https://web.stanford.edu/class/cs224n/

https://www.youtube.com/watch?v=rmVRLeJRkl4&list=PLoROMvodv4rOSH4v6133s9LFPRHjEmbmJ&index=2

for word embedding if we use one hot ---→ no

در نمایش "one-hot" هر واژه به یک بردار دودویی تبدیل می‌شود، که تنها یک عضو آن برابر با ۱ است و سایر عناصر برابر با ۰ هستند. به عبارت دیگر، هر واژه در یک فضای بلند بُعدی نمایش داده می‌شود و تنها یک بُعد از آن یک است.

مشکلات این نمایش:

عدم درک ارتباطات معنایی: این نمایش اطلاعات معنایی بین واژگان را نمی‌فهمد. دو واژه با این نمایش تماماً مستقل از یکدیگر هستند و هیچ اطلاعاتی در مورد شباهت یا تفاوت‌های معنایی آنها در این نمایش نهفته نیست.

اندازه بالای فضای ویژگان: اگر تعداد واژگان زیاد باشد، این نمایش باعث ایجاد یک فضای بسیار بزرگ می‌شود که نه تنها می‌تواند مشکلات محاسباتی ایجاد کند، بلکه به اندازهٔ کافی اطلاعات مفید برای یادگیری ارائه نمی‌دهد.

عدم درک تفاوت‌ها و شباهت‌ها: این نمایش تفاوت‌ها و شباهت‌های معنایی بین واژگان را نمی‌فهمد. مثلاً فاصله بین دو بردار "one-hot" برابر با ۲ خواهد بود، بدون اینکه به ما بگوید این تفاوت ناشی از چه معناست.

اضافه‌وزنی اطلاعات: این نمایش به هر واژه یک بُعد دارد و اطلاعات معنایی بسیار محدودی ارائه می‌کند. این در حالی است که بسیاری از واژگان در زبان‌ها ارتباطات پیچیده‌تری دارند که با نمایش "one-hot" به درستی نمی‌توان آنها را ادراک کرد.

به همین دلیل، مدل‌های مبتنی بر نمایش‌های جدیدی مانند "word embeddings" (تعبیر واژگان) استفاده می‌شوند که این مشکلات را حل کرده و واژگان را در یک فضای چگال و معنایی بهتر نمایش می‌دهند.

ی روش دیگه میشه featurized word embedding که با یک سری فیچر بیایم هر word رو نمایش بدیم اینطوری مفهوم sim را داریم

wordnet

WordNet [Miller, 1995]:

WordNetیک پایگاه داده لغت‌نامه است که ارتباطات معنایی بین واژگان را ارائه می‌دهد.

برای واژگان، مترادف‌ها، زیرمجموعه‌ها(hyponyms) ، و ارتباطات معنایی دیگر را برچسب‌گذاری می‌کند.

UniMorph [Batsuren et al., 2022]:

UniMorphیک منبع دیگر است که برای اطلاعات مربوط به ساختار زیر واژگانی (morphology)در بسیاری از زبان‌ها برچسب‌گذاری می‌کند.

اطلاعات مربوط به ساختار زیر واژگانی شامل جزئیاتی مانند پیشوند، پسوند، یا تغییرات دیگر در واژه‌ها می‌شود.

مشکلات:

کمبود منابع انسانی:

منابع با برچسب انسانی همواره در حجم واژگان نسبت به روش‌هایی که می‌توانند واژگان را از منابع متنی طبیعی استخراج کنند، کمبود دارند.

به روزرسانی این منابع هزینه‌بر است و همواره ناقص می‌مانند.

تضاد بین ابعاد و کارایی تعبیه (Embedding):

برای نمایش همه دسته‌ها نیاز به بردار با ابعاد بسیار بالا (بیشتر از اندازه واژگان) وجود دارد.

روش‌های نورونی مدرن که عمدتاً با بردارهای فشرده (dense vectors) کار می‌کنند، با این بردارهای بزرگ به خوبی هماهنگی ندارند.

کمبود دقت در نمایش‌های ایده‌آل برای متن:

اشاره به این نکته شده که دیدگاه انسانی در مورد نمایش مناسب برای متن، به عنوان یک پیشنهاد مکرر در دوره، معمولاً عملکرد کمتری دارد نسبت به روش‌هایی که به داده امکان می‌دهند بیشتری از جزئیات را مشخص کنند، حداقل زمانی که داده به میزان زیادی برای یادگیری وجود دارد.

similarity

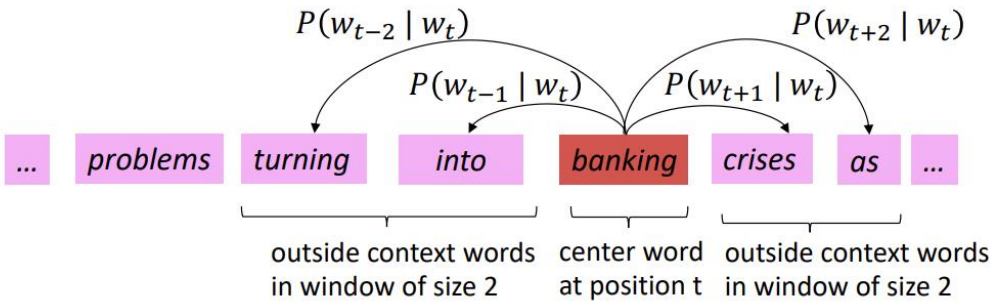
context word

distributional semantic

You shall know a word by the company it keeps.

word2vec 2013

Example windows and process for computing $P(w_{t+j} \mid w_t)$



Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t. Data likelihood:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

- **Question:** How to calculate $P(w_{t+j} \mid w_t; \theta)$?
- **Answer:** We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2vec: prediction function

② Exponentiation makes anything positive

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ ← Open region

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

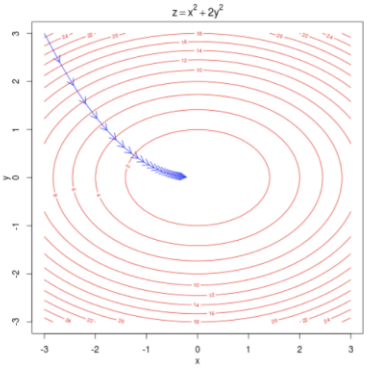
But sort of a weird name because it returns a distribution!

To train the model: Optimize value of parameters to minimize loss

To train a model, we gradually adjust parameters to minimize a loss

- Recall: θ represents **all** the model parameters, in one long vector
- In our case, with *d*-dimensional vectors and *V*-many words, we have →
- Remember: every word has two vectors

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



- We optimize these parameters by walking down the gradient (see right figure)
- We compute all vector gradients!

Remember: every word has two vectors ---→ u and v

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w v_c)}$$

$$= \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c)}_{①} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)}_{②}$$

① $\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) = \frac{\partial}{\partial v_c} u_o^T v_c = u_o$

Vector! Not high school single variable calculus

inverses

You can do things one variable at a time, and this may be helpful when things get gnarly.

$$\forall j \quad \frac{\partial}{\partial (v_c)_j} u_o^T v_c = \frac{\partial}{\partial (v_c)_j} \sum_{i=1}^{d} (u_o)_i (v_c)_i$$

$$= (u_o)_j$$

Each term is zero except when $i = j$

7

--→ hint:

Apply the chain rule where $u = e^x$

$$\frac{1}{\ln(10)} \cdot \frac{d}{dx}\left(\ln(e^x)\right)$$

- Replace e^x with u
- Apply the chain rule where $u = e^x$
- Take the derivative of a logarithm expression
- Replace u with e^x

$$\frac{1}{\ln(10)} \cdot \frac{1}{e^x} \cdot \frac{d}{dx}\left(e^x\right)$$

② $\frac{\partial}{\partial v_c} \underbrace{\log}_{f} \underbrace{\sum_{w=1}^{V} \exp(u_w^T v_c)}_{z = g(v_c)}$

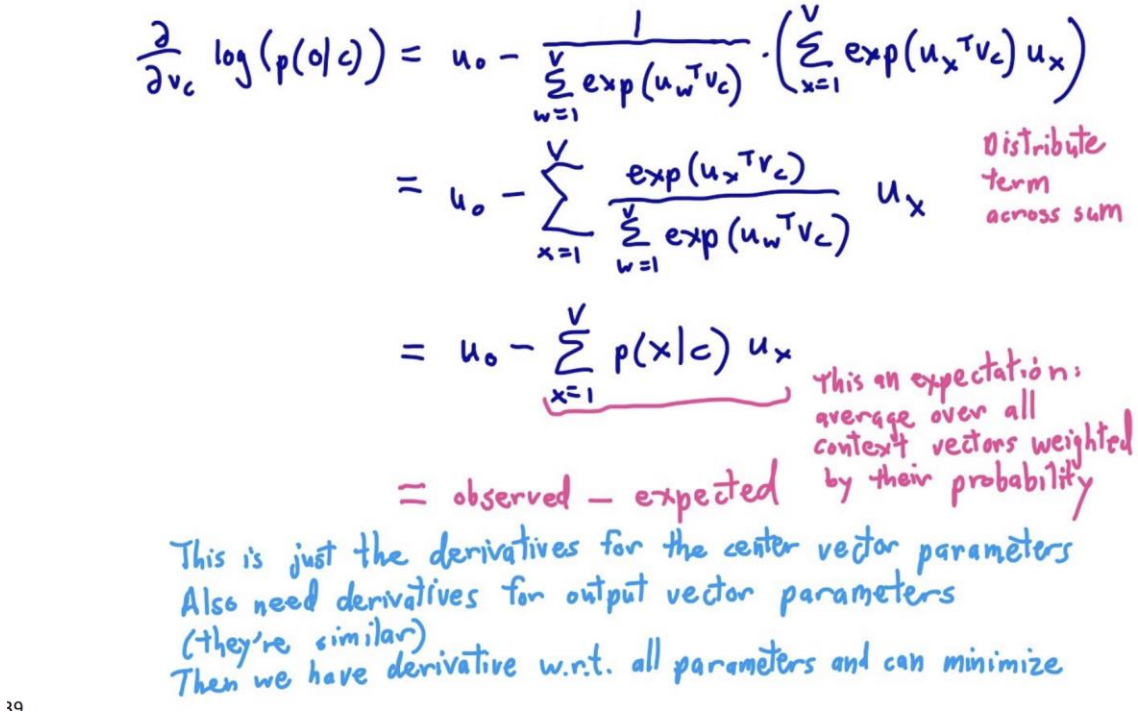
$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \frac{\partial}{\partial v_c} \sum_{x=1}^{V} \exp(u_x^T v_c)$ — Important to change index

$\frac{\partial}{\partial v_c} f(\overbrace{g(v_c)}^{z}) = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial v_c}$ Use chain rule

$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \left(\sum_{x=1}^{V} \frac{\partial}{\partial v_c} \underbrace{\exp}_{f} \underbrace{(u_x^T v_c)}_{z = g(v_c)} \right)$ Move deriv inside sum

$\left(\sum_{x=1}^{V} \exp(u_x^T v_c) \frac{\partial}{\partial v_c} u_x^T v_c \right)$ Chain rule

$\left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x \right)$



به این مدلا میگن "Bag of words" چون به همه اعضای context میخواد prob بالایی بده حالا مهم نی کجای context باشه

گرفتاری:

- **Problem**: $J(\theta)$ is a function of **all** windows in the corpus (potentially billions!)
  - So $\nabla_\theta J(\theta)$ is very expensive to compute
- You would wait a very long time before making a single update!

- **Very** bad idea for pretty much all neural nets!

حل:

- **Solution:** **Stochastic gradient descent (SGD)**
  - Repeatedly sample windows, and update after each one
- Algorithm:

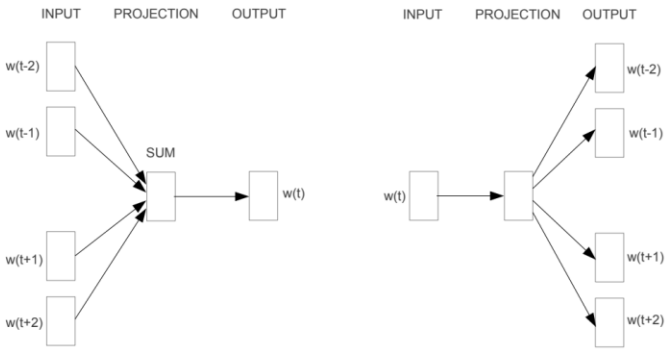
Mini Batch Gradient Descent

```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J,window,theta)
    theta = theta - alpha * theta_grad
```

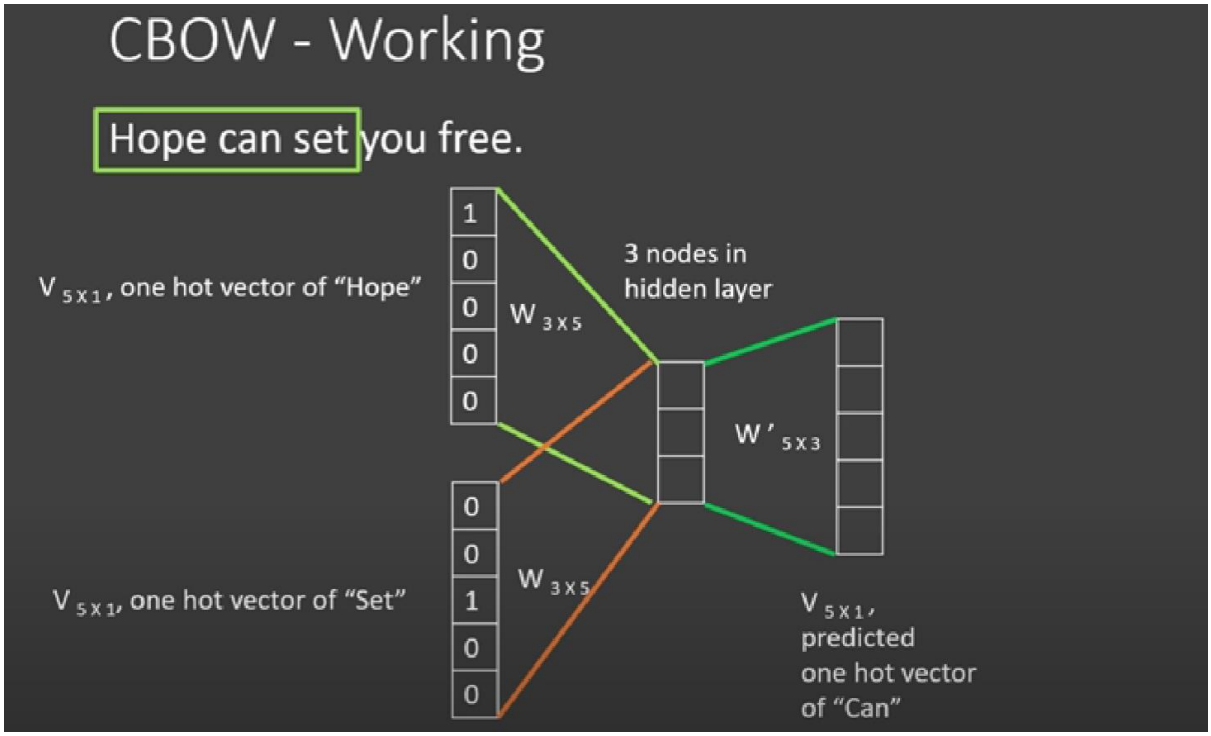
دوتا روش داریم

1. Skip-grams (SG)
   Predict context (“outside”) words (position independent) given center word
2. Continuous Bag of Words (CBOW)
   Predict center word from (bag of) context words

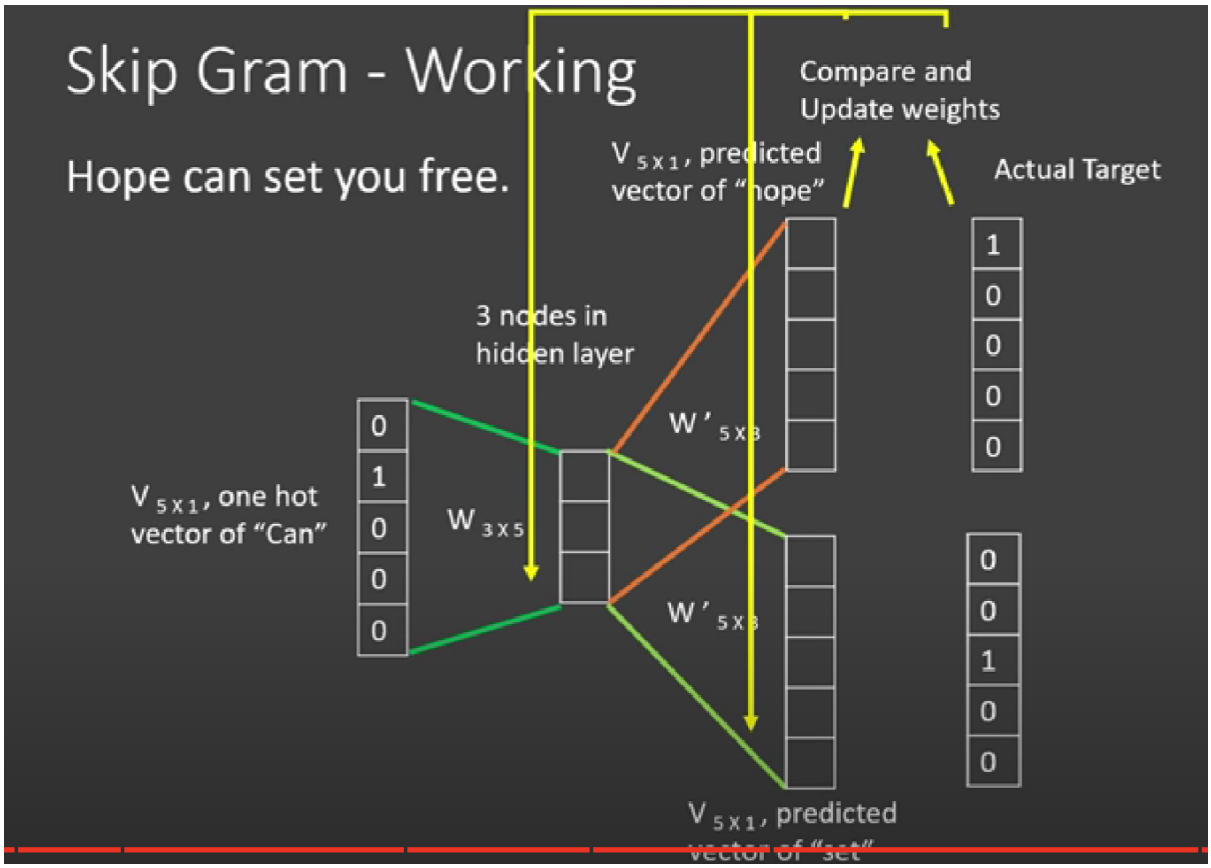
معماری در مقاله:



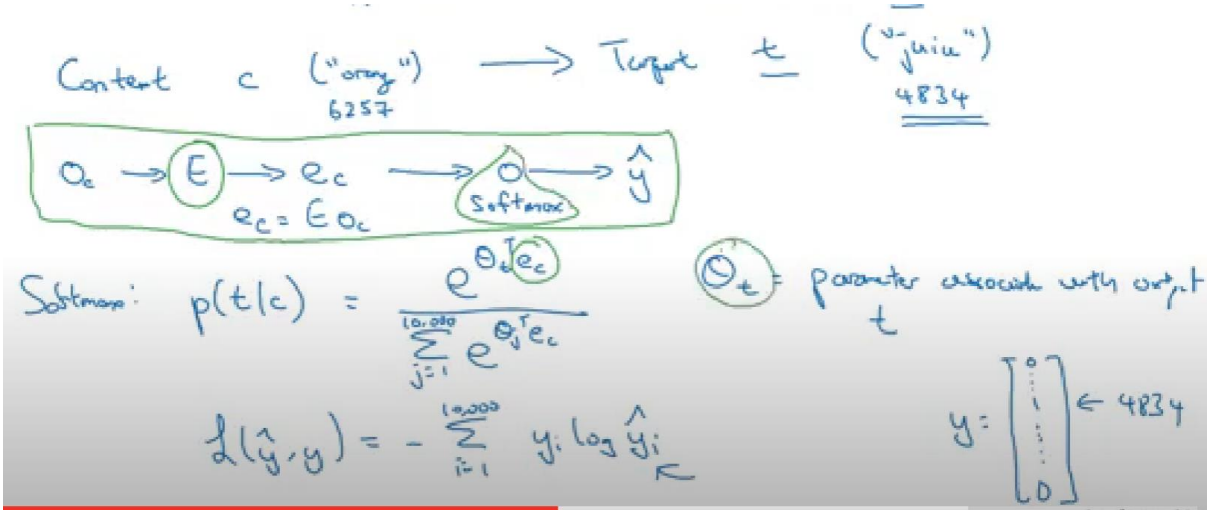
مثال :


CBOW - Working
Hope can set you free.
V 5X1, one hot vector of "Hope"
1
0
0
0
0
W 3X5
3 nodes in hidden layer
W ' 5X3
0
0
1
0
0
W 3X5
V 5X1, one hot vector of "Set"
V 5X1, predicted one hot vector of "Can"

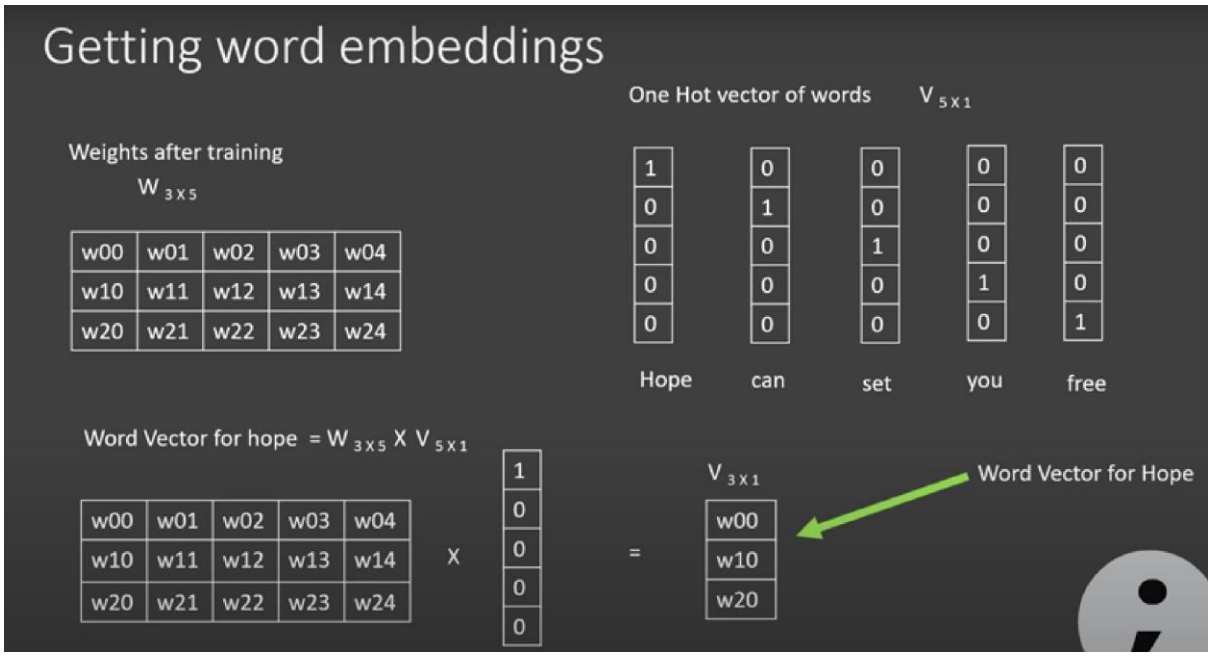
مثال :


