Natural Language Processing with Deep Learning

Introduction and Word Vectors

Myungjun Kim

Seoul National University

January 14, 2022

Contents

What is so special about NLP?

- Human language is a system specifically constructed to convey meaning, and is not produced by a physical manifestation of any kind.
- Most words are just symbols for an extra-linguistic entity.
- Natural language is a discrete, symbolic and categorical system.

Goals and Examples of NLP

- There are different levels of tasks in NLP, from speech processing to semantic interpretation and discourse processing.
- The goal of NLP is to be able to design algorithms to allow computers to *understand* natural language in order to perform some task.
- Example tasks come in varying level of difficulty:

Easy: Spell checking, keyword search, finding synonyms

Medium: Parsing information from websites, documents, etc.

Hard: Maching translation, semantic analysis, coreference, question answering

Contents

How to represent words?

- The first and arguably most important common denominator across all NLP tasks is how we represent words as input.
- There are an estimated 13 million tokens for the English and they might not be all unrelated.
- We want to encode word tokens each into some vector that represents a point in some sort of word space.

Word Representations

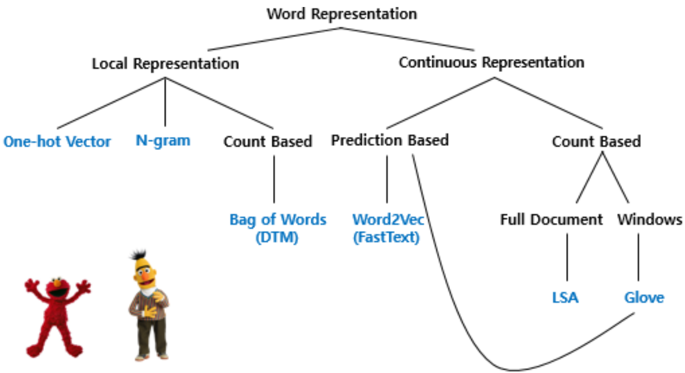

Figure 1: The examples of word representations: Local (discrete) representations represent words as symbols; Continuous (distributed) representations take advantage of the context to find the meaning of words.

Denotational Semantics

In traditional NLP, we regard words as discrete **symbols**. The most simple example is **one-hot vector**: Represent every word as an $\mathbb{R}^{|V|}$ vector with all 0s and 1 at the index of a word in the sorted English.

$$w^{feline} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$$
$$w^{cat} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$

- Vector dimension = number of words in vocabulary $|V|$.
- A one-hot vector is sparse.
- There is no natural notion of **similarity** for one-hot vectors.

$$(w^{feline})^T w^{cat} = 0$$

Distributional Semantics

- One of the most successful ideas of modern statistical NLP is **distributional semantics**: A word's meaning is given by the words that frequently appear close-by.
- When a word w appears in a text, its **context** is the set of words that appear nearby.

... government debt problems turning into banking crises as happened in 2009 ...

... saying that Europe needs unified banking regulation to replace the hodgepodge ...

... India has just given its banking system a shot in the arm ...

Context words will represent banking.

Contents

Word2vec: Overview

- **Word2vec** (Mikolov et al. 2013) is a framework for learning word vectors (word embeddings).
- Every word in a fixed vocabulary is represented by a vector.
- Go through each position t in the text, which has a center word c and context (or outside) words o.
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa).
- