به نام خدا

عنوان: پیاده‌سازی word۲vec و visualization

درس: NLP

استاد درس: مهندس قاسمی

مهدی دهقانی

۲۲۰۷۹۷۰۴۸

mehdi.dehghani@ut.ac.ir

mahdiazadi18@yahoo.com

داده‌های ورودی

داده‌های ورودی درون پوشه Data قرار می‌گیرند. هر چقدر داده‌های ورودی بزرگتر باشند، خروجی بهتری حاصل خواهد شد.

برای این قسمت دو داده fa.fooladvand.txt و voa_fa_۲۰۰۳-۲۰۰۸_orig.txt در نظر گرفته شده است که اولی بخاطر گوچک بودن خروجی بسیار خوبی را فراهم نمی‌کند اما دومی به نسبت اندازه داده‌ها خروجی بسیار خوبی خواهد داد.

پیش پردازش داده‌ها

برای پیش پردازش داده‌های ورودی، ساده‌ترین راه حذف تمام حرف‌ها به‌جز حروف الفبای زبان مربوطه است. که می‌توان با استفاده از قسمت کامنت شده برای just words و استفاده نکردن از بقیه قسمت‌ها تنها کلمات فارسی را نگاه داشت.

می‌توان به جای اینکه تنها کلمات را نگه داشت از راه بهتری استفاده کرد

اما می‌توان از این جلوتر رفت و به جای اینکه تنها کلمات را نگه داشت از راه بهتری استفاده کرد و بعضی از علائم نگارشی و مواردی نظیر آن که به پردازش کمک می‌کند را نگه داشت و از کتابخانه‌هایی که در این زمینه وجود دارد استفاده کرد تا داده‌های ورودی برای اعمال word۲vec شکل بهتری داشته باشند. در این قسمت از کتابخانه hazm یا همان هضم استفاده شده است که بنظر برای زبان فارسی نسبتا مناسب است. می‌توان با استفاده از هضم و متدهایی مثل Normalizer یا InformalNor-malizer متن ورودی را نسبت به زبان فارسی نرمال‌سازی کرد (به طور مثال فعلی مانند داده خواهد شد به داده_خواهد_شد تبدیل می‌شود که این یعنی به جای ۳ توکن به ۱ توکن تبدیل می‌شود که بسیار به ما کمک خواهد کرد)

در ادامه می‌توان stop words زبان مربوطه را از داده‌های نرمال‌سازی شده حذف کرد که این قسمت کامنت شده است ولی قابل استفاده است و حتی می‌توان از stop words های ارائه شده دیگر استفاده کرد ولی در حال حاضر stop words ای که خوده هضم ارائه داده است در این قسمت قرار داده شده است.

در ادامه می‌توان از stem یا lemmatize استفاده کرد تا کلماتی مثل کتاب‌ها، کتاب‌های و کتاب به عنوان سه کلمه جدا در نظر گرفته نشوند و همه به عنوان یک کلمه کتاب در نظر گرفته شوند. در این قسمت نیز از هضم استفاده شده است اما به دلیل اینکه در بعضی از موارد عملکرد خوبی از این دو متد گرفته نشد (به طور مثال کلمه داشت به دا تبدیل می‌شد یا رسمی به رسم) تصمیم بر این شد که روی داده کوچک‌تر استفاده نشود اما این دو متد در نظر گرفته شده و فقط در قسمت اجرایی کامنت شده است و می‌توان از آن‌ها نیز استفاده نمود.

در نهایت داده‌هایی که پیش پردازش شده‌اند و به صورت لیستی از توکن‌ها در اختیار ما هستند را در فایل جدیدی با پسوند _preprocessed می‌نویسیم. به این صورت که بین هر توکن یک فاصله قرار می‌دهیم تا برای بعد به سادگی قابل خواندن از فایل باشند. همچنین خطوطی که خالی از کلمات فارسی هستند نیز قبل از این حذف شده و در نظر گرفته نشده‌اند تا مشکلی به وجود نیاید.

word۲vec

برای پیاده‌سازی word۲vec از tensorflow و pytorch استفاده کردیم که در pytorch تنها از روش skip gram و negative sampling استفاده شده است اما در tensorflow هر ۴ روش مربوط به word۲vec شامل موارد زیر پیاده‌سازی شده است:

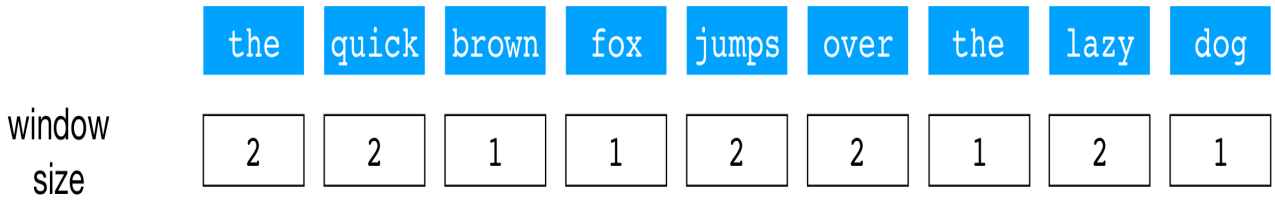
- skip gram with negative sampling
- skip gram with hierarchical softmax
- cbow with negative sampling
- cbow with hierarchical softmax

