Text Representation Learning

Machine Learning Course - CS-433
Dec 6, 2023
Martin Jaggi
Last updated on: December 3, 2023

□

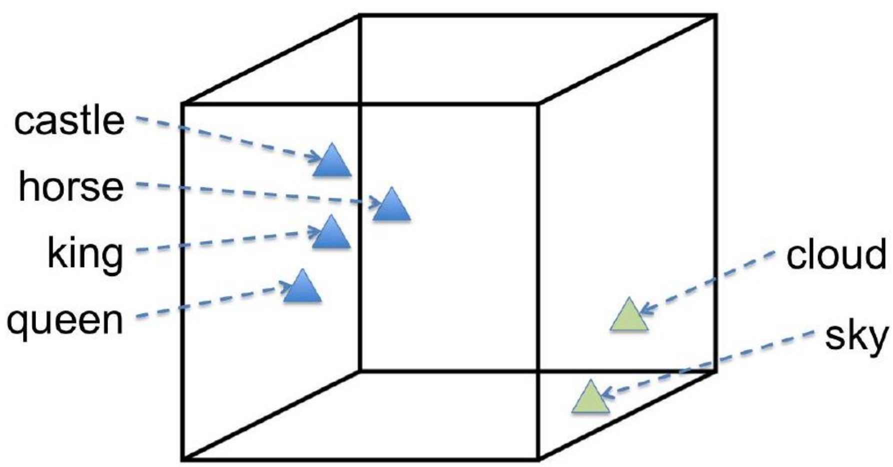
Motivation

Finding numerical representations for words is fundamental for all machine learning methods dealing with text data.

Goal: For each word, find mapping (embedding)

$$w_i \mapsto \mathbf{w}_i \in \mathbb{R}^K$$

Representation should capture semantics of the word.



Constructing good feature representations (= representation learning) benefits all ML applications.

The Co-Occurence Matrix

A big corpus of un-labeled text can be represented as the co-occurrence counts $n_{ij} :=$ #contexts where word w_i occurs together with word w_j.

	1	1		
		3		
	1			
	2		1	
1				1
		1		
	1		1	1

Needs definition of

- Context e.g. document, paragraph, sentence, window
- Vocabulary

$$\mathcal{V} := \{w_1, \ldots, w_D\}$$

For words $w_d = 1, 2, \ldots, D$ and context words $w_n = 1, 2, \ldots, N$, the co-occurence counts n_{ij} form a very sparse $D \times N$ matrix.

Learning Word-Representations (Using Matrix Factorization)

Find a factorization of the cooccurence matrix!

Typically uses log of the actual counts, i.e. $x_{dn} := \log(n_{dn})$.

We will aim to find $\mathbf{W}, \mathbf{Z}$ s.t.

$$\mathbf{X} \approx \mathbf{W}\mathbf{Z}^\top$$

So for each pair of words (w_d, w_n), we try to 'explain' their co-occurence count by a numerical representation of the two words

- in fact by the inner product of the two feature vectors $\mathbf{W}_{d:}, \mathbf{Z}_{n:}$.

$$\min_{\mathbf{W},\mathbf{Z}} \mathcal{L}(\mathbf{W}, \mathbf{Z}) := \frac{1}{2} \sum_{(d,n)\in\Omega} f_{dn} \left[x_{dn} - \left(\mathbf{W}\mathbf{Z}^\top\right) dn\right]^2$$

where $\mathbf{W} \in \mathbb{R}^{D\times K}$ and $\mathbf{Z} \in \mathbb{R}^{N\times K}$ are tall matrices, having only $K \ll D, N$ columns.

The set $\Omega \subseteq [D] \times [N]$ collects the indices of non-zeros of the count matrix $\mathbf{X}$.

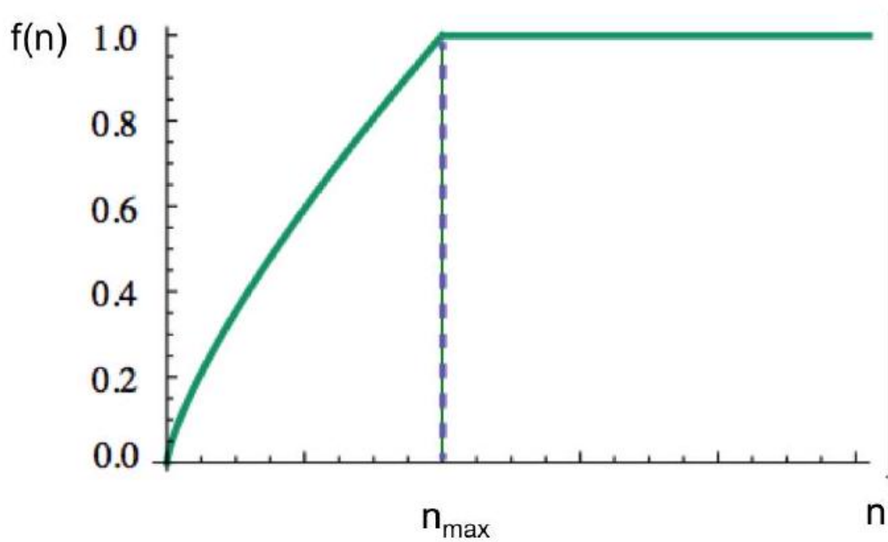
Each row of those matrices forms a representation of a word ($\mathbf{W}$) or a context word ($\mathbf{Z}$) respectively.

