

Session 03&04

Feature Extraction for Text

NLP | Machine Learning | Zahra Amini

Telegram: @zahraamini_ai & Instagram:@zahraamini_ai & LinkedIn: @zahraamini-ai

<https://zil.ink/zahraamini>

فرآیندی است که طی آن، متن به واحدهای کوچکتری به نام "تو肯" تقسیم می‌شود. تو肯‌ها می‌توانند شامل کلمات، جملات یا حتی کاراکترها باشند. در واقع، Tokenization به ما امکان می‌دهد تا یک متن طولانی را به واحدهای کوچکتر بشکنیم تا بتوانیم بهتر روی آن تحلیل انجام دهیم

Tokenization چیست؟

تفاوت بین Tokenization کلمات و جملات

کلمات: در این حالت، هر کلمه به عنوان یک تو肯 در نظر گرفته می‌شود. برای مثال: [من، عاشق، هوش، مصنوعی، هستم] → من عاشق هوش مصنوعی هستم این نوع توکنیزه کردن برای مدل‌هایی که نیاز به درک تک‌تک کلمات، تحلیل معنای کلمات یا بررسی ساختار گرامری دارند بسیار مفید است

جملات: در این روش، متن به جملات جداگانه شکسته می‌شود. برای مثال: ["دیروز به سینما رفتم"، "فیلم فوق العاده بود"] → دیروز به سینما رفتم. فیلم فوق العاده بود. این نوع توکنیزه کردن معمولاً برای پردازش‌هایی که سطح جمله را هدف قرار می‌دهند، کاربرد دارد؛ مانند زمانی که نیاز به تحلیل احساسات در جملات یا شناسایی موضوعات مختلف در یک متن طولانی داریم

تحلیل سطح جمله یعنی چه؟ پردازش و بررسی ویژگی‌های یک جمله به صورت جداگانه، به جای اینکه کل متن را یکجا بررسی کنیم. این کار به ما کمک می‌کند که موضوع یا احساس هر جمله را بفهمیم و بخش‌های مهم آن را شناسایی کنیم

✓ در کتابخانه NLTK، می‌توان از تابع `word_tokenize` استفاده کرد که به صورت خودکار کلمات را از جمله جدا می‌کند و حتی علائم نگارشی را به عنوان تو肯 مجزا در نظر می‌گیرد

نقشی حیاتی در آماده‌سازی داده‌ها برای مدل‌های یادگیری ماشین ایفا می‌کند. با درک تفاوت‌ها و اهمیت توکنیزه کردن، شما می‌توانید داده‌های متنی را به شکلی آماده کنید که مدل‌ها بهتر بتوانند از آن استفاده کنند

مفهوم ویژگی‌ها Features

ویژگی‌ها اجزای اطلاعاتی کلیدی هستند که از داده‌های خام استخراج می‌شوند و به ما کمک می‌کنند تا داده‌ها را بهتر تحلیل و مدل‌سازی کنیم. در پردازش زبان طبیعی، ویژگی‌ها شامل اطلاعات مهمی هستند که از متون استخراج می‌شوند و بازنمایی عددی متن را ممکن می‌سازند تا برای الگوریتم‌های یادگیری ماشین قابل درک باشند.

استخراج ویژگی‌ها یکی از مهمترین مراحل در هر پروژه NLP است، زیرا الگوریتم‌های یادگیری ماشین قادر به پردازش متن به شکل خام نیستند. آن‌ها به داده‌های عددی نیاز دارند، و به همین دلیل ما باید اطلاعات مفیدی از متن استخراج کنیم که بتواند نمایانگر مفاهیم اصلی و ساختار جمله باشد. ویژگی‌ها می‌توانند شامل فراوانی کلمات، اطلاعات نحوی (مثل نوع کلمه)، توالی کلمات، و غیره باشند.

اهمیت استخراج ویژگی از داده‌های متنی

چرا استخراج ویژگی‌ها از متن ضروری است؟

❖ **ارتباطات معنایی:** با استخراج ویژگی‌هایی که معنای کلمات یا عبارات را نمایان می‌کنند، می‌توانیم به مدل کمک کنیم تا بهتر ارتباط بین کلمات را درک کند برای مثال، کلماتی مانند "خوشحال" و "شاد" در یک فضای معنایی مشابه قرار دارند که برای تشخیص احساسات متنی مهم است

❖ **بهبود کارایی الگوریتم‌ها:** مدل‌های یادگیری ماشین با استفاده از ویژگی‌های مهم و مرتبط بهتر آموزش می‌بینند، که منجر به کارایی بالاتر در وظایف مختلف پردازش زبان طبیعی می‌شود

❖ **کاهش نویز:** با استخراج ویژگی‌های مناسب، می‌توانیم بخش‌های غیرضروری متن را حذف کرده و اطلاعات کاربردی‌تر را به مدل ارائه دهیم

مثلاً، کلمات پرکاربرد و کم‌ارزشی مانند "و"، "به"، و "در" که به تنها بی معنای خاصی ندارند، حذف می‌شوند. در نتیجه، ویژگی‌ها کمک می‌کنند تا مدل بتواند به جای توجه به کلمات غیرضروری، بر کلمات کلیدی تمرکز کند

Word-level Features

ویژگی‌های مبنی بر واژه

این روش شامل شمارش تعداد دفعات حضور هر کلمه در متن است. در BoW، ترتیب کلمات اهمیت ندارد و صرفاً به حضور یا غیاب کلمات توجه می‌شود. این روش به ما اجازه می‌دهد که به سادگی متون مختلف را به یک فرمت عددی تبدیل کنیم

در این روش، متن به یک بردار عددی تبدیل می‌شود که هر عنصر آن نشان‌دهنده حضور یا تعداد تکرار یک کلمه خاص است و ترتیب کلمات اهمیتی ندارد. این مدل به‌ویژه در پردازش زبان طبیعی و در کارهایی مانند طبقه‌بندی متن یا تحلیل احساسات بسیار مفید است

مراحل BoW

① پیش‌پردازش متن :

- △ حذف علائم نگارشی: حذف علائم نگارشی مانند نقطه، ویرگول و سایر نشانه‌ها
- △ تبدیل به حروف کوچک: به دلیل اینکه حروف بزرگ و کوچک ممکن است یک کلمه را دو بار محاسبه کنند، تمامی حروف را کوچک می‌کنیم
- △ توکن‌سازی: تقسیم متن به کلمات مجزا

② ساخت واژگان :

- △ فهرستی از تمام کلمات منحصر به فرد در کل مجموعه متون تهیه می‌شود. این فهرست به عنوان "واژگان" شناخته می‌شود

③ ایجاد ماتریس ویژگی :

- △ یک ماتریس X ایجاد می‌شود که در آن هر سطر نشان‌دهنده یک سند و هر ستون نشان‌دهنده یک کلمه منحصر به فرد در واژگان است
- △ هر عنصر x_{ij} در این ماتریس نشان‌دهنده تعداد دفعات تکرار کلمه j در سند i است

مثال BoW

فرض کنید دو سند زیر را داریم

سند ۱: "گربه دوست دارد بازی کند"

سند ۲: "گربه و سگ دوست دارند بخوابند"

ابتدا، واژگان مشترک بین این دو سند را ایجاد می‌کنیم:

واژگان = ["گربه", "دوست", "دارد", "بازی", "کند", "و", "سگ", "دارند", "بخوابند"]

واژگان = ["گربه", "دوست", "دارد", "بازی", "کند", "و", "سگ", "دارند", "بخوابند"]

سپس، برای هر سند، تعداد دفعات تکرار هر واژه را در یک ماتریس نمایش می‌دهیم

سند/واژه	گربه	بخوابند	دارند	و	کند	بازی	دارد	دوست	سگ	دارد	بازی	کند	و	سگ	دارند	بخوابند	دارد	بازی	کند	و	سگ	دارند	بخوابند		
سند ۱	1	0	1	1	0	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	
سند ۲	1	1	0	1	1	0	0	0	1	1	0	0	1	1	0	1	0	0	1	1	1	1	0	1	1

اگر X را ماتریس ویژگی با اندازه $n \times m$ در نظر بگیریم که m تعداد اسناد و n تعداد واژگان است، هر ورودی i, j تعداد دفعات تکرار واژه j در سند i است. این ماتریس معمولاً به صورت اسمی Sparse Matrix نمایش داده می‌شود، چرا که در اکثر مواقع تعداد زیادی از واژگان در هر سند خاص حضور ندارند و بسیاری از ورودی‌های ماتریس صفر خواهند بود

هر مقدار در جدول بالا تعداد دفعات تکرار هر واژه در سند مربوطه را نشان می‌دهد. به این ترتیب، برای هر سند یک بردار ویژگی ایجاد کرده‌ایم که نشان‌دهنده تعداد تکرار هر واژه است

مزایا

 ساده و موثر برای پردازش‌های اولیه
 سریع و آسان برای اجرا در مقیاس‌های بزرگ

معایب

ترتیب کلمات را نادیده می‌گیرد، بنابراین ارتباط میان کلمات در جمله‌ها از دست می‌رود
 معنی کلمات و ارتباطات معنایی بین آن‌ها نادیده گرفته می‌شود
 ماتریس حاصل از این روش معمولاً بسیار پراکنده است، که باعث پیچیدگی در مدل‌های یادگیری ماشین می‌شود

کیسه‌ای از بیگرام‌ها و n -گرام‌ها یکی از تکنیک‌های اصلی استخراج ویژگی در پردازش زبان طبیعی است که به‌طور خاص در مدل‌های یادگیری ماشین کاربرد دارد. در این روش، به‌جای تحلیل کلمات به صورت مستقل، دنباله‌هایی از کلمات در نظر گرفته می‌شود که می‌توانند مفهوم بهتری از جملات ارائه دهند.

◊ Bag of Bigrams & N-grams

$n = 1 \rightarrow$ Unigram

دنباله‌ای از کلمه متوالی در یک جمله یا سند است

$n = 2 \rightarrow$ Bigram

$n = 3 \rightarrow$ Trigram

یک نمایش برداری از سند است که تعداد تکرار هر n -گرام را ذخیره می‌کند. در این روش، ترتیب کلمات مهم نیست و تنها تکرار n -گرام‌ها در نظر گرفته می‌شود.

N-grams

Bag of N-grams

فرض کنید جمله‌ای به صورت زیر داریم:
"I love natural language processing."

می‌توان این جمله را به صورت زیر تجزیه کرد:

تری‌گرام‌ها ($n=3$)
 "I love natural"
 "love natural language"
 "natural language processing"

بیگرام‌ها ($n=2$)
 "I love"
 "love natural"
 "natural language"
 "language processing"

نحوه‌ی ساخت N-grams

تعداد کل
N-grams

فرض کنید جمله‌ای داریم که شامل N کلمه است. تعداد کل n -گرام‌های ممکن در این جمله برابر است با:

$$N - n + 1$$

به عنوان مثال، برای جمله‌ای که ۵ کلمه دارد و با بیگرام‌ها را محاسبه می‌کنیم، تعداد بیگرام‌های ممکن برابر است با:

گرام به صورت یک ویژگی نمایش داده می‌شود. فرض کنید مجموعه‌ای از جملات داریم که پس از تجزیه به- n گرام‌ها، هر- n در مدل کیسه‌ای بیگرام‌ها، مجموعه‌ی زیر از بیگرام‌ها به دست می‌آید

"I love"
"love natural"
"natural language"
"language processing"

سپس می‌توانیم هر جمله را به برداری تبدیل کنیم که تعداد دفعات تکرار هر بیگرام را نشان دهد

:مثال
اگر دو جمله زیر را داشته باشیم:

"I love natural language processing."
"I love machine learning."

مجموعه‌ی بیگرام‌های منحصر به فرد ما به صورت زیر خواهد بود:

"I love"
"love natural"
"natural language"
"language processing"
"love machine"
"machine learning"

بردارهای ویژگی برای این دو جمله:

1: [1, 1, 1, 1, 0, 0], 2: [1, 0, 0, 0, 1, 1]

مزایا: با استفاده از n -گرام‌ها می‌توان اطلاعات بیشتری در مورد توالی کلمات به دست آورد، که برای تحلیل متن‌هایی که ترتیب کلمات در آنها مهم است، بسیار مفید می‌باشد

معایب: افزایش مقدار n باعث افزایش تعداد ویژگی‌ها می‌شود که می‌تواند منجر به پیچیدگی محاسباتی و مشکلات حافظه‌ای شود. همچنین، ممکن است سیاست برآرایش (Overfitting) به مدل بیفزاید

یکی از محبوب‌ترین روش‌های استخراج ویژگی از متن در پردازش زبان طبیعی است. این روش برای ارزیابی اهمیت یک واژه در یک سند و در کل مجموعه اسناد به کار می‌رود. TF-IDF ترکیبی از دو مقدار مهم است: TF (تعداد تکرار واژه در سند) و IDF (معکوس تعداد دفعات تکرار در مجموعه اسناد)

◊ TF-IDF
Term Frequency-Inverse Document Frequency

محاسبه TF (Term Frequency)

$$TF_{ij} = \frac{f_{ij}}{\sum_k f_{ik}}$$

$i_{ij}f$: تعداد دفعات تکرار واژه j در سند i

$\sum_k f_{ik}$: تعداد کل واژه‌های موجود در سند i

فرض کنید سندی به شکل زیر داریم:

سند: "گربه دوست دارد بازی کند، گربه همیشه دوست دارد بازی کند"

تعداد کل واژه‌ها در این سند برابر با ۸ است و واژه "گربه" دو بار تکرار شده است.

$$TF_{گربه} = \frac{2}{11} = 0.18$$

تکمله: اینجا برای شمارش کل کلمات موجود در سند، کلمات تکراری را حذف نمی‌کنیم بنابراین مخرج کسر برابر ۱۱ می‌شود

محاسبه IDF (Inverse Document Frequency)

IDF اهمیت واژه در کل مجموعه اسناد را محاسبه می‌کند. ایده پشت این مقدار این است که اگر یک واژه در بسیاری از اسناد تکرار شود، احتمالاً اهمیت کمتری دارد. فرمول محاسبه IDF به شکل زیر است:

$$IDF_j = \log \left(\frac{N}{1 + n_j} \right)$$

با افزودن ۱ به n_j ، اطمینان حاصل می‌شود که برای واژه‌هایی که در هیچ سندی

N : تعداد کل اسناد.

n_j : تعداد اسنادی که واژه j در آن‌ها حضور دارد.

\log : معمولاً از لگاریتم طبیعی یا لگاریتم پایه ۱۰ استفاده می‌شود.

وجود ندارند، تقسیم بر صفر رخ نمی‌دهد

فرض کنید سه سند به شکل زیر داریم:

"گربه دوست دارد بازی کند"

"گربه و سگ دوست دارند بخوابند"

"گربه بازی می‌کند و استراحت می‌کند"

اگر بخواهیم IDF را محاسبه کنیم، چون این واژه در هر سه سند وجود دارد ($3 = n_j$) ، محاسبه به این صورت خواهد بود:

$$IDF_{\text{گربه}} = \log \left(\frac{3}{1 + 3} \right) = \log(3/4) = -0.125$$

$$TF = \frac{\text{تعداد دفعات حضور کلمه در سند}}{\text{تعداد کل کلمات در سند}}$$

$$IDF = \log \frac{\text{تعداد کل اسناد}}{\text{تعداد اسنادی که شامل کلمه هستند}}$$

$$TF-IDF = TF \times IDF$$

مقدار نهایی TF-IDF از ترکیب مقادیر TF و IDF به دست می‌آید. فرمول محاسبه TF-IDF به شکل زیر است:

$$\text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_j$$

این مقدار به ما کمک می‌کند تا وزن نهایی یک واژه در یک سند را با توجه به مجموعه اسناد تعیین کنیم.