Skip Gram - Working
Hope can set you free.
Compare and Update weights
V 5X1, predicted vector of "hope"
Actual Target
1
0
0
0
0
3 nodes in hidden layer
W ' 5X3
0
1
0
0
0
V 5X1, one hot vector of "Can"
W 3X5
W ' 5X3
0
0
1
0
0
V 5X1, predicted vector of "set"

البته یک چیز جالب دیگه هم فهمیدم ما اینجا میخوایم کلمه بعدی رو حدس بزنیم بنابراین میام میگیم به ازای یک context word که داریم بیا و target word منو پیدا کن و اندرو میگف ما میایم context word رو تبدیل میکنیم به one hot میدیم به word embedding تبدیل میشه به e ما حالا اینو میدیم یک شبکه و میگیم کلمه بعدی رو حدس بزن و اون میاد نزدیک ترین کلمه رو حدس میزنه



در واقع اینطوری e را پیدا میکنیم:



الان قضیه ای که وجود داره اینه در skip gram ما اخرش ی soft max داریم که حیلی هزینه بره

یک راه اینه مدل سلسله مراتبی soft max رو داشته باشیم --- > مثلا بگیم جواب در نصفه اوله بعد بگیم ایا در یک چهارو اوله و ....

ی حرف جالبی که زده اینه که مسئله ای که برای ما اهمیت داره اینه که ایا دوتا word باهم همسایه هستند یا نه ؟

یعنی بیایم مسئله چند کلاسمونو به مسئله دو کلاسه تبدیل کنیم

Soft max ------ > sigmoid

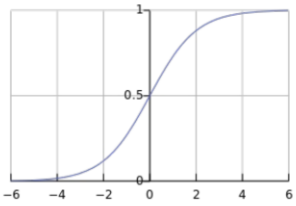
اینجاست که negative sampling میاد

- Introduced in: "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013)
- Overall objective function (they maximize): $J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j\sim P(w)}\left[\log \sigma\left(-u_j^T v_c\right)\right]$$

sigmoid
rather than softmax

- The logistic/sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$ (we'll become good friends soon)
- We maximize the probability of two words co-occurring in first log and minimize probability of noise words in second part



در واقع ماکزیمم میکنیم احتمال اون کلمه ای که کنارشه و مینیمم میکنیم احتمال noise ها رو

$$J_{neg-sample}(\boldsymbol{u}_o, \boldsymbol{v}_c, U) = -\log \sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - \sum_{k\in\{K \text{ sampled indices}\}} \log \sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c)$$

فانشن loss میشه این ریختی

Sample with P(w)=U(w)$^{3/4}$/Z, the unigram distribution U(w) raised to the 3/4 power (We provide this function in the starter code).

The power makes less frequent words be sampled more often

احتمال هم اینطوری انتخاب میکنیم ----- > چون کلمات انگلیسی of و the اینا زیاد داره

لینک مقاله ها:

https://arxiv.org/pdf/1301.3781.pdf

https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

ی قضیه دیگ داریم co-occurrence matrix

این ریختی:

Example: Window based co-occurrence matrix

- Window length 1 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)

- Example corpus:
  - I like deep learning
  - I like NLP
  - I enjoy flying

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

در واقع دو نوع رویکرد وجود داره یکی window بیس یکی روی کل doc

Building a co-occurrence matrix *X*

- 2 options: windows vs. full document
- Window: Similar to word2vec, use window around each word → captures some syntactic and semantic information (“word space”)
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis” (“document space”)

بدبختی: یک ماتریس گنده sparse و دارای اطلاعات کم و نویزی

راه حل:Low-dimensional vectors

فان فکتز

واریانس دامنه تغیرات هر متغیر نسبت ب خودشه

کوواریانس دامنه تغیرات نسبت به بقیه متغیر ها

بردار ویژه هم برداریه ک میاد حاصل ضرب دو تا ماتریس رو خطی میکنه

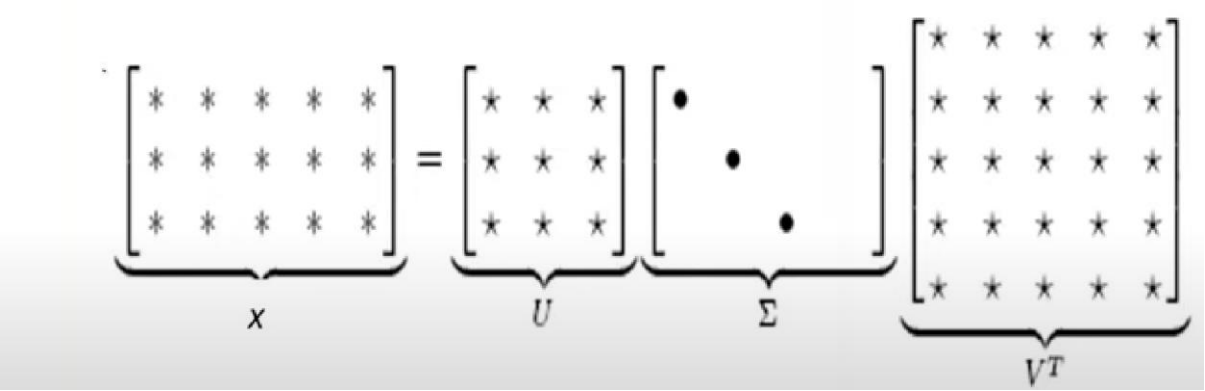
به اون عدده هم که تو بردار ویژه ضرب میشه میگن مقدار ویژه

