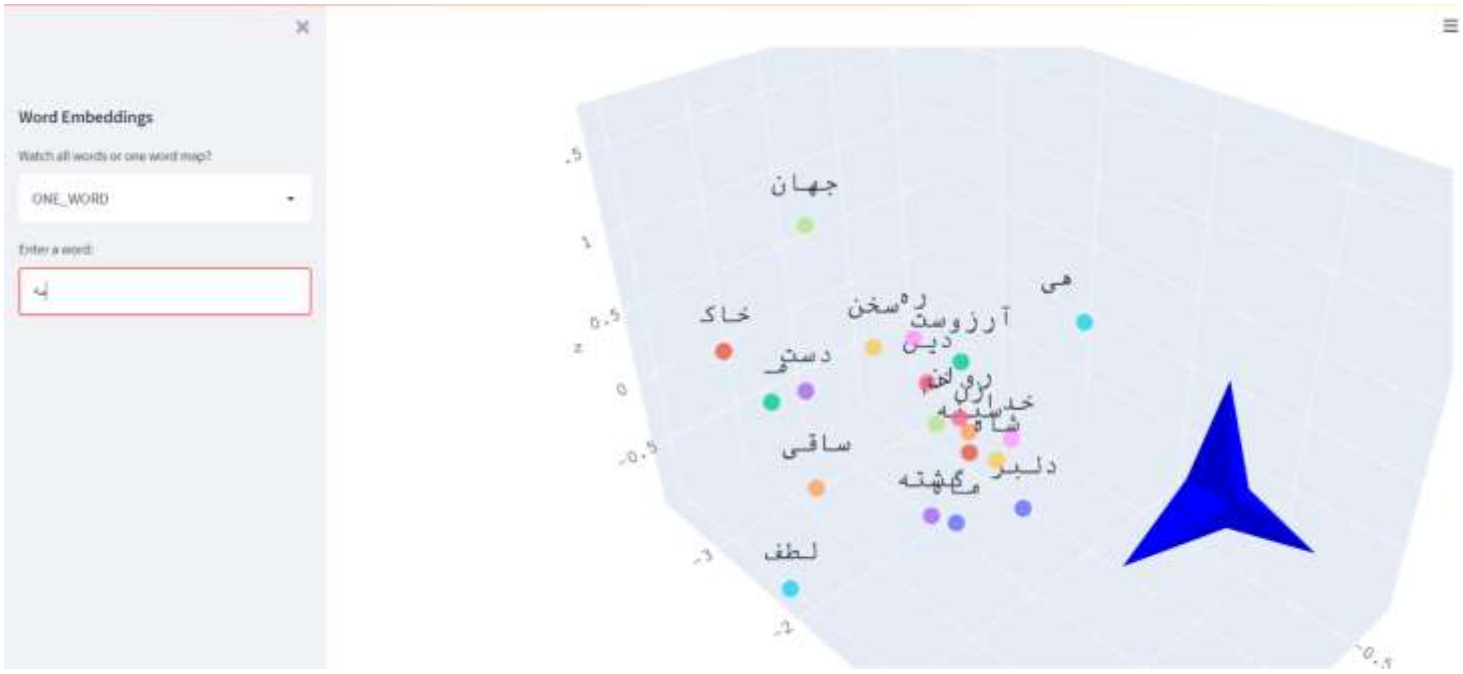
قسمت های سایت:

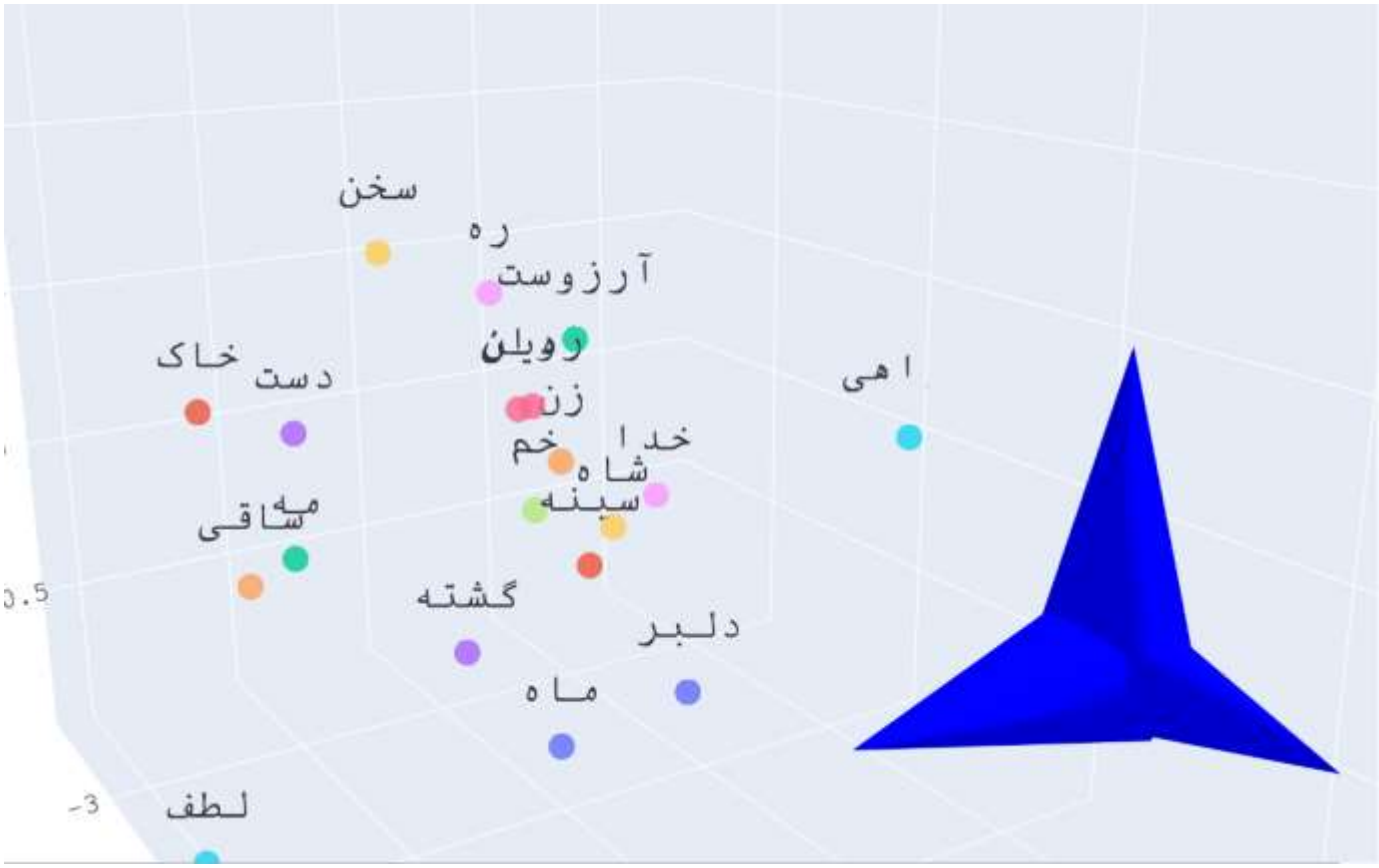
همان‌طور که مشاهده می‌کنید وبسایت دو قسمت دارد قسمت اول نقشه لغات تمام کلمات را بدون کندی و با قابلیت بزرگنمایی مثبت و منفی، به صورت سه بعدی رسم می‌کند.