ساخت dataset pipeline

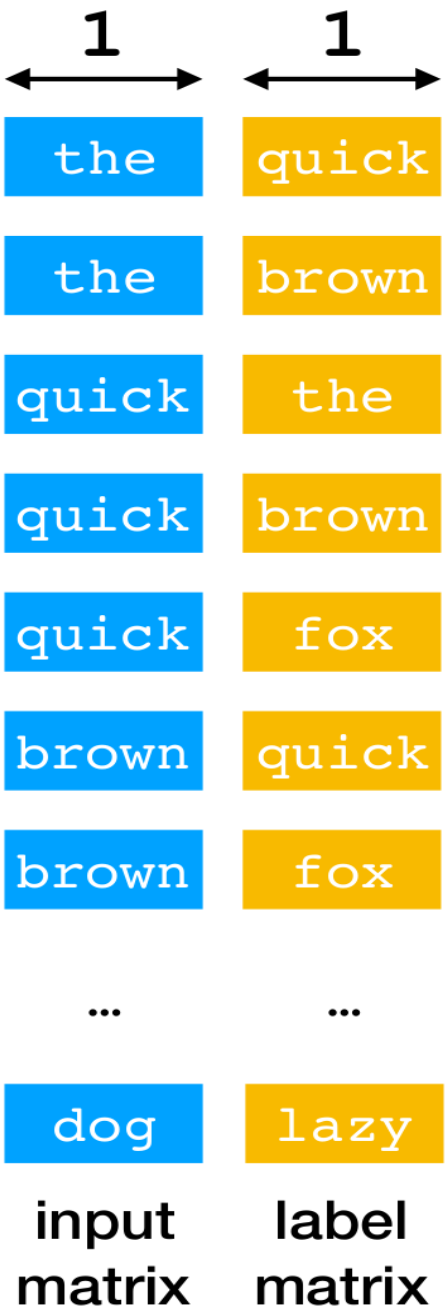
در ادامه یک مثال عینی از تبدیل یک جمله خام به ماتریس‌هایی که داده‌ها را برای آموزش مدل word۲vec با معماری skip_gram یا cbow نگه می‌دارند را می‌بینیم.

فرض کنید یک جمله در corpus داریم: the quick brown fox jumps over the lazy dog، با اندازه پنجره (حداکثر تعداد کلمات در سمت چپ یا راست کلمه مورد نظر) در زیر کلمات. فرض کنید جمله قبلاً subsample شده است و کلمات به شاخص‌ها نگاشت شده‌اند.

هر یک از کلمات در جمله را target word می‌نامیم و کلمات درون پنجره که در مرکز آن‌ها target word قرار دارد را context words می‌نامیم. به عنوان مثال، 'quick' و 'brown'، 'the' و 'the'، target word با context words هستند و 'brown' و 'fox'، target word با context words، 'quick' هستند.