دترمینان هم معکوس یک ماتریسه

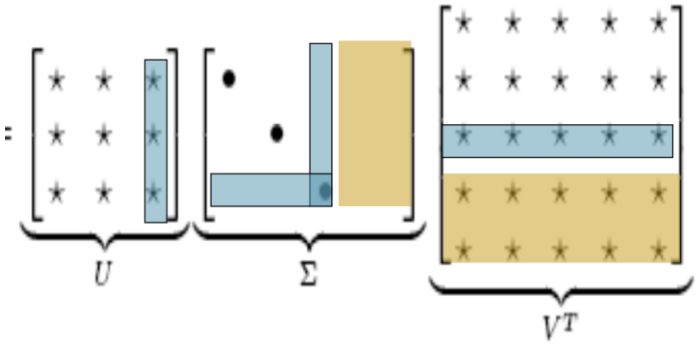
حالا ترانهاده چیه میاد جای سطر و ستونا رو عوض میکنه

ایده pca اینه اون فیچر های correlated رو با ایده بردار ویژه اینا تبدیل به یک ویژگی کن

اما svd میاد ماتریسه ما رو تبدیل میکنه به حاصل ضرب ی ماتریس متفپقارن و قطری و متقارن



اعداد در ماتریس قطری با یک order هستند پس ما میتونیم اینطوری حذف کنیم

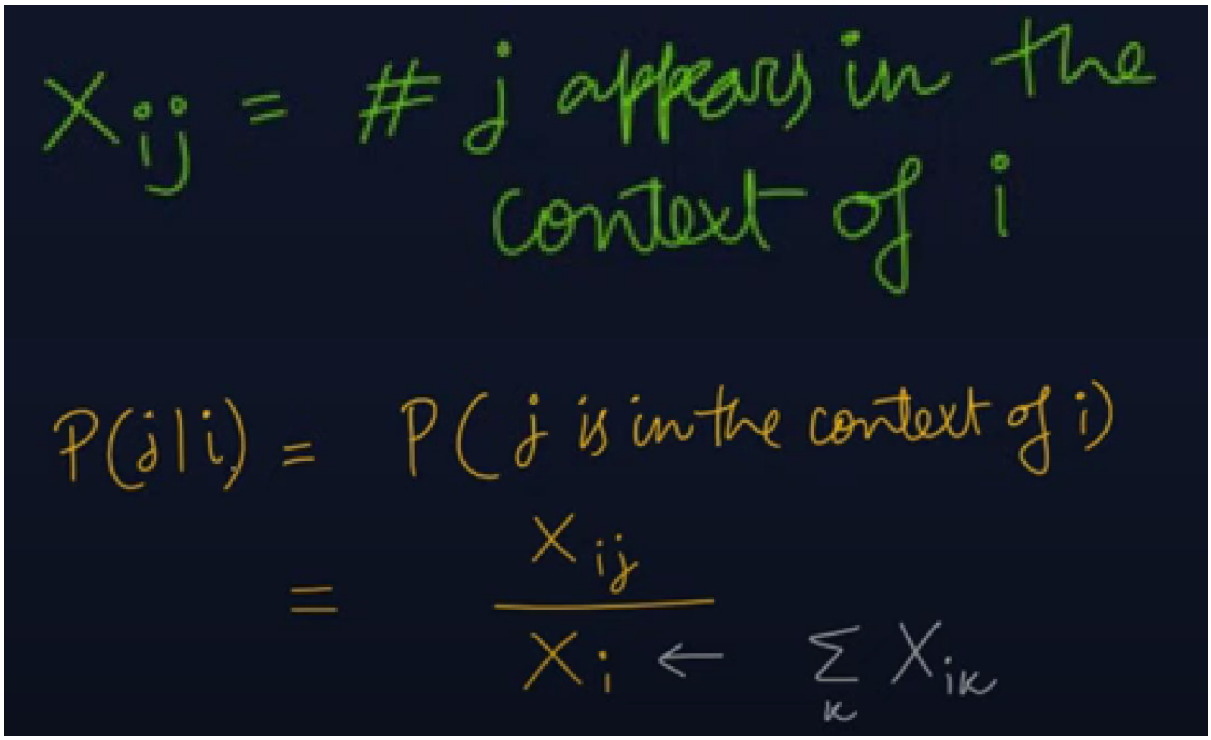


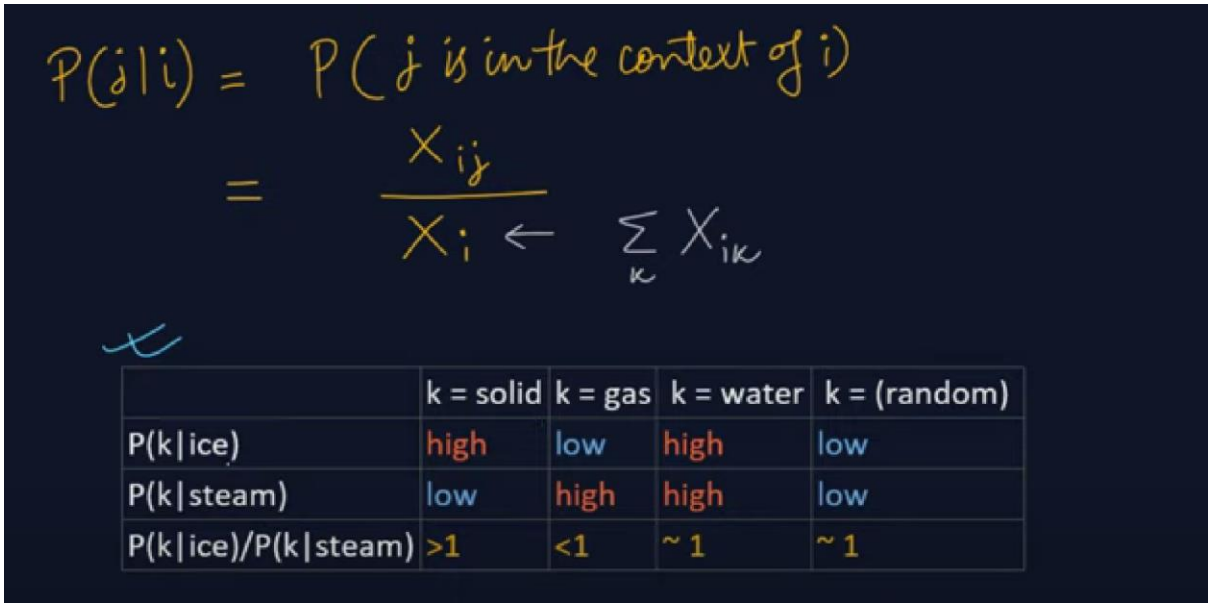
اما اگر ما یک Co-occurrence vector داشته باشیم بیایم روش ی svd بزنیم اونقد خوب کار نمیکنه چون ما کلمات خیلی پرتکراری همچون the و and و ... داریم