محاسبه TF-IDF

فرض کنید دو سند داریم:

سند ۱: "گربه دوست دارد بازی کند"

سند ۲: "گربه و سگ دوست دارند بخوابند"

برای IDF برای هر واژه

$$0 = \left(\frac{2}{2} \right) \log = \left(\frac{2}{1+1} \right) \log = "گربه"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "دوست"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "دارد"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "بازی"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "کند"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "و"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "سگ"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "دارند"$$

$$0 = \left(\frac{2}{1+1} \right) \log = "بخوابند"$$

برای TF برای هر واژه

برای سند ۱ :

$$= 1/6 \approx 0.167 = "گربه"$$

$$= 1/6 \approx 0.167 = "و"$$

$$= 1/6 \approx 0.167 = "سگ"$$

$$= 1/6 \approx 0.167 = "دوست"$$

$$= 1/6 \approx 0.167 = "دارند"$$

$$= 1/6 \approx 0.167 = "بخوابند"$$

برای سند ۲ :

$$= 1/5 = 0.2 = "گربه"$$

$$= 1/5 = 0.2 = "دوست"$$

$$= 1/5 = 0.2 = "دارد"$$

$$= 1/5 = 0.2 = "بازی"$$

$$= 1/5 = 0.2 = "کند"$$

در مثال بالا، نتایج نشان می‌دهند که چون همه واژه‌ها در هر دو سند مشترک‌اند، مقدار TF-IDF آن‌ها صفر شده است. این یعنی هیچ واژه‌ای ویژگی منحصر به‌فردی برای یکی از اسناد ارائه نمی‌دهد و همه واژه‌ها عمومی و غیرمتمايز هستند.

به طور خلاصه، TF-IDF به ما کمک می‌کند تا واژه‌های پرترکار و مهم در هر سند را پیدا کنیم و از واژه‌های عمومی که در اکثر اسناد وجود دارند، صرف‌نظر کنیم.

محدودیت‌ها	نحوه کارکرد	تعریف	روش
<ul style="list-style-type: none"> - ابعاد بالا: با افزایش تعداد کلمات منحصر به فرد، اندازه وکتور افزایش یافته و داده‌ها به ابعاد بالایی می‌رسند. - عدم وجود بافت معنایی: ترتیب و بافت معنایی کلمات نادیده گرفته می‌شود، مثلًاً "دوست دارم" و "دارم دوست" به یک شکل نمایش داده می‌شوند. - نمایش پراکنده: بیشتر مقادیر وکتور صفر هستند، زیرا همه کلمات در همه اسناد وجود ندارند. 	<p>هر کلمه در متن به عنوان یک ویژگی نمایش داده می‌شود و مقدار آن تعداد دفعات تکرار کلمه در سند است.</p>	<p>تبديل متن به مجموعه‌ای از کلمات جداگانه و شمارش تعداد تکرار هر کلمه، بدون در نظر گرفتن ترتیب یا بافت معنایی کلمات.</p>	Bag-of-Words (BoW)
<ul style="list-style-type: none"> - ابعاد بالا: مشابه BoW، این روش نیز داده‌ها را به ابعاد بالایی می‌برد. - عدم وجود بافت معنایی: ترتیب و معنای کلمات در نظر گرفته نمی‌شود و تنها به فراوانی و اهمیت آن‌ها توجه می‌شود. - نمایش پراکنده: همچنان داده‌ها پراکنده و بیشتر مقادیر صفر هستند. 	<p>ترکیب دو مؤلفه:</p> <ul style="list-style-type: none"> - TF (فراوانی کلمه): تعداد دفعات تکرار کلمه در سند. - IDF (معکوس فراوانی سند): وزن کلمه براساس نادربودن آن در اسناد، کلمات رایج وزن کمتری می‌گیرند. 	<p>تنظيم وزن کلمات براساس فراوانی آن‌ها در یک سند و اهمیت آن‌ها در کل مجموعه اسناد، به طوری که کلمات خاص وزن بیشتری بگیرند.</p>	TF-IDF