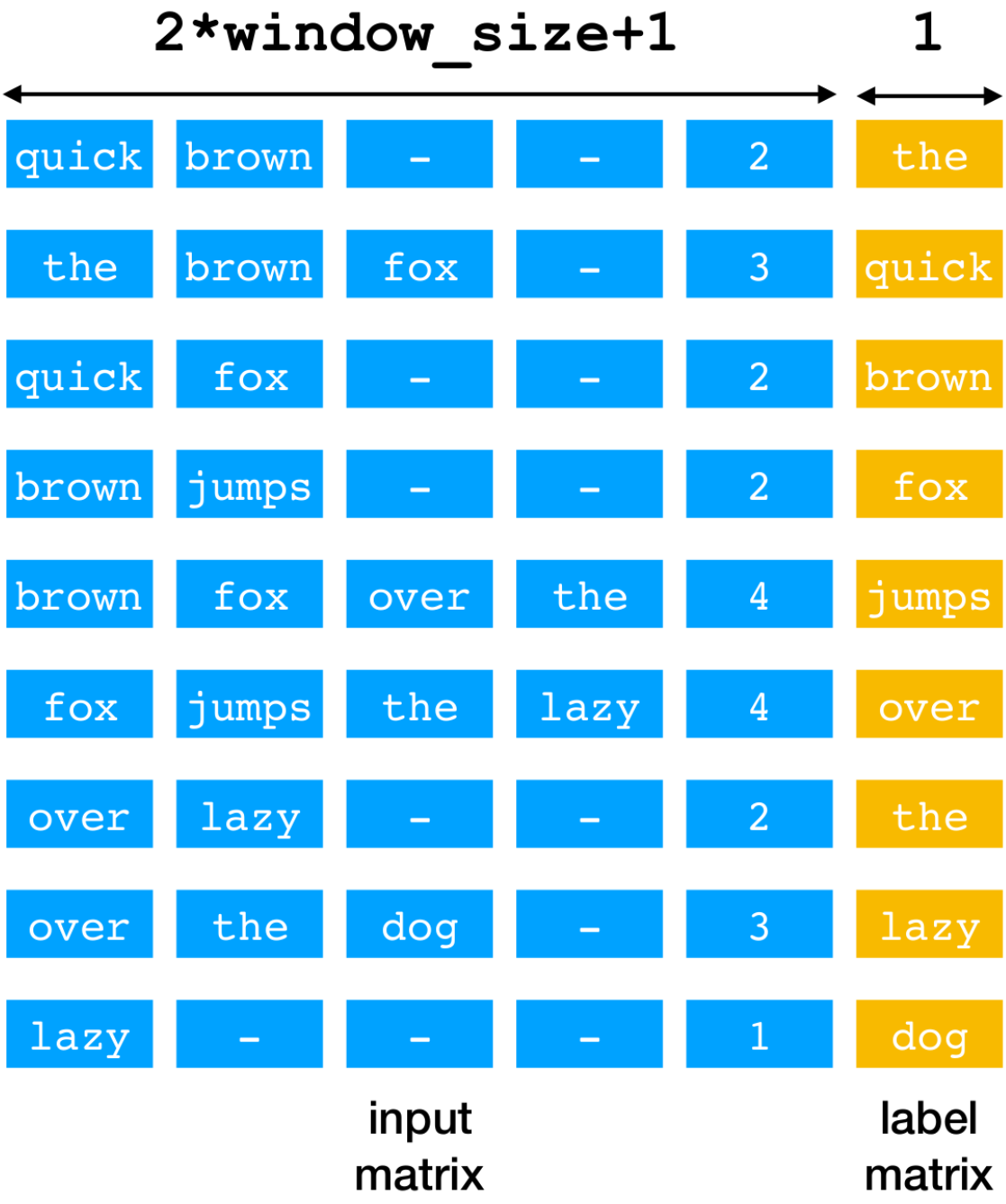


برای skip_gram، وظیفه پیش‌بینی context words با توجه به target word است. شاخص هر target word به سادگی تکرار می‌شود تا با تعداد context words آن مطابقت داشته باشد. این ماتریس ورودی ما خواهد بود.



Skip gram with negative sampling

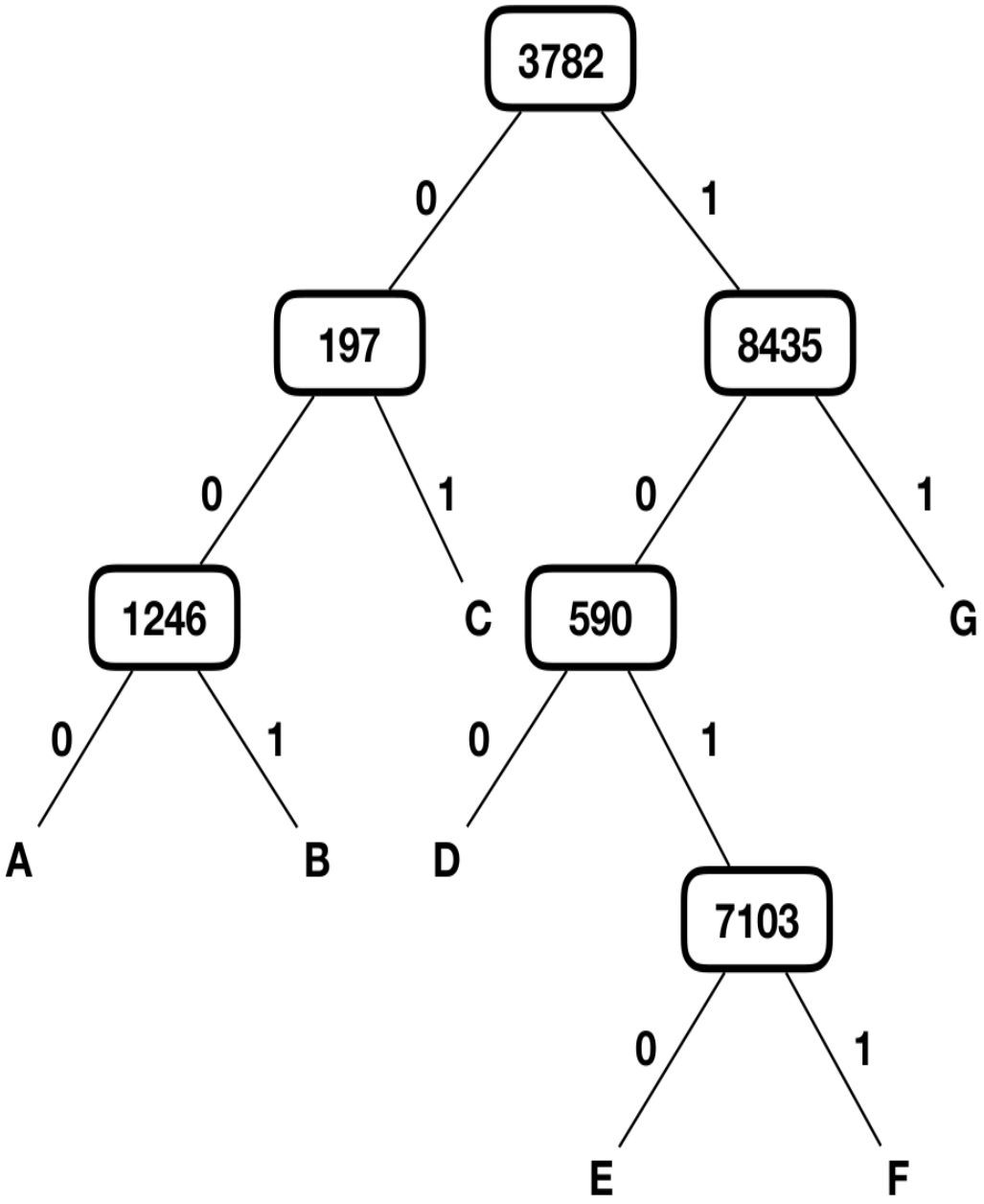
برای cbow، وظیفه پیش‌بینی target word با توجه به context words است. از آنجایی که هر target word ممکن است دارای تعداد متغیری از context words باشد، لیست context words را به حداکثر اندازه ممکن (اندازه پنجره * ۲) می‌رسانیم (pad می‌کنیم) و اندازه واقعی target word را اضافه می‌کنیم.