راه حل COALS :

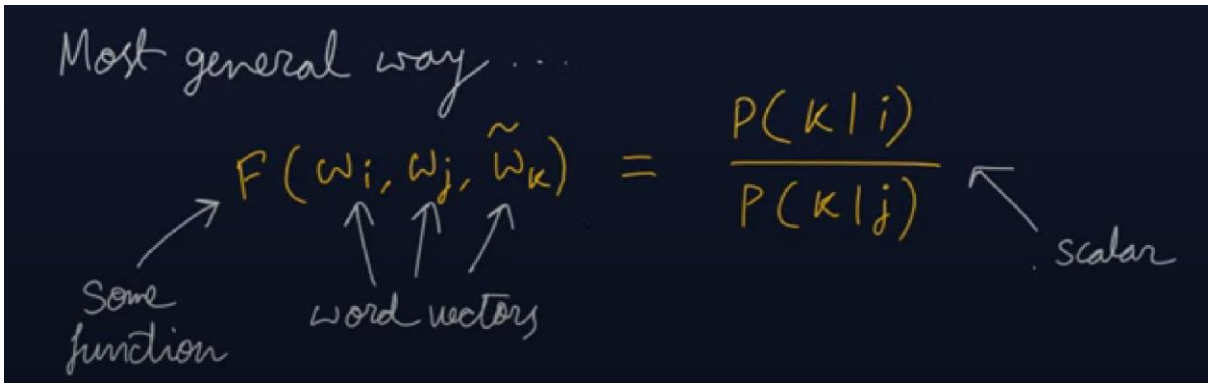
- Scaling the counts in the cells can help ***a lot***
  - Problem: function words (*the, he, has*) are too frequent → syntax has too much impact. Some fixes:
    - log the frequencies
    - min(*X*, *t*), with *t* ≈ 100
    - Ignore the function words
- Ramped windows that count closer words more than further away words
- Use Pearson correlations instead of counts, then set negative values to 0
- Etc.

Glove:

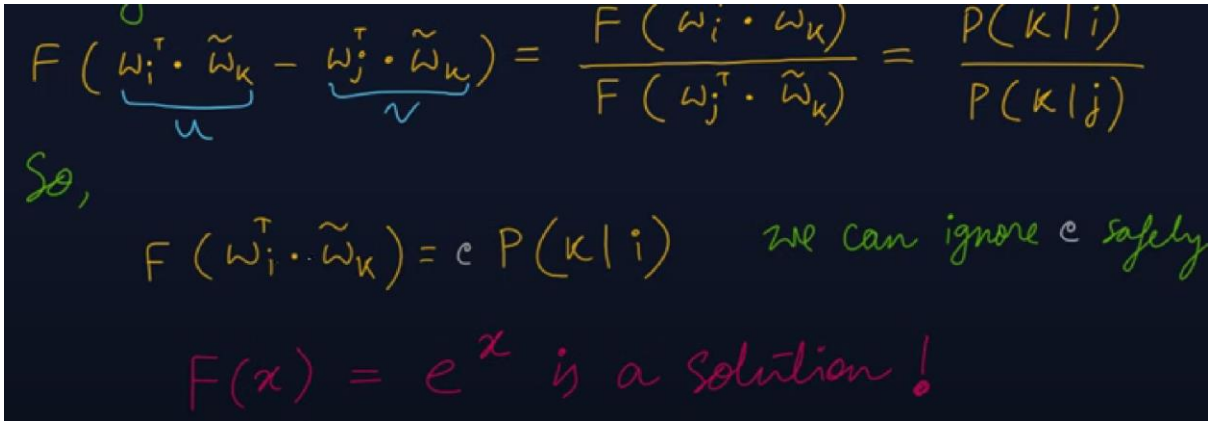




برای حل بزرگ بودن ماتریس:

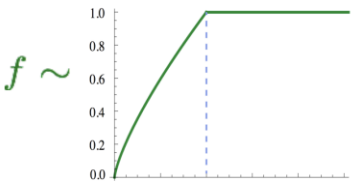


حالا چطوری F رو بدست آوریم:



پس loss function

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



- Fast training
- Scalable to huge corpora

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

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

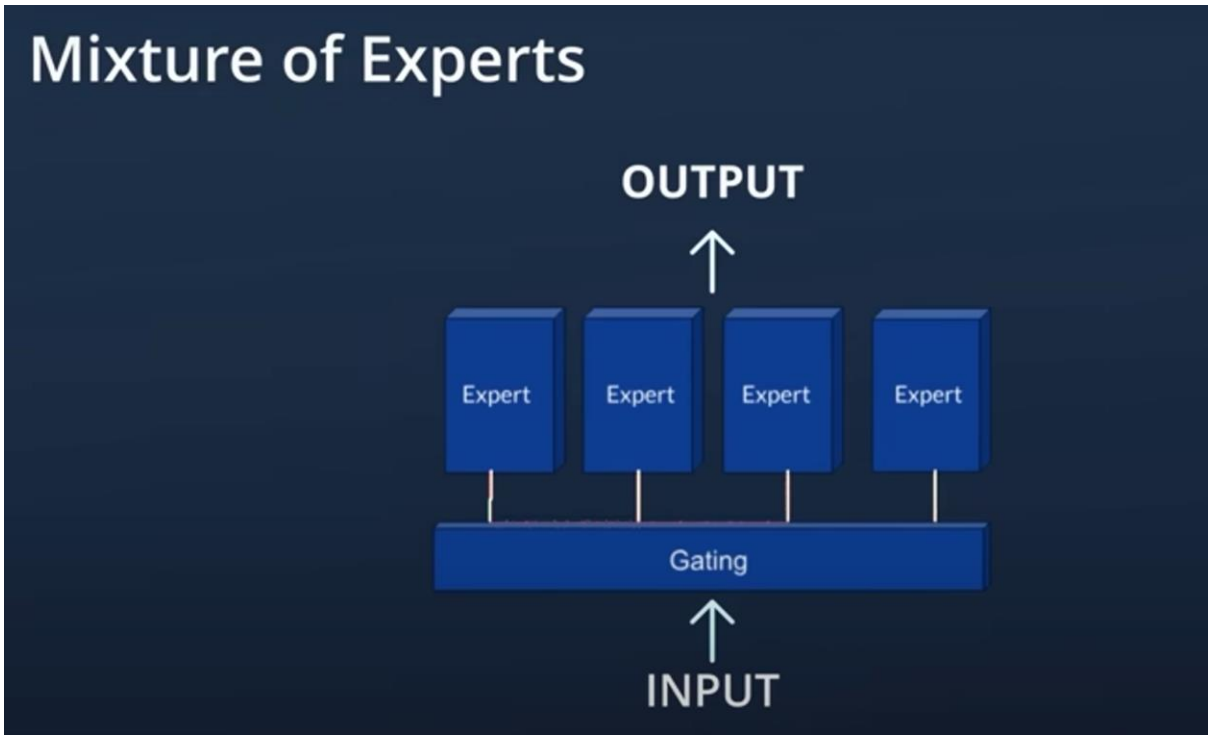
($x_{\max} = 100$, $\alpha = 3/4$)

مدل‌های زبانی را می‌توان به دو صورت Intrinsic یا Extrinsicارزیابی کرد.

Intrinsic: شامل مقایسه مدل Word Embeddings با یک مدل مرجع) مانند lexical database)می‌شود.

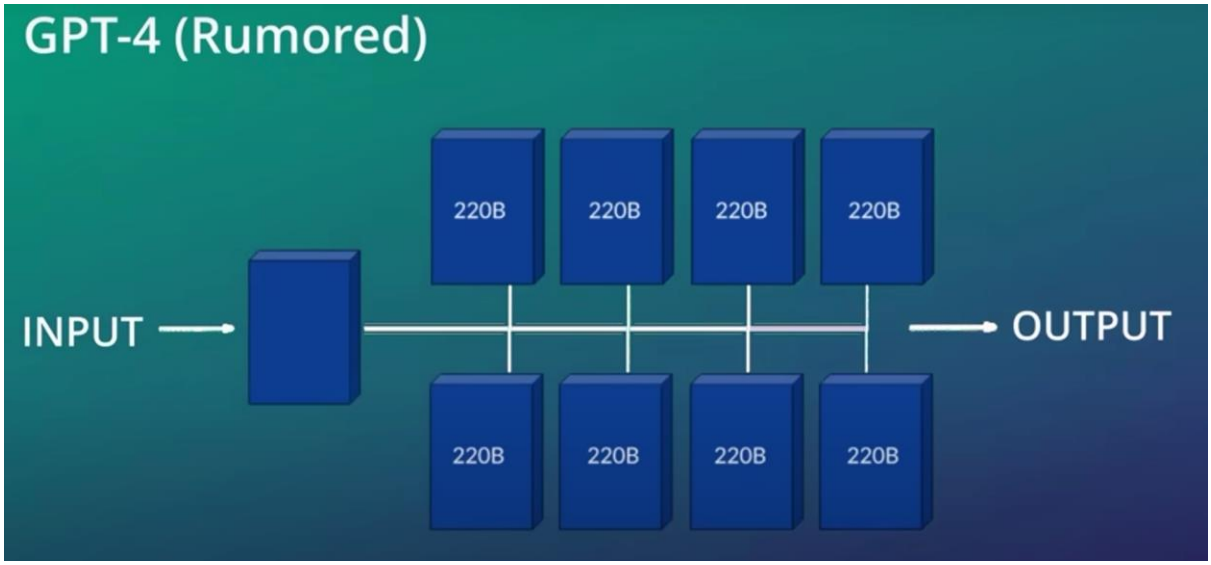
Extrinsic: برای ارزیابی مدل، نتایج آن را در یک تسک یا پروژه‌ای مانند text classification، machine translation وsummarization استفاده کنیم

میسترال :



یک تعدادی expert داریم اینجا هشت تا هفت میلیاردی و هر کدوم متخصص یک چیزی هستند

و وقتی یک تسک میاد gating تصمیم میگیره بره سمت کدوم expert شایعه :



Fine Tuning LLMs with Instruction

What is instruction Tuning?

Instruction tuning represents a specialized form of fine-tuning in which a model is trained using pairs of input-output instructions

Prepare training data:

- Use existing datasets or create new ones.
- Use prompt template libraries to format data as instructions.
- Divide the data into training, validation, and test sets.

Fine-tune the LLM:

- Select prompts from the training set.
- Pass prompts to the LLM and generate completions.
- Compare completions to responses in training data.
- Calculate loss using the cross-entropy function.
- Update model weights using backpropagation.
- Repeat for multiple batches and epochs.

Evaluate LLM performance:

- Use validation data set to calculate validation accuracy.
- Use test data set to calculate test accuracy.

catastrophic forgetting

Catastrophic forgetting happens because the full fine-tuning process modifies the weights of the original LLM. While this leads to great performance on a single fine-tuning task, it can degrade performance on other tasks.

https://medium.com/@veer15/the-hitchhikers-guide-to-instruction-tuning-large-language-models-d6441dbf1413#:~:text=Instruction%20tuning%20represents%20a%20specialized,Output%3A%20%E2%80%9CEnglish%2C%20French%E2%80%9D

لینک باحال برای fin ai

https://github.com/Sahaj777/Mastering-AI-in-Finance?tab=readme-ov-file#llms