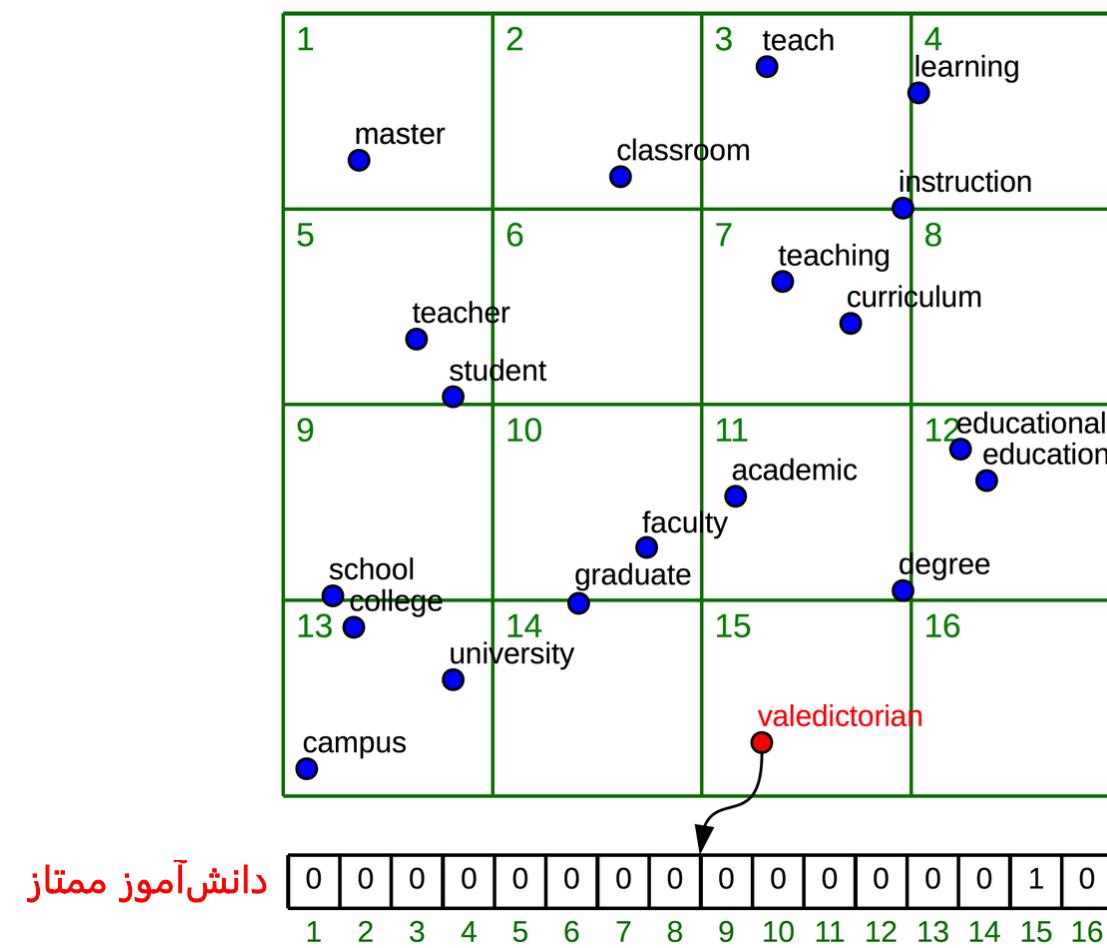
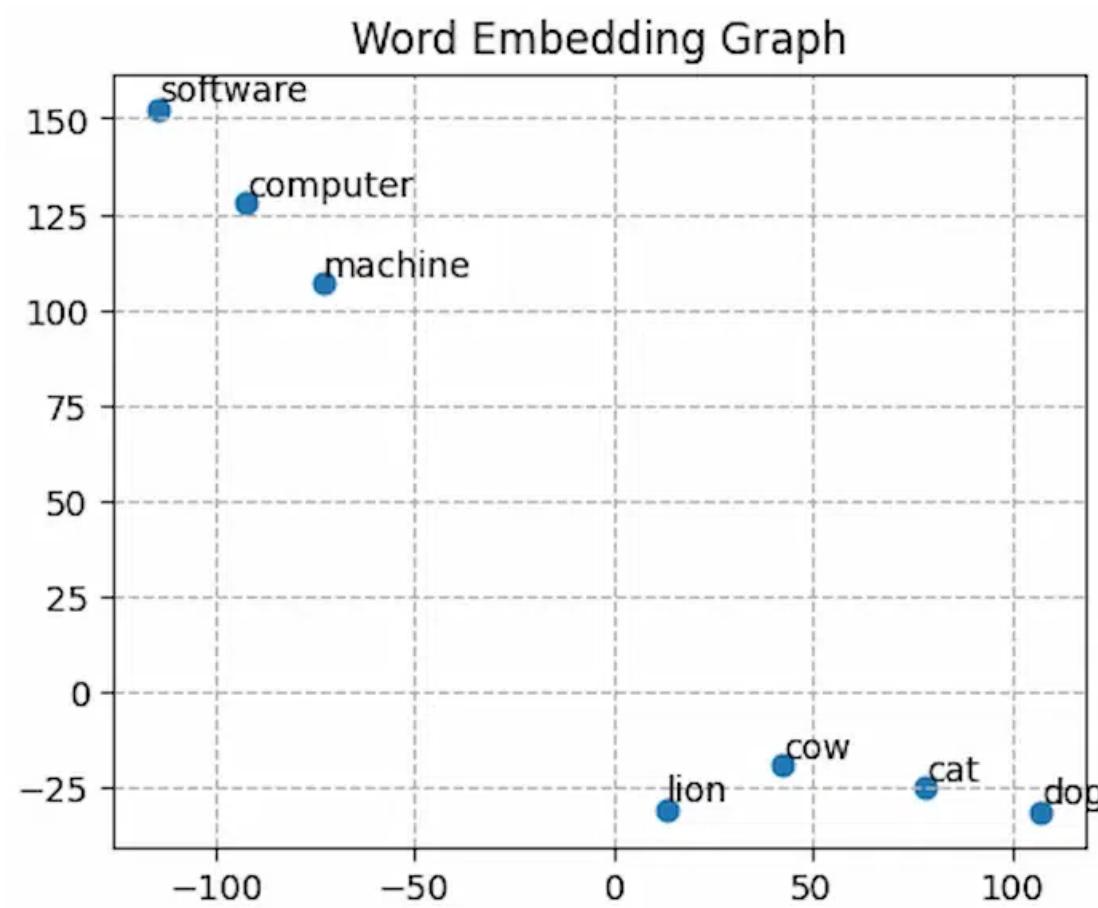
؟ چطوری می‌توانیم از روش‌هایی استفاده کنیم که معنای کلمات و ترتیب‌شون رو هم در نظر بگیرن و

نمایش بهتری از متن داشته باشیم؟

نمایش فشرده و پیوسته‌ای از کلمات هستند که در آن هر کلمه به یک وکتور عددی با ابعاد محدود و ثابت نگاشت می‌شود. این وکتورهای فشرده قادرند بافت معنایی و ارتباطات معنایی بین کلمات را حفظ کنند.