Word Embeddings
Watch all words or one word map!
ALL_WORDS
جهان
خاک
خواهی
ساقی
لطف
all wo


زمین
غم
آتش
مرد
جمله
مستان
خار
پیشش
خورشید
اصلشیر
خبر
اسرار
دوست
رب
روان
چرخ
آسمان
کای
نیم
خدایا
شکر
مخسب
بری
زنم
شهر
ساقیا
دام
اقبال
اندیشه
دمی
خلق
رقص
جام
دلبگشت
عشقش
باغ
روح

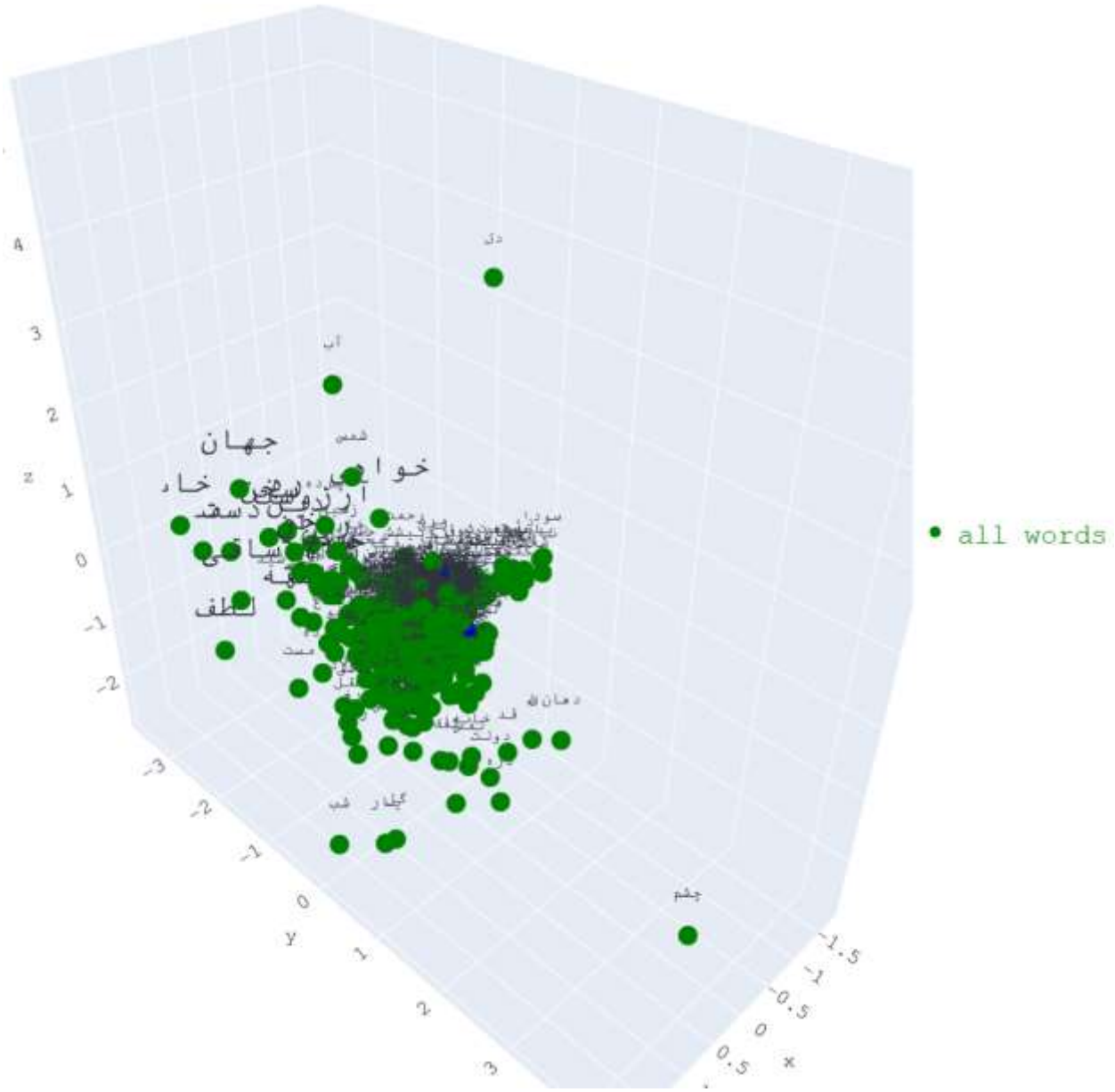
همانطور که مشاهده میکنید میتوان یک کلمه از کاربر گرفت و ۱۵ کلمات اطراف آن را نمایش داد.

