تمرین سری دوم کامیار مرادیان زه آب ۹۹۵۲۲۱۰۴

سوال اول

بخش الف) تفاوت بین one-hot encoding و word embedding ها را بیان کنید.

پاسخ: هر دو روش متدهایی هستند که برای نمایش کلمات یا داده‌های متنی به شکل عددی استفاده می‌شوند، اما در رویکرد و قابلیت‌هایشان تفاوت دارند.

- روش One-Hot Encoding: یک روش ساده است که در آن هر کلمه با یک بردار باینری، معمولاً با اندازه‌ای برابر با اندازهٔ واژگان، یعنی اندازهٔ Vocabulary که همان $|V|$ است، نمایش داده می‌شود. بردار هر کلمه به طریقی مقداردهی می‌شود که مقدار متناظر با تمام کلمات صفر باشد، مگر مقدار متناظر با کلمهٔ مورد نظر. این رویکرد با هر کلمه به عنوان یک موجودیت مستقل برخورد می‌کند و هیچ رابطه معنایی بین کلمات را در بر نمی‌گیرد.
- روش Word Embedding: این روش مبتنی بر دریافت ارتباط معنایی و رابطهٔ توزیع کلمات با یکدیگر است. در Word Embedding، کلمات به عنوان بردارهای متراکم و با تعداد کمی از ابعاد در فضای چندبعدی نمایش داده می‌شوند به طوری که تعداد این ابعاد چیزی در حدود ۵۰ تا ۱۰۰۰ مورد است. در این روش در صورت شباهت دو کلمه، بردارهای آن‌ها نیز به یکدیگر شباهت خواهد داشت و می‌توان این شباهت را اندازه‌گیری نیز کرد. این تعبیه‌ها معمولاً از مقادیر زیادی داده متنی با استفاده از تکنیک‌های یادگیری عمیق، مانند شبکه‌های عصبی، یاد می‌گیرند.

در زیر مقایسه در چند معیار آورده شده است:

- ابعاد بردار متناظر با هر کلمه: همانطور که گفته شد در روش One-Hot Encoding ابعاد هر کلمه برابر است با اندازهٔ تمام واژگان موجود در Vocabulary. در حالی که در روش Word Embedding هر بردار فشرده‌تر و کوچکتر بوده و در نتیجه فضای کمتری را اشغال می‌کند.
- پراکندگی و فشردگی بردارها: باز هم به دلیل گفته شده، در روش One-Hot Encoding در هر بردار از میان $|V|$ مقدار تنها یک مقدار آن یک بوده و باقی صفر هستند و در نتیجه پراکندگی داده به شدت بالا است. این در حالی است که در روش Word Embedding با توجه به فشرده بودن بردارها تعداد مقادیر صفر کمتر می‌شود.
- ارتباط معنایی: در صورتی که بخواهیم دو واژه را در روش One-Hot Encoding با هم مقایسه کنیم و این کار را با فاصلهٔ اقلیدسی انجام دهیم، فاصلهٔ هر دو کلمهٔ متفاوت، برابر با مقدار ثابت $\sqrt{2}$ خواهد شد و در نتیجه هیچ اطلاعاتی در رابطه با ارتباط معنایی دو واژه بدست نخواهد آمد. این در حالی است که در روش Word Embedding سعی بر این است که بردار منتسب به هر دو واژهٔ مرتبط به هم را تا حد امکان نزدیک کرد تا بتوان واژگان شبیه به هم و یا متضاد را یافت.
- آموزش و بدست آوردن بردارها: در روش One-Hot Encoding نیازی به آموزش وجود نخواهد داشت و کافیست که یک بردار به طول تمام واژگان در اختیار داشت و برای هر کلمه آنچه که در بالا گفته شد را انجام داد. این در حالی است که برای بردارهای موجود در Word Embedding نیاز به آموزش بردارها داریم. آموزشی که نیازمند وجود یک شبکهٔ عصبی است.
- اطلاعات متنی: در روش One-Hot Encoding هر بردار مستقل از متن و صرفاً با توجه به مجموعه واژگان مقداردهی می‌شود. این در حالی است که در روش Word Embedding هر واژه با توجه به ارتباط متنی و واژگان نزدیک به

خود در متن مورد یادگیری قرار می‌گیرد و در نتیجه اطلاعات متنی و نحوهٔ توزیع واژگان و ارتباطات میان آن‌ها در نظر گرفته خواهد شد.