Word Embeddings چیست؟

برای توضیح بهتر، می‌توانید از یک تشبیه واقعی استفاده کنید. فرض کنید هر کلمه شبیه یک نقطه روی نقشه است. در روش‌های سنتی این نقاط به صورت تصادفی در فضای نقشه قرار دارند و هیچ ارتباط معنایی بین نقاط همسایه وجود ندارد. اما در "نقاطی که معنای مشابهی دارند نزدیک به هم قرار می‌گیرند. مثلًا کلماتی مانند "ملک" و "خانه" یا "ماشین" و "وسیله نقلیه" در فضای وکتوری نزدیک به هم خواهند بود



چرا Word Embeddings بهتر است؟

کاهش ابعاد: در Word Embedding به طور معمول از وکتورهایی با ابعاد محدود (مثل ۱۰۰ یا ۳۰۰) استفاده می‌کند و نیازی به ذخیره‌سازی وکتورهای بزرگ مانند BoW ندارد.

حفظ بافت معنایی: این وکتورها می‌توانند بافت معنایی کلمات را نیز نمایش دهند. به عنوان مثال، کلماتی که هم معنی یا مشابه هستند، در وکتورهای Word Embedding به یکدیگر نزدیک هستند.

نمایش پیوسته: برخلاف نمایش‌های پراکنده، Word Embedding بیشتر مقادیر در وکتور عددی هستند، که باعث کارآمدی بیشتر در پردازش می‌شود.

این الگوریتم تلاش می‌کند با مشاهده یک واژه (واژه هدف) کلمات نزدیک به آن را پیش‌بینی کند **Skip-gram** ♦

Word2Vec

این الگوریتم سعی می‌کند با مشاهده کلمات اطراف، واژه هدف را پیش‌بینی کند **Continuous Bag of Words (CBOW)** ♦

هدف اصلی آن است که کلمات را به صورت وکتورهای عددی فشرده نمایش دهد، به طوری که ارتباطات معنایی و بافت آن‌ها را نیز حفظ کند. Word2Vec این مدل، برخلاف روش‌های سنتی، به جای استفاده از تعداد تکرار کلمات، تلاش می‌کند تا بافت معنایی را با استفاده از یادگیری مجاورت کلمات به دست آورد.

Skip-gram ◊

یک روش برای یادگیری وکتورهای کلمات است که هدف آن پیش‌بینی کلمات اطراف یک واژه‌ی هدف مشخص است. این روش در مدل‌های Word2Vec استفاده می‌شود و یکی از پرکاربردترین تکنیک‌ها در پردازش زبان طبیعی است.

هدف و ساختار الگوریتم Skip-gram

هدف این است که با داشتن یک واژه‌ی هدف، کلمات اطراف آن را در یک پنجره‌ی ثابت پیش‌بینی کنیم. برای مثال، اگر جمله‌ی زیر را داشته باشیم:

"جمله: "هوا امروز بسیار خوب است و واژه‌ی هدف "امروز" باشد، مدل سعی می‌کند کلمات اطراف "امروز" را پیش‌بینی کند. در اینجا، فرض کنید طول پنجره‌ی موردنظر برابر با ۲ است. این به این معناست که مدل باید دو کلمه قبل و دو کلمه بعد از واژه‌ی هدف را پیش‌بینی کند.

پس کلمات اطراف "امروز" شامل "هوا" و "بسیار" می‌شود

فرض کنید واژه‌ی هدفی به نام w_t داریم و واژگان اطراف آن که قرار است پیش‌بینی شوند، شامل

$w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$

باشند. هدف الگوریتم این است که احتمالات شرطی زیر را به حداقل برساند :

$$P(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} \mid w_t)$$

به‌طور دقیق‌تر، این احتمال را می‌توان به صورت ضرب احتمال‌های شرطی جداگانه‌ای بازنویسی کرد:

$$P(w_{t-2} \mid w_t) \cdot P(w_{t-1} \mid w_t) \cdot P(w_{t+1} \mid w_t) \cdot P(w_{t+2} \mid w_t)$$

هدف مدل این است که پارامترهای کلمات را طوری تنظیم کند که این احتمال‌ها برای کلمات اطراف واژه‌ی هدف به حداقل مقدار ممکن برسد. این کار با استفاده از شبکه عصبی انجام می‌شود که وکتورهای کلمات را آموخته می‌دهد.

احتمال، اندازه‌گیری این است که چقدر امکان دارد یک رویداد خاص رخ دهد. احتمال معمولاً به صورت یک عدد بین ۰ و ۱ نشان داده می‌شود. عدد ۰ به این معنی است که رویداد غیرممکن است و عدد ۱ نشان می‌دهد که رویداد حتماً رخ می‌دهد. هرچقدر عدد احتمال به ۱ نزدیک‌تر باشد، احتمال رخ دادن رویداد بیشتر است.

مفهوم احتمال

فرض کنید یک سکه داریم. وقتی سکه را پرتاب می‌کنیم، دو نتیجه ممکن وجود دارد: شیر یا خط چون فقط دو نتیجه وجود دارد و هر کدام از آن‌ها به‌طور مساوی ممکن هستند، احتمال رخ دادن هر کدام برابر است با:

$$\frac{1}{2} = 0.5$$

اعداد حالت‌های مطلوب چطور احتمال‌ها را محاسبه می‌کنیم؟
= احتمال رویداد

اعداد کل حالت‌ها

در مثال پرتاب سکه

اعداد حالت‌های مطلوب: ۱ (مثلاً اگر بخواهیم احتمال آمدن "شیر" را محاسبه کنیم)
 اعداد کل حالت‌ها: ۲ (شیر یا خط)

حالا فرض کنید یک تاس شش وجهی داریم (از ۱ تا ۶). اگر بخواهیم احتمال آمدن عدد زوج را محاسبه کنیم

$$\frac{3}{6} = 0.5$$

اعداد حالت‌های مطلوب: ۳ (چون سه عدد زوج داریم: ۲، ۴ و ۶)
 اعداد کل حالت‌ها: ۶ (چون تاس شش وجه دارد)