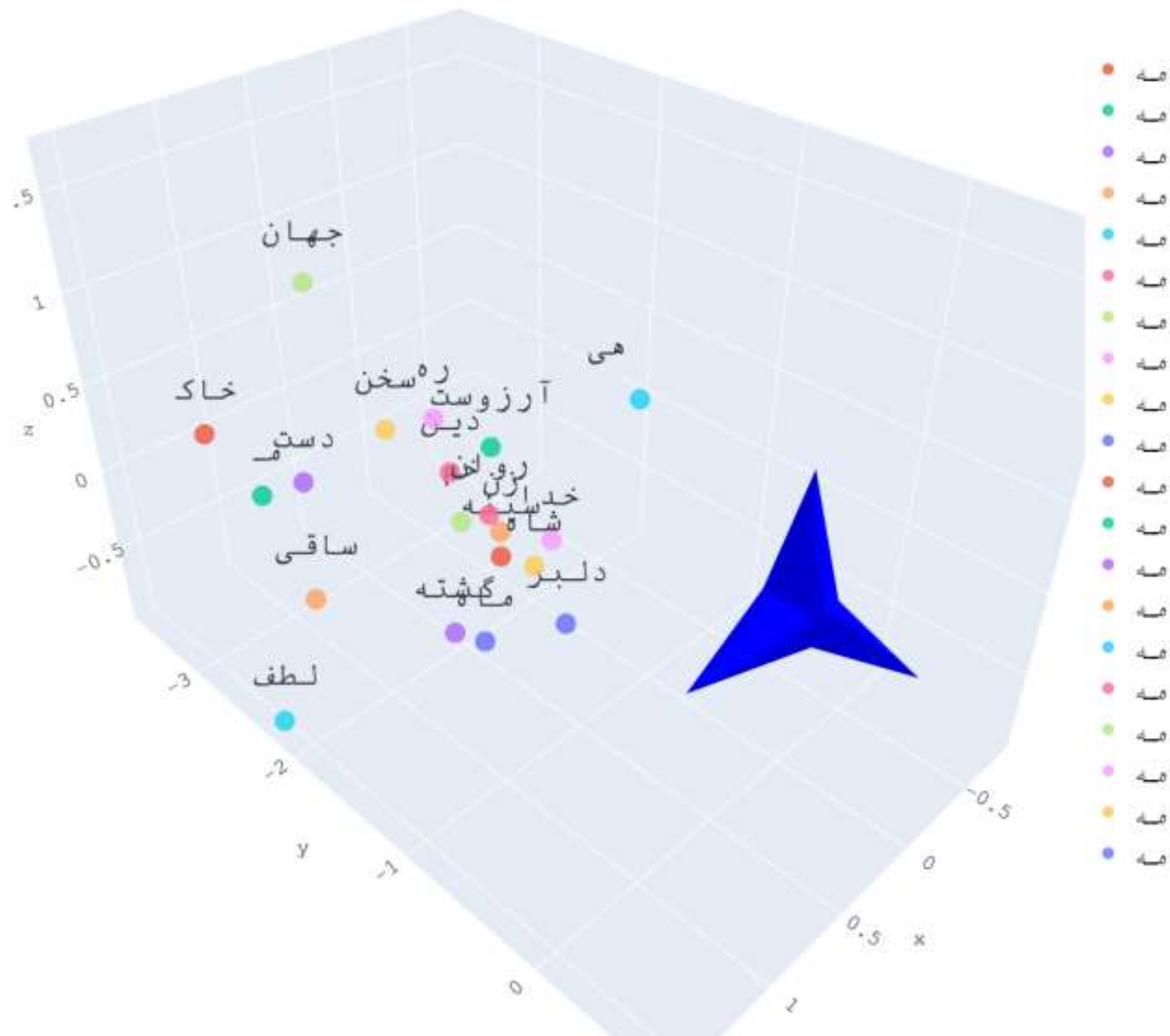




همچنین میتوان از نقشه لغات خروجی گرفت.

برای مثال از دو نمودار ها خروجی png. گرفته ام.




جهان
هی
خاک
ساقی
لطف
دلبر
x
y
z