- تعمیم‌دهی: روش Word Embedding قدرت تعمیم‌دهی بالاتری داشته و می‌تواند در صورت مشاهدهٔ داده‌هایی خارج از واژگان بر اساس اطلاعات مرتبط با توزیع واژگان که آموخته است، به خوبی عمل کند.

بخش ب) به طور خلاصه توضیح دهید که الگوریتم GloVe چگونه word embedding ها را تولید می‌کند.

پاسخ: الگوریتم GloVe (Global Vectors for Word Representation) روشی برای بدست آوردن نمایش برداری برای کلمات است. این روش طراحی شده است تا نه تنها از وقوع کلمات در یک مجموعهٔ داده، بلکه نحوهٔ هم‌زبانی کلمات با دیگر واژگان را نیز ثبت کند، که به درک ارتباط معنایی و نحوی آنها کمک می‌کند. الگوریتم GloVe را می توان به دو مرحله اصلی تقسیم کرد:

مرحله ۱: ساخت ماتریس Co-Occurrence

اولین گام شامل جمع آوری آمار کلمات Co-Occur از یک مجموعهٔ معین است. این کار با اسکن داده‌های متنی و ایجاد یک ماتریس Co-Occurrence انجام می‌شود. هر المان $X_{i,j}$ از این ماتریس نشان می‌دهد که چند بار واژهٔ i در نزدیکی واژهٔ j و در داخل پنجرهٔ تعریف شده، دیده شده است. اندازهٔ پنجره تعیین می‌کند که چه تعداد کلمه قبل و بعد از کلمهٔ مورد نظر در نظر گرفته می‌شود. علاوه بر این، به کلماتی که دورتر از کلمه هدف هستند، وزن کمتری داده می‌شود که معمولاً با استفاده از یک تابع فروپاشی مانند $\text{decay} = 1/\text{offset}$ انجام می‌شود.

مرحلهٔ ۲: آموزش بردار واژگان

مرحله دوم شامل یادگیری بردارهای کلمه با تعریف تابع هزینه و بهینه‌سازی آن است. تابع هزینه به گونه ای طراحی شده است که بردارهای کلمه را تشویق کند تا یک ضرب نقطه ای داشته باشند که برابر با لگاریتم احتمال وقوع همزمان آنها باشد. به طور خاص، برای هر جفت از کلمات، مدل، محدودیت‌های نرم را به صورت $w_i^T w_j + b_i + b_j = \log(X_{ij})$ تعریف می‌کند، به طوری که در آن w_i و w_j بردارهای مربوط به واژگان اصلی و Context هستند. همچنین b_i و b_j بایاس‌های این واژگان می‌باشند. در نتیجه تابع ضرر به صورت زیر تعریف می‌شود:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(w_i^T w_j + b_i + b_j - \log X_{ij}\right)^2$$

که در این رابطه f یک تابع وزن‌دهی است که به جلوگیری از تسلط یادگیری توسط جفت کلمات بسیار مکرر کمک می‌کند. نویسندگان GloVe تابع خاصی را برای f انتخاب می‌کنند که شامل یک پارامتر قطع X_{max} و یک توان a است.

مطالب مربوط به این بخش با کمک ChatGPT جمع‌آوری شده است:

The GloVe (Global Vectors for Word Representation) algorithm is a method for obtaining vector representations for words. This method is designed to capture not just the occurrence of words in a corpus but also how words co-occur with others, which helps in understanding their semantic and syntactic meanings. The GloVe algorithm can be broken down into two main steps:

Step 1: Construct a Co-occurrence Matrix

The first step involves collecting word co-occurrence statistics from a given corpus. This is done by scanning the text data and creating a co-occurrence matrix X. Each element X_{ij} of this matrix represents how often word i appears in the context of word j, within a defined window size around each word. The window size determines how many words before and after the target word will be considered its context. Additionally, words that are farther away from the target word are given less weight, usually calculated using a decay function such as $decay = 1/offset$ [2].

Step 2: Learn Word Vectors

The second step involves learning word vectors by defining a cost function and optimizing it. The cost function is designed to encourage the word vectors to have a dot product that equals the logarithm of their co-occurrence probability. Specifically, for each word pair, the model defines soft constraints $w_i^T w_j + b_i + b_j = log(X_{ij})$, where w_i and w_j are the word vectors for the main and context words, respectively, and b_i, b_j are scalar biases for the main and context words. The cost function to be minimized is:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

Here, f is a weighting function that helps to prevent learning from being dominated by very frequent word pairs. The GloVe authors choose a specific function for f that involves a cutoff parameter X_{max} and an exponent α [2].