بنابراین، احتمال آمدن یک عدد زوج برابر است با

در الگوریتم Skip-gram، مشابه مفهوم پرتاب سکه، می‌خواهیم احتمال اینکه کلمات خاصی در اطراف یک کلمه هدف بیانند را محاسبه کنیم. اگر درکی از احتمال به این شکل داشته باشیم، می‌توانیم راحت‌تر بفهمیم که این الگوریتم چطور کار می‌کند: احتمال اینکه کلمات خاصی (مثل "هوا" یا "خوب") در اطراف کلمه هدف (مثل "امروز") دیده شوند، محاسبه می‌شود تا الگوریتم بتواند پیش‌بینی‌های بهتری انجام دهد

مراحل الگوریتم Skip-gram

- ① ساخت زوج‌های هدف و کلمات اطراف: ابتدا جمله را به زوج‌هایی از واژه هدف و واژگان اطراف تقسیم می‌کنیم. در مثال جمله‌ی "هوا امروز بسیار خوب است"، با انتخاب "امروز" به عنوان واژه هدف و پنجره‌ی اندازه ۲، زوج‌های زیر به دست می‌آیند:
 - (امروز، هوا)
 - (امروز، بسیار)
 - (امروز، خوب)
 - (امروز، است)
- ② پیش‌بینی کلمات اطراف با استفاده از وکتورهای کلمات: هر کلمه در متن به یک وکتور عددی تبدیل می‌شود. مدل با استفاده از این وکتور، تلاش می‌کند زوج‌ها را به‌طور صحیح پیش‌بینی کند. این وکتورها ابتدا به صورت تصادفی مقداردهی می‌شوند و سپس طی فرآیند آموزش به‌روزرسانی می‌شوند.
- ③ محاسبه‌ی خطا و بهینه‌سازی: برای هر زوج هدف-کلمه‌ی اطراف، مدل احتمال وقوع کلمه‌ی اطراف را با توجه به کلمه‌ی هدف محاسبه می‌کند و سپس با استفاده از الگوریتم‌های بهینه‌سازی (مثل گرادیان کاهشی) سعی می‌کند خطای بین پیش‌بینی و مقدار واقعی را کاهش دهد

نحوه محاسبه احتمال‌ها در Skip-gram

برای هر کلمه‌ی هدف w_t و کلمه‌ی اطراف w_{t+k} , الگوریتم Skip-gram به دنبال حداکثر کردن تابع احتمال زیر است:

$$P(w_{t+k} | w_t) = \frac{\exp(v'_{w_{t+k}} \cdot v_{w_t})}{\sum_{w=1}^V \exp(v'_w \cdot v_{w_t})}$$

که در آن:

v_w وکتور کلمه‌ی هدف است.

v'_{t+k} وکتور کلمه‌ی اطراف است.

V تعداد کل کلمات در واژگان (Vocabulary) است.

P(weather | today) = ?

Dot product of the vectors "today" and "weather":

- Vector for "today": [0.2, 0.5, 0.1]
- Vector for "weather": [0.3, 0.7, 0.2]

Calculating the dot product:

$$(0.3 \times 0.2) + (0.7 \times 0.5) + (0.2 \times 0.1) = 0.06 + 0.35 + 0.02 = 0.43$$

Therefore, the dot product result for "today" and "weather" is 0.43.

Dot product of the vectors "today" and "good":

- Vector for "good": [0.4, 0.1, 0.5]

Calculating the dot product:

$$(0.4 \times 0.2) + (0.1 \times 0.5) + (0.5 \times 0.1) = 0.08 + 0.05 + 0.05 = 0.18$$

Therefore, the dot product result for "today" and "good" is 0.18.

Calculating the sum for normalization:

Now, to normalize, we find the sum of the computed values for the words "weather" and "good":

$$0.43 + 0.18 = 0.61$$

Calculating the final probability:

The probability $P(\text{weather} | \text{today})$ is given by the dot product of "today" and "weather" divided by the sum for normalization:

$$P(\text{weather} | \text{today}) = \frac{0.43}{0.61} \approx 0.705$$

Therefore, the probability that "weather" appears around "today" is approximately 0.705 or 70.5%.

مثال برای نحوه محاسبه احتمال‌ها در Skip-gram

در این فرآیند، مدل تلاش می‌کند تا برای هر کلمه (مانند "ها") یک وکتور عددی ایجاد کند که خصوصیات معنایی و نحوی آن کلمه را در فضای چندبعدی نمایش دهد.

درنهایت چه طور یک کلمه به وکتور عددی تبدیل می‌شود؟

تعریف فضای چندبعدی

مدل یک فضای چندبعدی با تعداد مشخصی بعد ایجاد می‌کند (معمولاً ۱۰۰ تا ۳۰۰ بعد). هر بعد در این فضا، اطلاعاتی خاص را درباره ارتباطات معنایی کلمه ذخیره می‌کند. این ابعاد به مدل کمک می‌کنند تا رابطه میان کلمات را به صورت عددی درک کند. به عبارت دیگر، هر بُعد می‌تواند به نوعی نمایانگر یک ویژگی معنایی باشد. مثلًا در یک بعد، کلمات مرتبط با "ها" می‌توانند عدد بالاتری داشته باشند تا ارتباط معنایی میانشان مشخص شود.

آموزش مدل و یادگیری وکتورهای کلمات

مدل آموزش می‌بیند که بتواند برای هر کلمه هدف، کلمات اطرافش را پیش‌بینی کند. برای این کار، مدل نیاز دارد تا کلمات را به صورت وکتورهای عددی ذخیره کند.