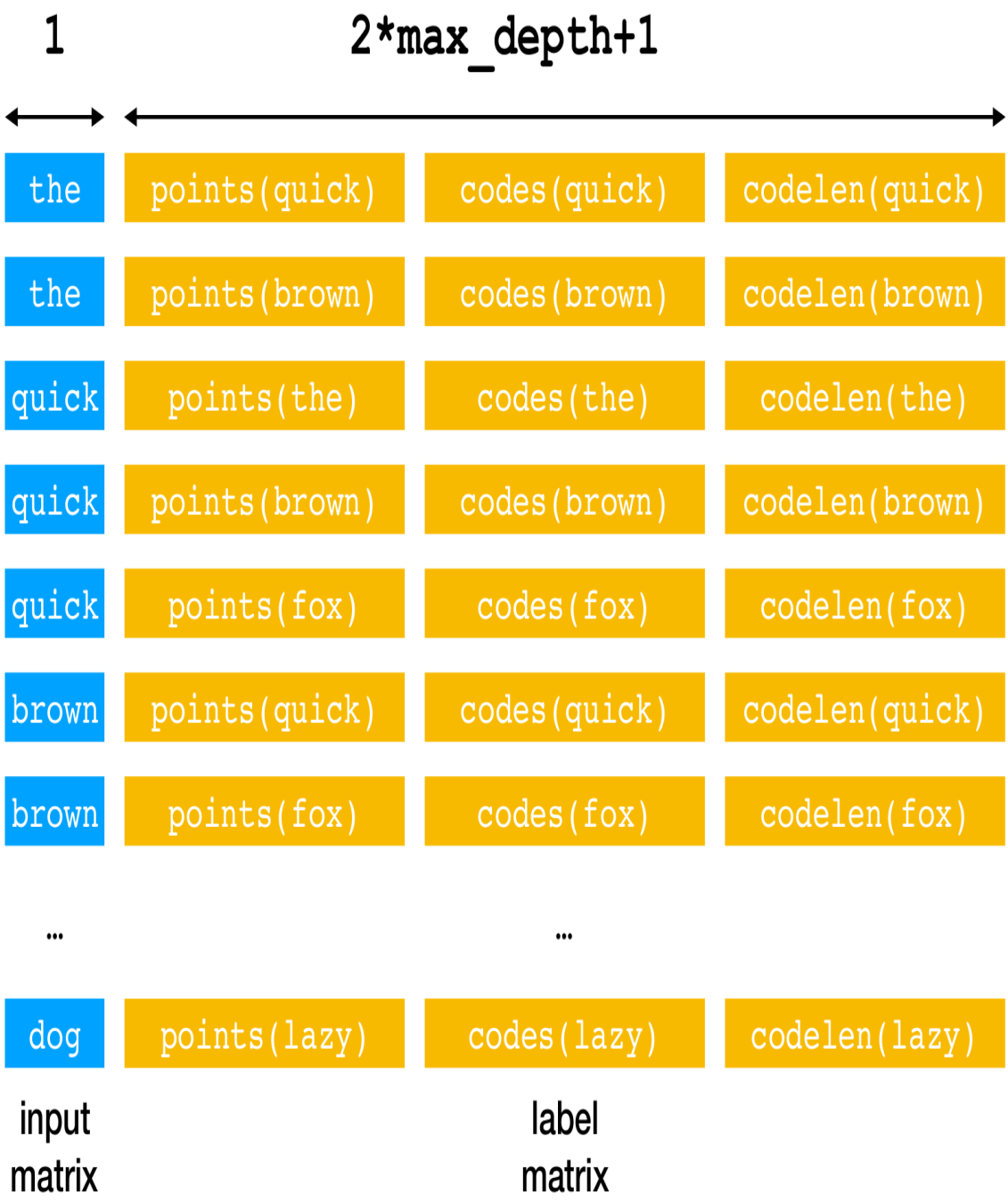
Continuous bag of words with negative sampling

اگر الگوریتم آموزشی negative_sampling باشد، ما به سادگی ماتریس برچسب را با شاخص‌های کلماتی که باید پیش‌بینی شوند پر می‌کنیم: context words برای skip_gram یا target words برای cbow.