GloVe

This model is called GloVe, and is a variant of word2vec.

Weights f_{dn} : Give "importance" of each entry. Choosing $f_{dn} := 1$ is ok. GloVe weight function:

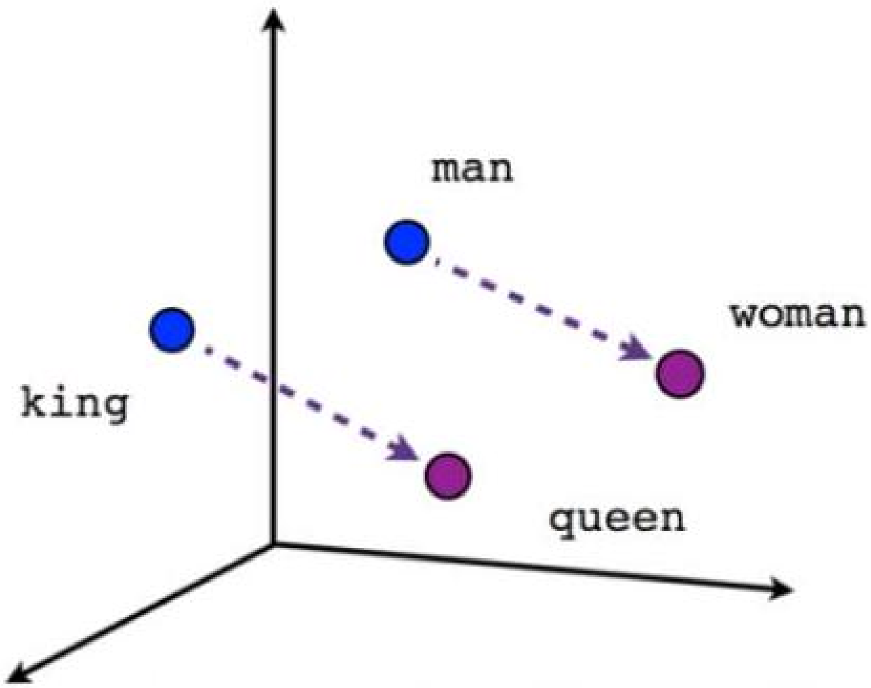
$f_{dn} := \min\left\{1, (n_{dn}/n_{\max})^{\alpha}\right\}, \quad \alpha \in [0;1] \quad$ e.g. $\alpha = \frac{3}{4}$



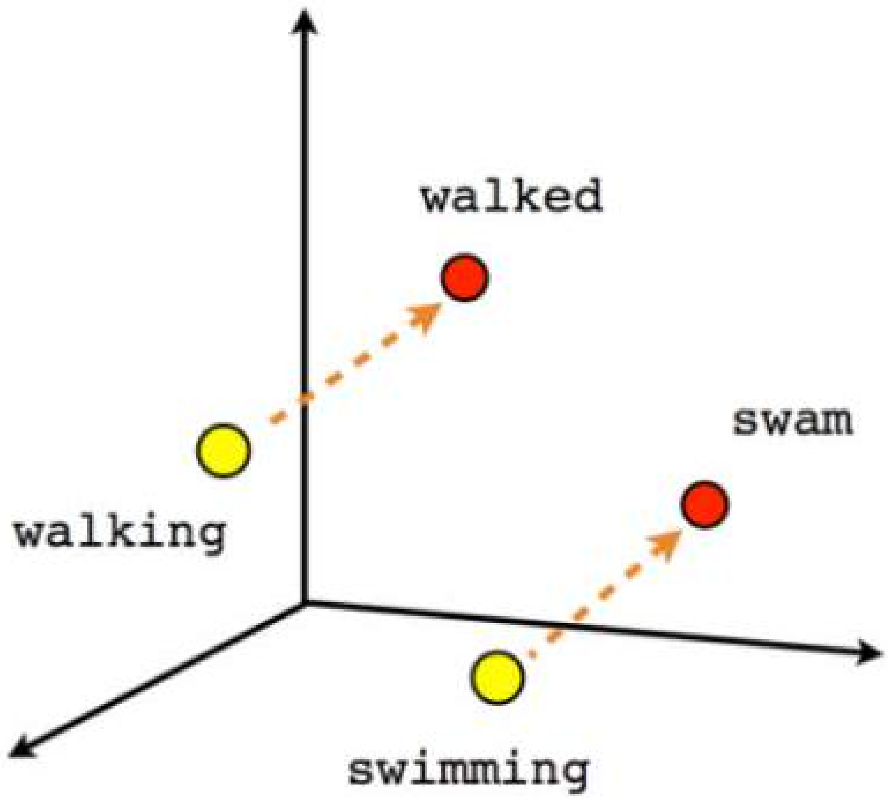
Choosing K

K e.g. 50, 100, 500

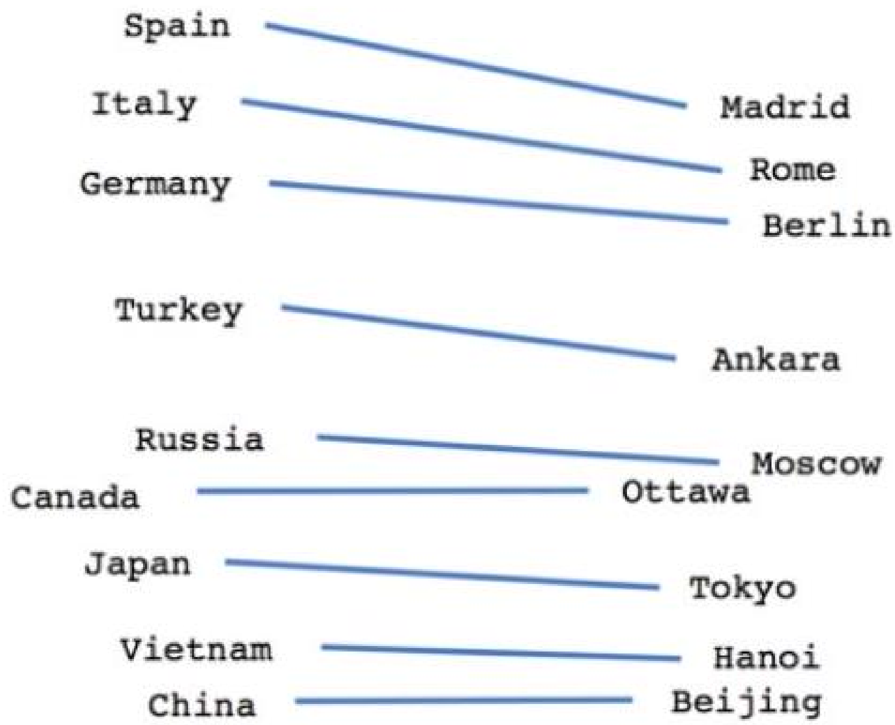
Word Analogies



Male-Female



Verb tense



Country-Capital

Training

- Stochastic Gradient Descent (SGD)
- Alternating Least-Squares (ALS)

Open questions:

- Parallel and distributed training
- Does regularization help?

Alternative: Skip-Gram Model

(Original word2vec)

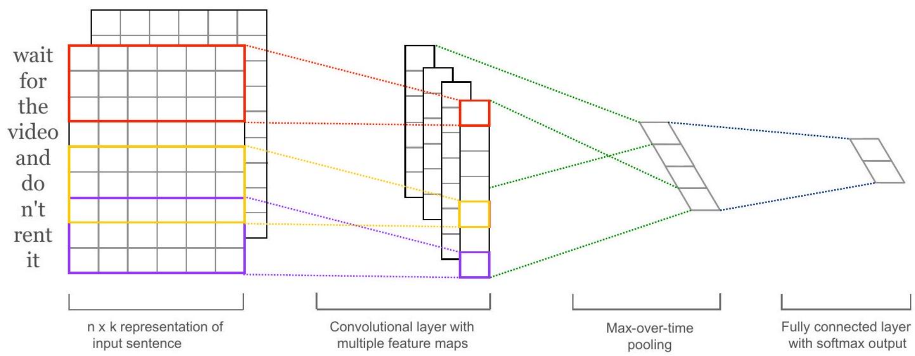
Uses binary classification (logistic regression objective), to separate real word pairs (w_d, w_n) from fake word pairs. Same inner product score = matrix factorization.

Given w_d, a context word w_n is

- real = appearing together in a context window of size 5
- fake = any word $w_{n'}$ sampled randomly: Negative sampling (also: Noise Contrastive Estimation)

Learning Representations of Sentences & Documents

Supervised: For a supervised task (e.g. predicting the emotion of a tweet), we can use matrixfactorization (below) or convolutional neural networks (see next weeks).



→ SemEval competition for tweet classification.

Unsupervised:

- Adding or averaging (fixed, given) word vectors
- Training word vectors such that adding/averaging works well
- Direct unsupervised training for sentences (appearing together with context sentences) instead of words

Fast Text

Matrix factorization to learn document/sentence representations (supervised).

Given a sentence $s_n = (w_1, w_2, \ldots, w_m)$, let $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ be the bag-of-words representation of the sentence.

$$\min_{\mathbf{W},\mathbf{Z}} \mathcal{L}(\mathbf{W},\mathbf{Z}) := \sum_{s_n \text{ a sentence}} f\left(y_n \mathbf{W}\mathbf{Z}^\top \mathbf{x}_n\right)$$

where $\mathbf{W} \in \mathbb{R}^{1\times K}, \mathbf{Z} \in \mathbb{R}^{|\mathcal{V}|\times K}$ are the variables, and the vector $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ represents our n-th training sentence.

Here f is a linear classifier loss function, and $y_n \in \{\pm 1\}$ is the classification label for sentence $\mathbf{x}_n$.

Language Models

Selfsupervised training:

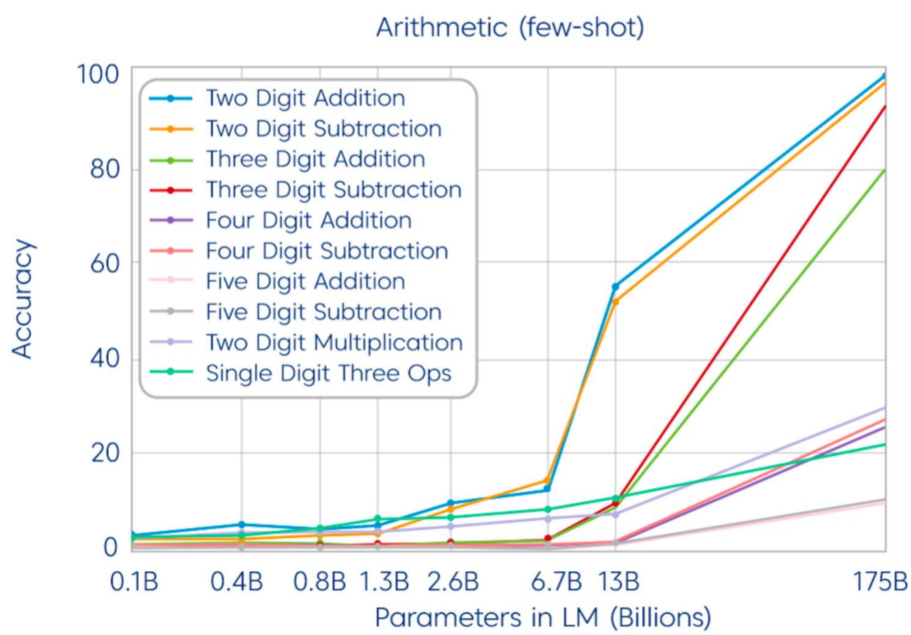
Can a model generate text? - train classifier to predict the continuation (next word) of given text

- Multi-class:

Use soft-max loss function with a large number of classes D = vocabulary size

- Binary classification:

Predict if next word is real or fake (i.e. as in word2vec)
Impressive recent progress using large models, such as transformers
(e.g. GPT-2, GPT-3, chatGPT
https://transformer huggingface.co/doc/gpt2-large,
https://chat.openai.com/)
Arithmetic:



Reasoning:

Maksym Andriushchenko @maksym_andr

I got curious about this and tested ChatGPT on last year's exam from our ML course at EPFL

(github.com/epfml/ML_course). Chain-of-thought evaluation with a majority vote over 5 trials gives 10/20

link: chatGPT on ML course exam

Further Pointers

1. word2vec:

   code: code.google.com/p/word2vec/

   paper:

   "Distributed representations of words and phrases and their compositionality" - T Mikolov, I Sutskever, K Chen, GS Corrado, J Dean. NIPS 2013

2. GloVe:

   code and vectors: nlp.stanford.edu/projects/glove/

   paper:

   "GloVe: Global Vectors for Word Representation" - Pennington, J., Socher, R., Manning, C. D.. EMNLP 2014

3. FastText & sent2vec

   code: github.com/facebookresearch/fastText

   papers:

   "Bag of Tricks for Efficient Text Classification" - Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. - EC-ACL, 2017.

   "Enriching Word Vectors with Subword Information" - Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. - TACL, 2017.

   "Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features" - Pagliardini, M., Gupta, P., Jaggi, M. NAACL 2018.

4. Write with transformers:

   code and demo: transformer.huggingface.co/doc/gpt2-large

5. ChatGPT

   demo: chat.openai.com/