Word Representation
Part II

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

Slides are adopted from the Stanford course ‘NLP with DL’ by C. Manning & the book ‘Speech and Language Processing’ by D. Jurafsky and J. H. Martin

Released on July 16, 2024
https://mistral.ai/news/codestral-mamba/

Codestral Mamba

Mistral AI collaborates with Mamba team to release this 7B non-transformer LLM trained for coding tasks.

Codestral Mamba is now the best code-LLM with fewer than 10B parameters, surpassing the transformer-based LLMs of similar size.

Its performance is also comparable to larger transformer-based code-LLMs like CodeLlama (34B) and Codestral (22B).

Codestral Mamba is tested on in-context retrieval capabilities up to **256k tokens** !!!

Image source: Medium blog, Emmanuel Mark Ndaliro

Count-based vs Prediction-based

Count-based
• Fast training • Efficient usage of statistics
• Primarily used to capture word similarity • Disproportionate importance given to large counts

Count-based vs Prediction-based

Count-based	Prediction-based
• Fast training • Efficient usage of statistics	• Scales with corpus size • Inefficient usage of statistics
• Primarily used to capture word Similarity • Disproportionate importance given to large counts	• Generate improved performance on other tasks • Can capture complex patterns beyond word similarity

GloVe – Global Vectors

Crucial insight: Ratios of co-occurrence probabilities can encode word meaning

pivot

fashion

	$x = solid$	$x = gas$	$x = water$	$x = random$
$P(x \mid ice)$	large	small	large	small
$P(x \mid steam)$	small	large	large	small
$\frac{P(x \mid ice)}{P(x \mid steam)}$	large	small	~1	~1

Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors for Word Representation", 2014

GloVe – Global Vectors

Crucial insight: Ratios of co-occurrence probabilities can encode word meaning

	$x = solid$	$x = gas$	$x = water$	$x = random$
$P(x \mid ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(x \mid steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\frac{P(x \mid ice)}{P(x \mid steam)}$	8.9	8.5×10^{-2}	1.36	0.96

Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors for Word Representation", 2014

Co-occurrence Matrix

- Let us denote the co-occurrence matrix as **X**.

count	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

Compute P(j | i) from X, for two words i and j in the corpus.

$$P(j|i) = \frac{X_{ij}}{\sum_j X_{ij}} = \frac{X_{ij}}{X_i}$$

Learn Word Vectors Based on These Counts

- For the two words, *i* and *j*, assume their corresponding representation vectors are w_i and w_j, respectively.
- $$\underbrace{w_i^T w_j}_{\text{Similarity between words i and j}} = \log \underbrace{P(j|i)}_{\text{How likely is j to occur in the context of i}}$$
- $$w_i^T w_j = \log \frac{X_{ij}}{X_i} = \log X_{ij} - \log X_i \qquad \dots (1)$$

Similarly, $w_j^T w_i = \log \frac{X_{ij}}{X_j} = \log X_{ij} - \log X_j \qquad \dots (2)$

$i \quad j$

$w_i \quad w_j$

$w_i w_j \sim \log(P(j|i))$

$\frac{X_{ij}}{X_i}$

Learn Word Vectors Based on These Counts

- $w_i^T w_j = \log \frac{X_{ij}}{X_i} = \log X_{ij} - \log X_i$ ... (1)

Similarly, $w_j^T w_i = \log \frac{X_{ij}}{X_j} = \log X_{ij} - \log X_j$ ... (2)

- Adding (1) and (2):

$$\cancel{2}\, w_i^T w_j = \cancel{2} \log X_{ij} - \log X_i - \log X_j$$

$$\Rightarrow w_i^T w_j = \log X_{ij} - \frac{1}{2} \log X_i - \frac{1}{2} \log X_j$$

i

j

Learn Word Vectors Based on These Counts

$$w_i^T w_j = \log X_{ij} - \frac{1}{2}\log X_i - \frac{1}{2}\log X_j$$

- $\log X_i$ and $\log X_j$ depends only on *i* and *j* respectively – can be thought of as word-specific biases
  - These are made learnable (considered as biases)

$$w_i^T w_j = \log X_{ij} - b_i - b_j$$
$$\Rightarrow w_i^T \; w_j + b_i + b_j = \log X_{ij}$$

$(w_i^T w_j - \log X_{ij})^2$

- w_i, w_j, b_i, b_j are the learnable parameters
- **Loss function:** $min_{w_i, w_j, b_i, b_j} \sum_{i,j} (w_i^T w_j + b_i + b_j - \log X_{ij})^2$

Learn Word Vectors Based on These Counts

Loss function: $min_{w_i,w_j,b_i,b_j} \sum_{i,j} (w_i^T w_j + b_i + b_j - \log X_{ij})^2$,

- **Problem:** Gives equal weightage to every co-occurrence
- **Ideally, rare and very frequent co-occurrences should have lesser weightage**
- **Modification:** Add a weighting function $f(x)$.
- **Modified loss function:** $min_{w_i,w_j,b_i,b_j} \sum_{i,j} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$

What can f possibly be?

Weighting function

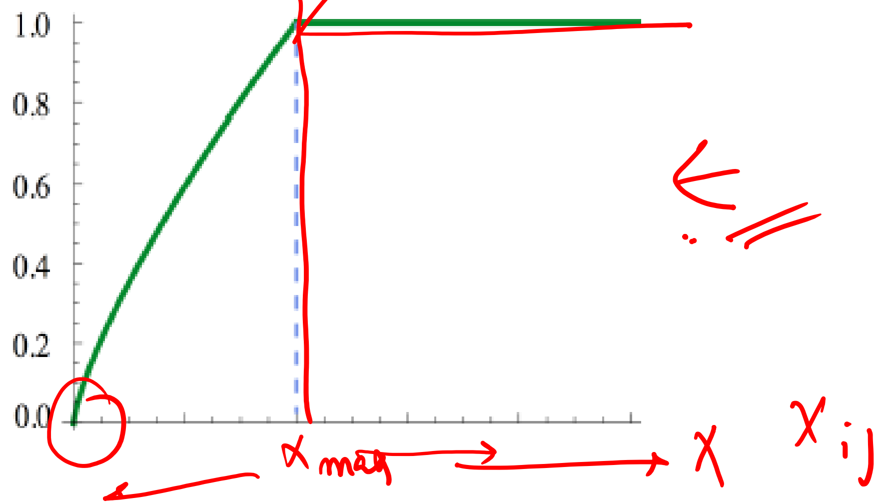
$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

α can be chosen empirically for a given dataset.

Properties of f:

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log^2 x$ is finite.
2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.
3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

$f \sim$



GloVe: Advantages

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors

Details About GloVe

Original paper: https://nlp.stanford.edu/pubs/glove.pdf

bi : Ch. = Del : ____

Father : Comp = M : ____

Man : Dr = W : ____

Blogs with easy explanations:

- https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db89639
- https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/?fbclid=IwAR3-pws3-K-Snfk6aqbixdxS8zFf-uuPDJ_0ipb94kWeygrdCSEqE9HWmNs
- https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010

We will see how we can use these separately trained word embeddings (or train/update embeddings on-the-fly) as we perform language modeling using **Neural Nets**!