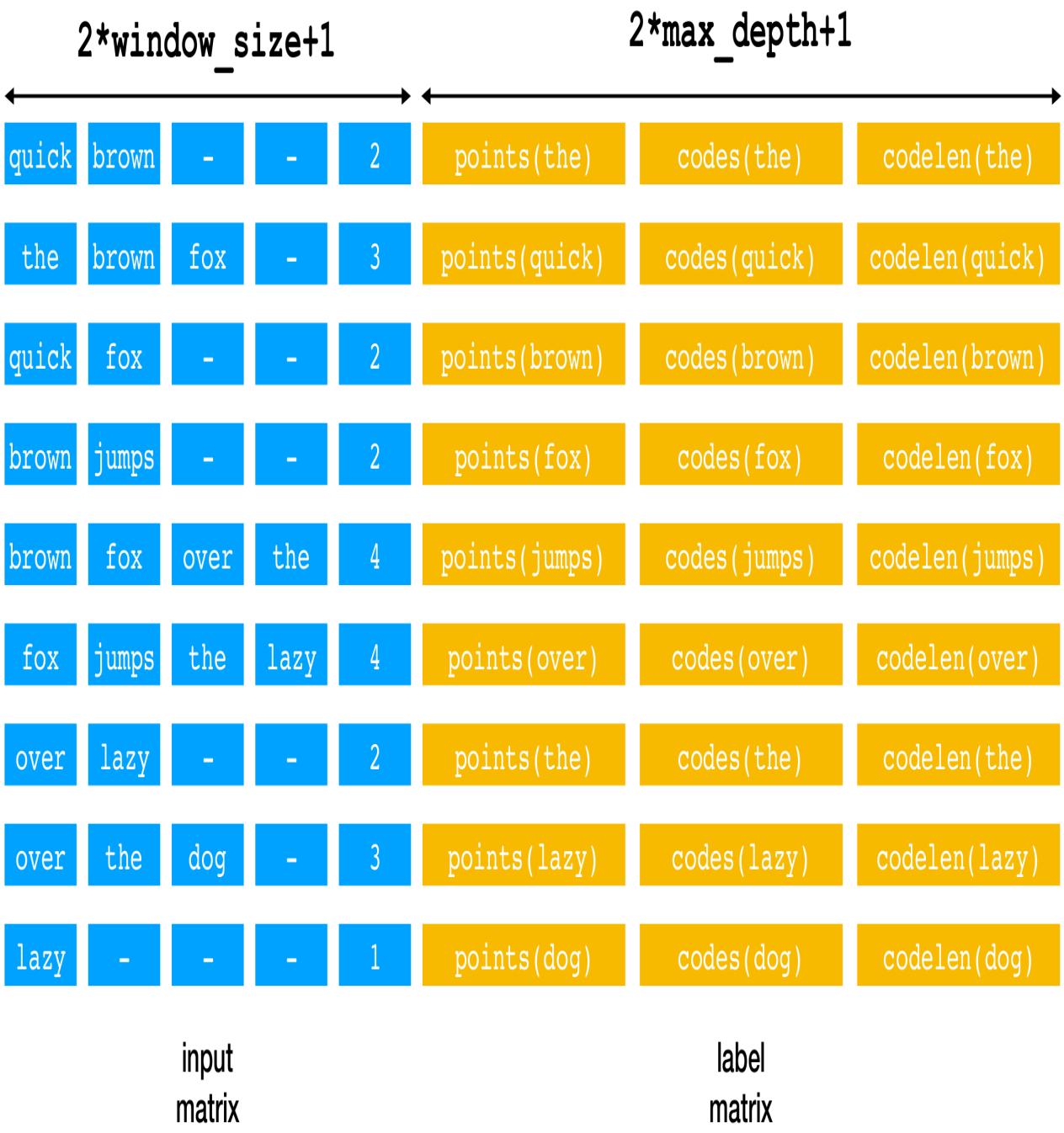
اگر الگوریتم آموزشی hierarchical_softmax باشد، یک درخت هافمن برای مجموعه کلمات واژگان ساخته می‌شود. هر کلمه واژگان دقیقاً با یک گره برگ مرتبط است و کلماتی که در negative_sampling باید پیش‌بینی شوند، با دنباله‌ای از کدها و point ها جایگزین می‌شوند که توسط گره‌های داخلی در مسیر ریشه به برگ تعیین می‌شوند. به عنوان مثال، کدها و point های E عبارتند از ۳۷۸۲، ۸۴۳۵، ۵۹۰، ۷۱۰۳ و ۱، ۰، ۱، ۰. ماتریس برچسب را با کدها و point های پر (pad) شده (تا حداکثر عمق)، همراه با طول واقعی کدها یا point ها پر می‌کنیم.



Huffman Tree



Skip gram with hierarchical softmax



Continuous bag of words with hierarchical softmax

به طور خلاصه، یک ماتریس ورودی و یک ماتریس برچسب از یک جمله ورودی خام ایجاد می‌شود که اطلاعات ورودی و برچسب را برای کار پیش‌بینی فراهم می‌کند.

تست و خروجی‌ها

برای بعضی از توابع و متدها قسمتی به عنوان test نوشته شده است که بر روی مثال خاصی، آن توابع یا متدها را تست می‌کند و همچنین بعد از مقداردهی پارامترهای مربوطه برای هرکدام از pytorch یا tensorflow و اجرای آن‌ها، به ازای هر چند مرحله، از مقدار فعلی لرنینگ ریت و لاس (Average یا Total) در آن مرحله خروجی گرفته شده و نمایش داده می‌شود. همچنین برای اجرای pytorch از اسکریپت word۲vec.py و برای اجرای tensorflow از اسکریپت run_training.py استفاده می‌شود. بدیهی است که به ازای مقادیر مختلف پارامترهای در نظر گرفته شده برای هرکدام، خروجی متفاوت خواهد بود.

ساخت فایل‌های خروجی

در هر دو pytorch و tensorflow، پس از اجرای اسکریپت‌های گفته شده در قسمت قبلی که روی corpus داده شده، آموزش انجام می‌دهند، از اسکریپت create_files.py استفاده می‌کنیم تا فایل‌های مورد نیاز برای تست یا شبیه‌سازی‌های مورد نظر، نظیر فایل‌های tsv برای projector و words-vectors برای ساخت گراف word۲vec و در نهایت اجرای webapp، ساخته شوند.

فایل‌های خروجی درون پوشه‌های out_word۲vec_pytorch برای pytorch و out_word۲vec_tensorflow برای tensorflow، قرار می‌گیرند.

نزدیک‌ترین همسایه‌ها

در هر دو pytorch و tensorflow، اسکریپتی با عنوان nearest_neighbors.py وجود دارد که پس از انجام دادن عملیات‌های قبلی و ساخت فایل‌های خروجی، می‌توان این اسکریپت را اجرا کرد تا برای ۱۰ کلمه از واژگان به صورت تصادفی، برای هر کلمه، ۲۰ کلمه‌ای که از بقیه به آن نزدیک‌تر هستند نمایش داده شود.