بخش پ) به طور خلاصه توضیح دهید که الگوریتم Word2Vec چگونه word embedding ها را تولید می‌کند.

پاسخ: این الگوریتم از دو روش CBOW و Skip-Grams استفاده می‌کند که در اینجا تنها Skip-Grams شرح داده می‌شود. در این الگوریتم، به عنوان ورودی یک مجموعه داده از متون مختلف و اندازهٔ مجموعهٔ واژگان 'N' را می‌گیرد. با اختصاص یک بردار Embedding تصادفی برای هر یک از N واژهٔ موجود در مجموعه الگوریتم آغاز شده و سپس مرحله به مرحله بردار Embedding هر کلمه را تغییر می‌دهد تا اینکه بیشتر شبیه بردار واژگانی باشد که در نزدیکی کلمه در متن قرار گرفته‌اند و کمتر شبیه جاسازی کلماتی باشد که در این نزدیکی وجود ندارد. بنابراین این روش یک روش Binary Classification بوده به جای آنکه یک روش پیشبینی واژگان باشد. پس نیاز است که یک شبکهٔ عصبی را برای این منظور آموزش داد تا بتوان پیش‌بینی لازم را انجام داد. این شبکهٔ عصبی تنها از یک لایه تشکیل شده و شامل یک تابع فعال‌ساز Sigmoid است.

همانطور که گفته شد نیاز است که یک کلمه در هر گام انتخاب شود که به عنوان Target Word شناخته می‌شود و از طرفی برای این کلمه یک مجموعه از کلمات اطراف آن را از داخل متن برداشت. برای انتخاب واژگان اطراف هر کلمه از یک پنجره به سایز L استفاده می‌شود. به واژهٔ بدست آمده، Context Word گفته می‌شود. سپس بر اساس الگوریتم Negative Sampling، برای هر واژه، به تعداد k لغت را به صورت تصادفی و از میان واژگان داخل مجموعهٔ واژگان Nتاییمان انتخاب می‌کنیم. این کار را بر اساس میزان تکرار لغات در داخل داده‌های متنی (Corpus) انجام می‌دهیم. به عبارتی رابطهٔ مورد استفاده برای انتخاب واژگان مطابق زیر بوده که برای اجتناب از لغات فراوان در مجموعه لغات است.

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

بنابراین حال برای یک Target Word به تعداد 2L دسته برای آموزش در اختیار داریم. به طوری که در هر دسته یک Context Word داشته که الگوریتم بایستی به ازای آن پیشامد مثبت پیشبینی کند، و همچنین k واژهٔ نویزی که الگوریتم می‌بایست به ازای هر کدام از آن‌ها پیشامد منفی را پیشبینی کند. بنابراین تابع هدف جهت آموزش را می‌توان به طریقی تعیین کرد که:

- بیشینه کردن شباهت میان Target و واژگان Context
- کمینه کردن شباهت میان Target و واژگان نویزی

بر این اساس، و پس از استفاده از این الگوریتم، شبکهٔ عصبی در طول فرآیند بهینه‌سازی خود، سعی می‌کند بردارها را جابجا کند تا بردار واژهٔ Target به بردار واژگان Context نزدیک‌تر شده و از برداهای واژگان Noisy فاصله بگیرد. همچنین این نکته نیز قابل ذکر است که در این الگوریتم رایج است که دو بردار برای هر واژه آموزش داده شود. یک بردار برای هنگامی که به عنوان Target در طول آموزش انتخاب شده و یکی هم برای زمانی که به عنوان Context بوده است.

توضیحات مربوط به این بخش از کتاب مرجع گرفته شده است.

بخش ت) واژه‌های دارای چند معنی چگونه در word embedding ها کنترل می‌شوند و چه چالشهایی را تولید می‌کنند؟

پاسخ: چالش‌های تولید شده توسط این واژگان:

به این نکته که ممکن است واژگان چند معنی را تولید کنند، Polysemous گفته می‌شود که باعث ایجاد دشواری در درک یک متن می‌شود. به عنوان مثال ممکن است که یک نفر در هنگام سرچ کردن این عبارت را جست‌وجو کند: اطلاعات در ارتباط با شیر. در این جا واژهٔ «شیر» بدلیل داشتن چند معنی باعث ایجاد سردرگمی می‌شود و در نتیجه Embeddingای که برای آن واژه بدست آورده‌ایم به طور دقیق معنای مورد نظر کلمه را نشان ندهد.

نحوهٔ مقابله با این چالش‌ها:

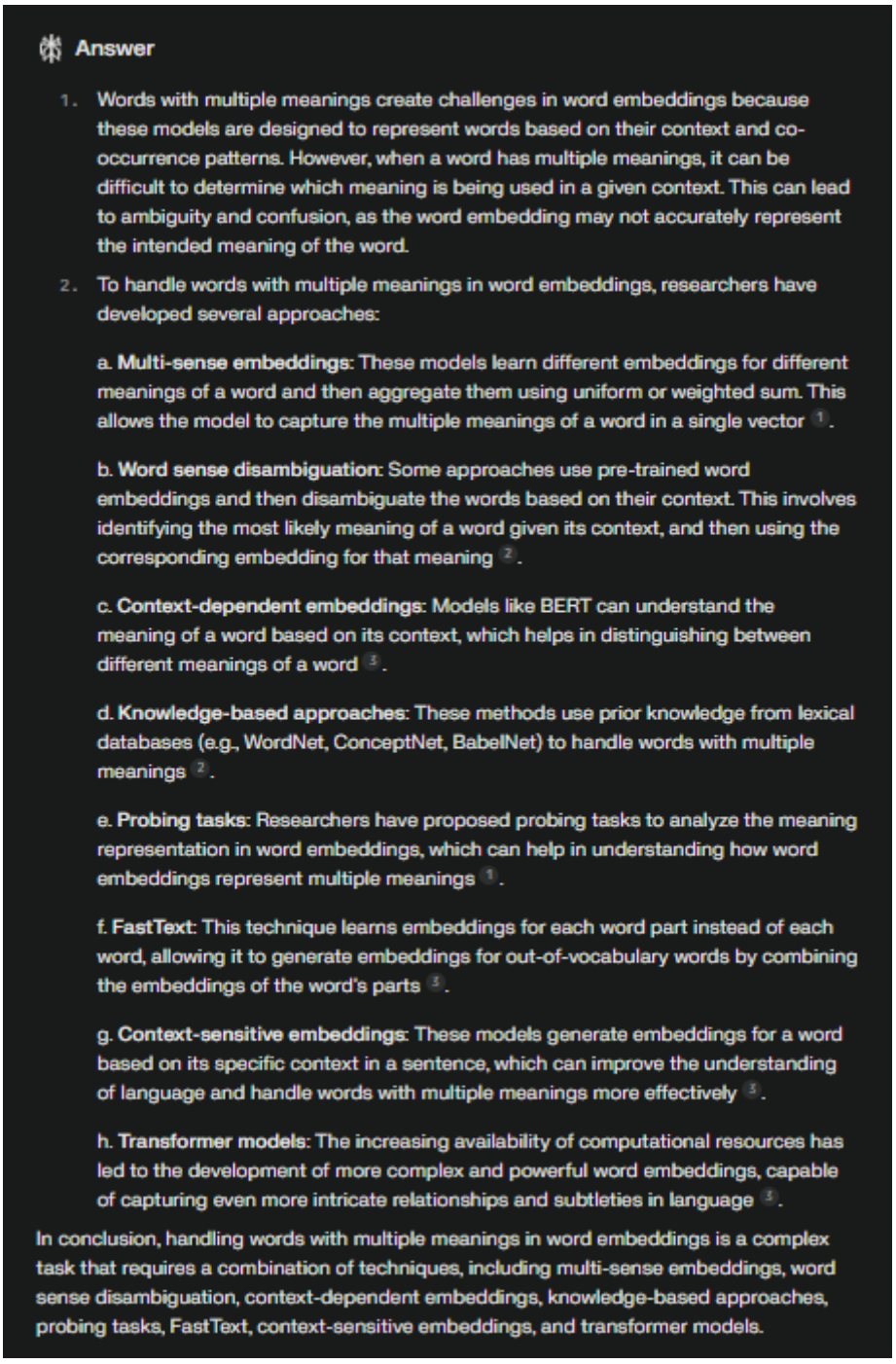
برای مدیریت کلمات با معانی متعدد در Word Embeddingها، محققان چندین رویکرد را توسعه داده‌اند:

- Multi-sense Embeddings: این مدل‌ها Embeddingهای مختلف را برای معانی مختلف یک کلمه یاد می‌گیرند و سپس با استفاده از مجموع یکنواخت یا وزنی آن‌ها را جمع می‌کنند. این به مدل اجازه می‌دهد تا معانی متعدد یک کلمه را در یک بردار واحد ثبت کند.
- Word Sense Disambiguation: برخی از رویکردها از Word Embeddingهای از پیش آموزش‌دیده استفاده می‌کنند و سپس کلمات را بر اساس متن تحت بررسی، رفع ابهام می‌کنند. این شامل شناسایی محتمل‌ترین معنای یک کلمه با توجه به بافت آن، و سپس استفاده از Embedding مربوط به آن معنا است.
- Contextual Embedding: این روش، یک روش پویا جهت ایجاد Embeddingهای واژگان است که در آن به جای آنکه مانند روش Word Embedding Word2Vecای یکتا برای هر واژه ایجاد کنند که صرف نظر از متنی که واژه در آن قرار گرفته است، آموزش داده شده است، از چندین Embedding برای هر واژه ایجاد می‌کنند که با

توجه به متنی که در آن قرار گرفته است، متفاوت خواهد بود. از جمله مدل‌های نمایشی که به این صورت تعریف شده‌اند، نمایش BERT است.

- Knowledge-based approaches: در این روش‌ها از دانش قبلی بدست آمده از پایگاه‌های واژگانی (مانند WordNet، ConceptNet، BabelNet) برای مدیریت کلمات با معانی متعدد استفاده می‌شوند.

برای این بخش از ChatGPT کمک گرفته شده است:



Answer

1. Words with multiple meanings create challenges in word embeddings because these models are designed to represent words based on their context and co-occurrence patterns. However, when a word has multiple meanings, it can be difficult to determine which meaning is being used in a given context. This can lead to ambiguity and confusion, as the word embedding may not accurately represent the intended meaning of the word.
2. To handle words with multiple meanings in word embeddings, researchers have developed several approaches:

   a. **Multi-sense embeddings**: These models learn different embeddings for different meanings of a word and then aggregate them using uniform or weighted sum. This allows the model to capture the multiple meanings of a word in a single vector [1].

   b. **Word sense disambiguation**: Some approaches use pre-trained word embeddings and then disambiguate the words based on their context. This involves identifying the most likely meaning of a word given its context, and then using the corresponding embedding for that meaning [2].

   c. **Context-dependent embeddings**: Models like BERT can understand the meaning of a word based on its context, which helps in distinguishing between different meanings of a word [3].

   d. **Knowledge-based approaches**: These methods use prior knowledge from lexical databases (e.g., WordNet, ConceptNet, BabelNet) to handle words with multiple meanings [2].

   e. **Probing tasks**: Researchers have proposed probing tasks to analyze the meaning representation in word embeddings, which can help in understanding how word embeddings represent multiple meanings [1].

   f. **FastText**: This technique learns embeddings for each word part instead of each word, allowing it to generate embeddings for out-of-vocabulary words by combining the embeddings of the word's parts [3].

   g. **Context-sensitive embeddings**: These models generate embeddings for a word based on its specific context in a sentence, which can improve the understanding of language and handle words with multiple meanings more effectively [3].

   h. **Transformer models**: The increasing availability of computational resources has led to the development of more complex and powerful word embeddings, capable of capturing even more intricate relationships and subtleties in language [3].

In conclusion, handling words with multiple meanings in word embeddings is a complex task that requires a combination of techniques, including multi-sense embeddings, word sense disambiguation, context-dependent embeddings, knowledge-based approaches, probing tasks, FastText, context-sensitive embeddings, and transformer models.

بخش ث) word embeddingها نیاز دارند که تمام کلمات در مجموعه آموزش حضور داشته باشند. چگونه با کلمات خارج از واژگان out of vocabulary برخورد می‌کنید؟ یک روش برای تولید word embeddingبرای کلماتی که در داده آموزش حاضر نبوده اند پیشنهاد دهید.

پاسخ: همانطور که گفته شد، مشکل اساسی موجود در word embeddingها آن است که در مقابله با کلمات ناشناخته راه خوبی ندارد. این مشکل به خصوص در الگوریتم Word2Vec نیز مشاهده می‌شود. در این الگوریتم علاوه بر مشکل ذکر شده، مشکل دیگری نیز در زمینهٔ پراکندگی لغات وجود دارد، که در زبان‌هایی با صرف‌شناسی غنی (مانند زبان عربی)، که در آن‌ها برخی از اشکال متعدد برای هر اسم و فعل ممکن است که به ندرت رخ دهد.

برای حل این مشکلات، یک توسعه بر Word2Vec تحت عنوان FastText ایجاد شده است. در این روش با استفاده از مدل‌های زیرکلمه‌ای (Sub-Word) که هر کلمه تبدیل به دو مجموعهٔ کلی تبدیل کرد: یکی خود کلمه و دیگری مجموعه‌ای از n-gramهای تشکیل‌دهندهٔ آن واژه که با نمادهای مرزی ویژه < و > که به هر کلمه اضافه می‌شود.

به عنوان مثال، با n=3 کلمهٔ Where را با دنبالهٔ <Where> نمایش می‌دهیم. سپس این دنباله را به صورت مجموعهٔ n-gram تعریف شده در بالا نمایش می‌دهیم: <wh, whe, her, ere, re>

به این ترتیب برای هر n-gram تشکیل‌دهندهٔ واژهٔ Where، می‌توان Embeddingای را با استفاده از الگوریتم Skip-Gram آموزش داد و داخل دیتابیس قرار داد. سپس در اثر مواجهه با هر واژهٔ جدید در هنگام تست، به طریق گفته شده n-gramهای تشکیل‌دهندهٔ آن واژه را ایجاد می‌کنیم و با استفاده از n-gramهای از پیش بدست آمده، Embedding مربوط به آن را با جمع کردن این Embeddingها بدست می‌آوریم.

را خودش به‌علاوه یک کیسه n-گرم تشکیل‌دهنده نشان می‌دهد، با نمادهای مرزی ویژه < و > که به هر کلمه اضافه می‌شود، با این مشکلات برخورد می‌کند.

تمرین سری دوم کامیار مرادیان زه آب ۹۹۵۲۲۱۰۴

سوال دوم

در ابتدا نیاز است که مجموعهٔ واژگان یکتا را که در متن داده شده حضور دارند، مشخص کنیم. در این مثال فرض می‌کنیم که نقطهٔ نهایی قرار گرفته در متن جز مجموعه واژگان قرار نمی‌گیرد. به عبارتی هر علامت نگارشی را در ابتدا از جمله حذف می‌کنیم و سپس واژگان آن را از داخل آن تشخیص داده و جدا می‌کنیم.

Corpus: I love computer science and I love NLP even more.

Unique Words (Vocabulary): [I, love, computer, science, and, NLP, even, more]

حال با توجه به اینکه سایز پنجره برابر با ۲ است، برای هر لغت ۲ واژه بعد و ۲ واژه قبل از آن را در نظر می‌گیریم و ماتریس Co-Occurrence را تشکیل می‌دهیم:

	I	love	computer	science	and	NLP	even	more
I	0	2	1	1	1	1	0	0
love	2	0	1	1	1	1	1	0
computer	1	0	0	1	1	0	0	0
science	1	1	1	0	1	0	0	0
and	1	1	1	1	0	0	0	0
NLP	1	1	0	0	0	0	1	1
even	0	1	0	0	0	1	0	1
more	0	0	0	0	0	1	1	0

قطر اصلی صفر شده است چرا که در Co-Occurrence بدنبال محاسبهٔ تعداد مشاهدهٔ دو کلمهٔ متفاوت باهمدیگر هستیم. برای توضیح بیشتر ۳ سلول رنگی شده در بالا در زیر بررسی می‌کنیم:

- رنگ آبی: در این سلول، مقداری که قرار گرفته است مربوط به کلمهٔ I بوده که کلمهٔ Love در پنجرهٔ دربردارندهٔ آن قرار گرفته است. این مقدار برابر با دو شده است که در زیر این دو مورد آورده شده‌اند:

I	**love**	computer	science	and	NLP	I	love	even	more
I	love	computer	science	and	NLP	I	**love**	even	more

- رنگ زرد: برای محاسبهٔ مقدار مربوط به این سلول نیاز است که Target را لغت Love و Context را لغت Even قرار دهیم که به نتیجهٔ زیر می‌رسیم:

I	love	computer	science	and	NLP	I	love	even	more
I	love	computer	science	and	NLP	I	love	**even**	more

همانطور که مشخص است کلمهٔ even تنها یک دفعه در پنجرهٔ کلمهٔ love قرار گرفته است.

- رنگ سبز: به ازای Target=even و context=and مقدار صفر بدست آمده است که دلیلش در زیر است:

I	love	computer	science	and	NLP	I	love	even	more

همانطور که مشاهده می‌شود در پنجرهٔ مربوط به لغت هدف، هیچ نشانی از کلمهٔ and وجود ندارد.