در طول آموزش، مدل وزن‌هایی را به وکتورهای عددی کلمات اختصاص می‌دهد و این وزن‌ها به روزرسانی می‌شوند تا به تدریج ارتباطات معنایی میان کلمات بهتر درک شود.

به عنوان مثال، اگر کلمات "ها" و "آب و هوا" و "اقليم" در جملات مشابهی تکرار شوند، مدل وزن‌های مشابهی را برای وکتورهای آن‌ها یاد می‌گیرد و آن‌ها را به وکتورهای نزدیک به هم در فضای چندبعدی تبدیل می‌کند.

ایجاد وکتورهای نهایی

بعد از اینکه مدل به تعداد کافی مثال‌های مختلف را دید، وکتورهای عددی کلمات ثبت می‌شوند. هر وکتور به صورت مجموعه‌ای از اعداد که موقعیت کلمه را در فضای معنایی تعیین می‌کند ذخیره می‌شود.

وکتورهای کلمات را می‌توان برای عملیات مختلفی استفاده کرد. مثلًا، از آن‌ها برای اندازه‌گیری شباهت میان کلمات استفاده می‌شود؛ دو کلمه‌ای که وکتورهای مشابه دارند، از نظر مدل شباهت معنایی بیشتری دارند.

به طور خلاصه، در این مرحله مدل، ارتباطات معنایی و نحوی کلمات را با استفاده از یک فضای چندبعدی به وکتورهای عددی تبدیل می‌کند تا بتواند بهتر کلمات را پردازش کند و پیش‌بینی کند.

بردارهای جهانی برای نمایش واژگان، یک روش نمایش کلمات به صورت بردارهای عددی است که به‌طور خاص برای پردازش زبان طبیعی طراحی شده است. هدف اصلی این است که کلمات را بر اساس هموقوعی‌هایشان در یک مجموعه داده بزرگ به نمایش درآورد، به‌طوری که معنای واژگان در قالب بردارهایی در فضای چندبعدی مشخص شود. این روش به عنوان یک مدل توزیع آماری برای نمایش واژگان استفاده می‌شود.

بر اساس مفهوم هموقوعی بین واژگان کار می‌کند. هموقوعی به معنی تعداد دفعاتی است که دو کلمه در یک متن یا مجموعه داده بزرگ، در محدوده نزدیکی به هم ظاهر می‌شوند. با ساخت یک ماتریس هموقوعی که هر سطر و ستون آن نماینده یک واژه است، تمام ارتباطات میان واژگان در سطح جهانی (و نه فقط در محدوده‌های محلی) جمع‌آوری و ذخیره می‌شود. به عبارت دیگر، تلاش می‌کند تا معنای یک کلمه را بر اساس تعداد دفعات هموقوعی آن با سایر کلمات تعیین کند. به عنوان مثال، اگر کلمه «پادشاه» به‌طور مکرر با کلماتی مثل «قصر»، «ملکه» و «تاج» هموقوعی داشته باشد، مدل نتیجه می‌گیرد که بین این کلمات یک ارتباط معنایی وجود دارد و بردارهای آن‌ها در فضای چندبعدی به یکدیگر نزدیک خواهند بود.

GloVe مفهوم

GloVe چگونه کار می‌کند؟

ساختار ریاضی GloVe

یک تابع هزینه به شکل زیر تعریف می‌شود که هدف آن بهینه‌سازی فاصله میان بردارهای کلمات بر اساس تعداد هم‌وقوعی‌ها است:

$$J = \sum_{i,j=1}^V f(X_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2$$

X_{ij} : تعداد هم‌وقوعی بین کلمات i و j

w_i : بردار کلمه i

\tilde{w}_j : بردار متن کلمه j

b_i : باIAS‌ها برای تنظیم دقت مدل

تابع وزنی که اهمیت هم‌وقوعی را نشان می‌دهد و معمولاً برای کم کردن تاثیر هم‌وقوعی‌های نادر: استفاده می‌شود.

این فرمول به GloVe کمک می‌کند تا بردارهای کلمات به گونه‌ای بهینه شوند که کلمات با ارتباطات معنایی قوی، بردارهای نزدیکی داشته باشند و فاصله میان بردارها به میزان هم‌وقوعی آن‌ها وابسته باشد.

Word2Vec	GloVe	ویژگی
مبتنی بر پیش‌بینی از طریق شبکه عصبی	مبتنی بر هم‌وقوعی جهانی واژگان	روش ساخت بردارها
CBOW و Skip-gram مدل‌های	مدل مبتنی بر ماتریس هم‌وقوعی	ساختار مدل
یادگیری به صورت محلی بر اساس زمینه	یادگیری بر اساس توزیع هم‌وقوعی	نوع یادگیری
مناسب برای متون تخصصی و زمینه‌های محدود	مناسب برای تحلیل‌های آماری گسترده	کاربرد
کندتر به دلیل استفاده از شبکه عصبی	نسبتاً سریع‌تر به دلیل استفاده از ماتریس هم‌وقوعی	سرعت یادگیری
منتواند با داده‌های کمتر هم کار کند	نیاز به داده‌های بزرگ با هم‌وقوعی بالا	حجم داده مورد نیاز
نمایش دقیق‌تری از روابط نزدیک بین واژگان	نمایش کلی و آماری از هم‌وقوعی واژگان	دقت در نمایش رابطه‌ها
تحلیل متون خاص و مجموعه داده‌های کوچک	تجزیه و تحلیل متن‌های عمومی و داده‌های بزرگ	مثال کاربردی