مثال برای کلمه 'عذاب':

```
nearest neighbors to عذاب:
(0.4668243 ,'عذابشان')
(0.4197559 ,'بچشید')
(0.37538737 ,'حفتu200cآور')
(0.36143243 ,'دچار')
(0.36032143 ,'شکنجه')
(0.35574484 ,'عذابی')
(0.3442301 ,'مرگ')
(0.3435179 ,'دردناک')
(0.3403393 ,'رسوایی')
(0.3319929 ,'میu200cترسم')
(0.3288855 ,'تهدید')
(0.31565815 ,'مراقب')
(0.31492293 ,'کیفر')
(0.31404504 ,'داده_میu200cشوند')
(0.31128582 ,'همانند')
(0.3107616 ,'واجب')
(0.3062084 ,'عذابش')
(0.3027309 ,'اعمال')
(0.30270836 ,'گرفتار')
(0.30194914 ,'منافقان')
------------------------------
```

شبیه‌سازی در projector

برای شبیه‌سازی در projector که tensorflow در نظر گرفته است، می‌توان فایل‌های خروجی tsv که در قسمت‌های قبل گفته شد را در لینک زیر، آپلود کرد.

https://projector.tensorflow.org/

شبیه‌سازی برای چند نمونه تست که در فایل‌های پروژه نیز موجود است:

word2vec_tensorflow:

= batch_size 1000 = epochs 'negative_sampling' = algm 'skip_gram' = arch 2 = window_size 1e-3 = sample 3 = min_count 0 = max_vocab_size 1024 = min_alpha 025.0 = alpha 75.0 = power 2 = negatives 100 = hidden_size 10000 = log_per_steps True = add_bias 0001.0

https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/mehdidn/dfb8ed30d60b0862c8a9c1cc6b4a8c6e/raw/443c8faf04d924f884bfcf1e29af14800a240cc9/word2vec_tensorflow_projector

word2vec_pytorch:

negatives=2 window_size=2 batch_size=1024 emb_dimension=100 min_count=3 initial_lr=0.025 iteration=1000

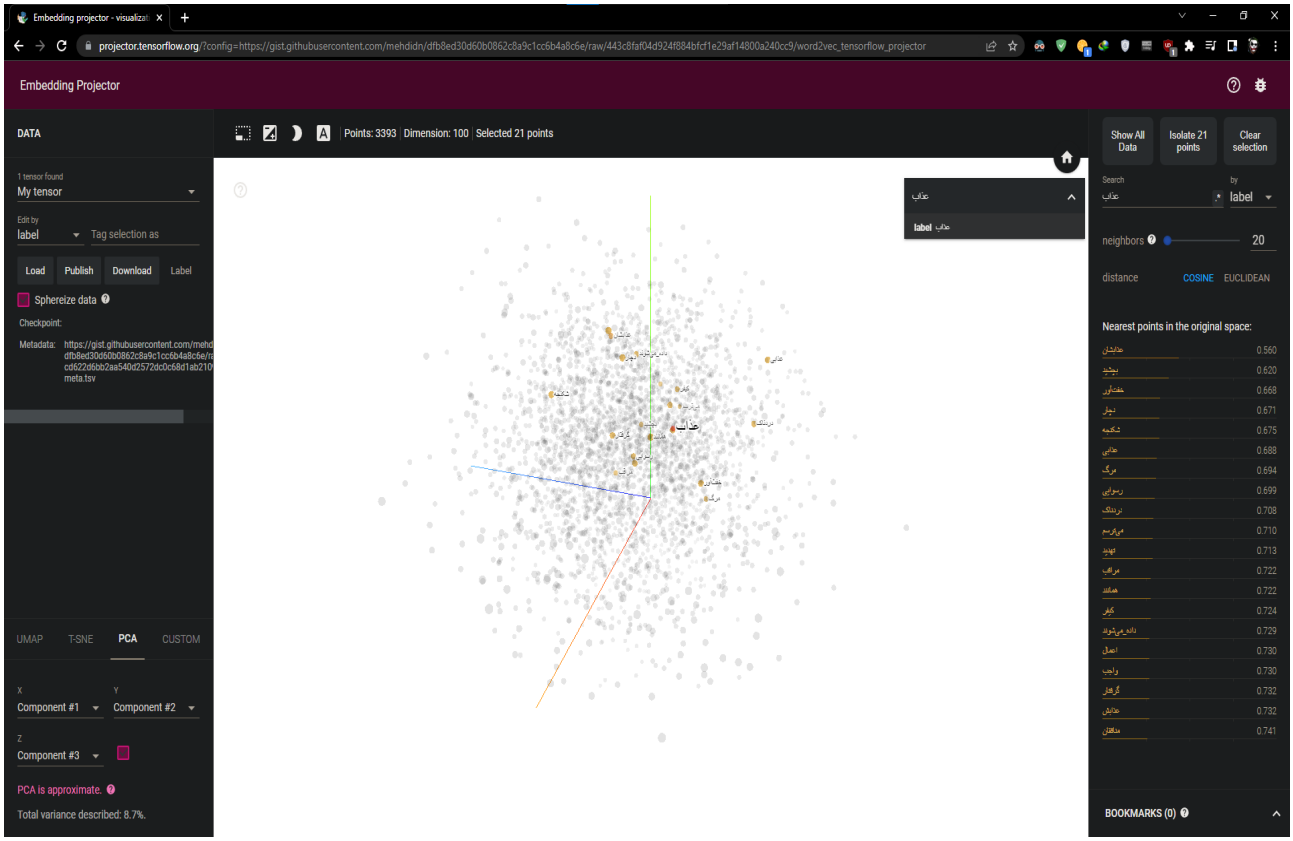
https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/mehdidn/fbbf5cb443eb50751c6a1bea9306bead/raw/805a17f300285aa454f408a853359ccbeff67054/word2vec_pytorch_projector

word2vec_pytorch:

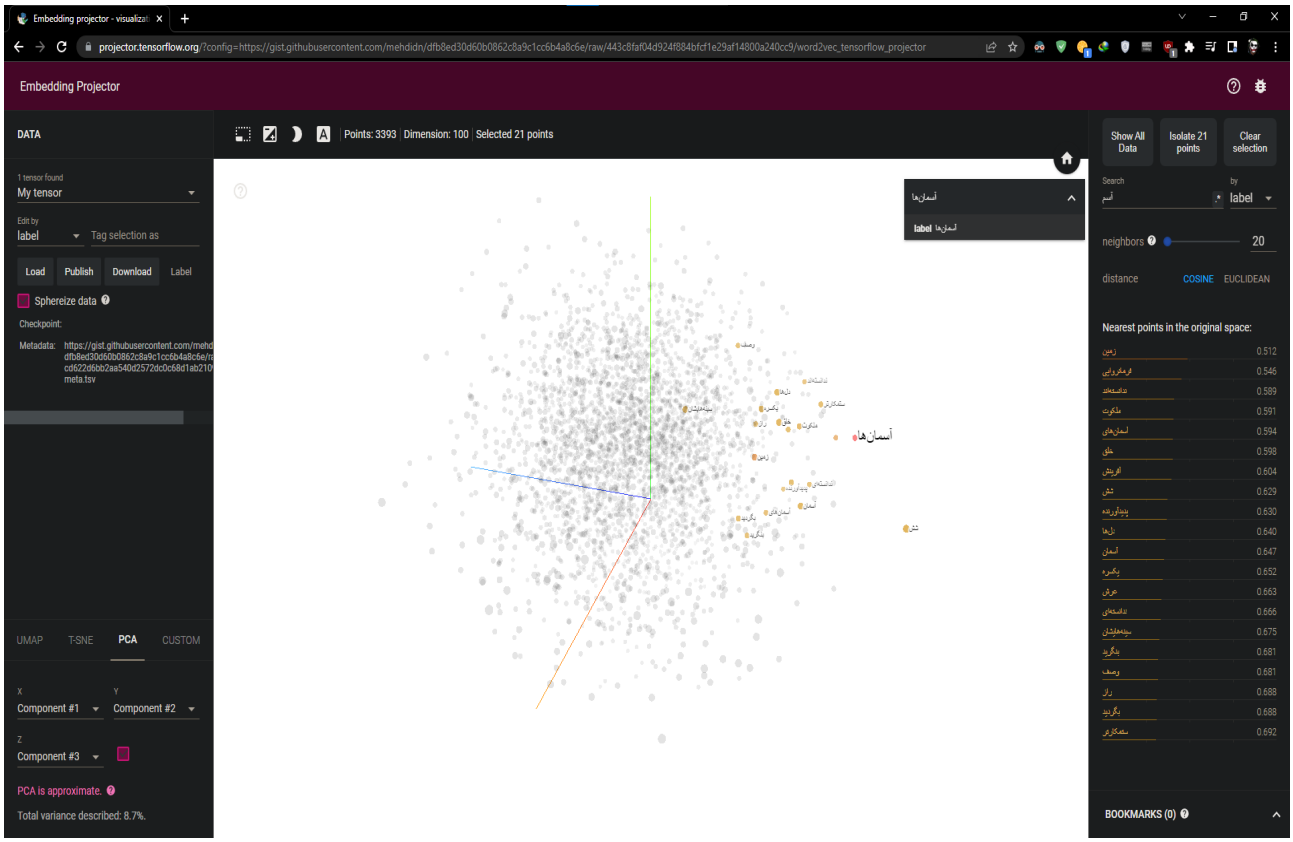
negatives=5 window_size=5 batch_size=256 emb_dimension=100 min_count=5 initial_lr=0.025 iteration=100

https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/mehdidn/8bdf6eb3d228bd521cd8b7e1bd2f8729/raw/bd8d491a24abdc0a55a016f2adaeb29473d2b8ea/word2vec_pytorch_projector_min_count_5

مثال برای کلمه 'عذاب' در لینک اول:



مثال برای کلمه 'آسمان‌ها' در لینک اول:



وب اپ

ساخت گراف

فایل words-vectors بدست آمده از قسمت word۲vec را در پوشه graph-data موجود در پوشه word۲vec_graph قرار می‌دهیم سپس با تنظیم پارامترهای موجود در اسکریپت words-vectors_to_edges_txt.py نسبت به موارد مورد نظر مشخص می‌کنیم که برای هر کلمه که به عنوان یک گره در گراف شناخته می‌شود، نزدیکترین کلمات را تا چه حدی در نظر بگیریم تا بین این کلمات، یال در نظر بگیریم.

به عنوان مثال اگر از threshold صرف نظر کنیم و num_nearest_neighbors را ۱۵ مقداردهی کنیم آنگاه برای هر کلمه موجود در واژگان، بین آن کلمه و ۱۵ نزدیک‌ترین همسایه آن یال در نظر گرفته می‌شود، یعنی از کلمه فعلی به ۱۵ نزدیک‌ترین همسایه آن out-degree متصل می‌شود یا به عبارت دیگر هر کلمه که به عنوان گره در گراف است، ۱۵ تا out-degree دارد که نشان دهنده ۱۵ نزدیک‌ترین همسایه آن کلمه است و به هر یک از این ۱۵ کلمه، ۱ عدد in-degree اضافه خواهد شد که همین کلمه یا همان گره فعلی است.

همچنین اگر به عنوان مثال از num_nearest_neighbors صرف نظر کنیم و threshold را ۰.۹ مقداردهی کنیم، برای هر کلمه، تنها به کلمه‌ای یال خواهیم داشت که شباهت آن به کلمه یا همان گره فعلی بیشتر از ۰.۹ باشد و آن‌هایی که شباهتشان کمتر از این مقدار باشد یا به عبارتی فاصله آن‌ها بیشتر از مقدار در نظر گرفته شده باشد را نادیده می‌گیرد.

بنابراین پس از اجرای اسکریپت words-vectors_to_edges_txt.py، در پوشه graph-data فایلی با نام edges.txt به وجود خواهد آمد که نشان دهنده یال‌های گراف است. به این صورت که یک گره در سمت چپ نوشته شده و سپس نزدیکترین همسایه‌های آن در سمت راست نوشته شده‌اند و به این معناست که بین گره سمت چپ و هر گره سمت راست، که هر کدام کلمه هستند، یال وجود دارد. بنابراین گره‌ها یا همان کلمات سمت راست برای گره یا همان کلمه سمت چپ، out-degree محسوب می‌شوند و همچنین گره سمت چپ برای هر یک از گره‌های سمت راست، یک in-degree محسوب می‌شود.

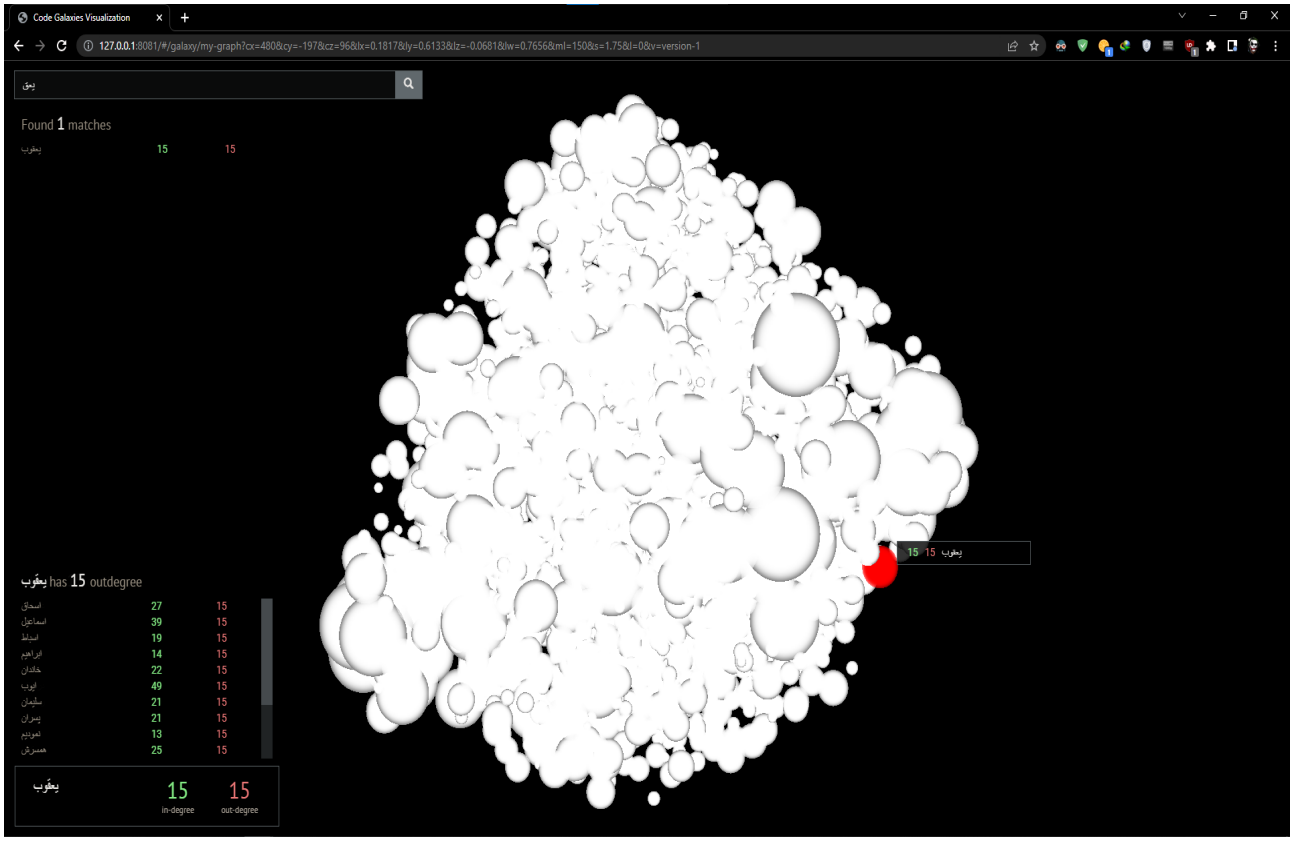
نحوه اجرای وب اپ

دستورالعمل کامل اجرای وب اپ در فایلی با نام how to run web app.txt آمده است. بدیهی است که ابتدا باید پیش‌نیازهایی مثل node.js، کتابخانه annoy و موارد دیگر نصب شده باشند.

لینک فایل how to run web app.txt:

https://github.com/mehdidn/persian_word2vec/blob/main/how%20to%20run%20web%20app.txt

مثال برای کلمه 'یعقوب' در وب اپ:



مثال برای کلمه 'عذاب' در وب اپ: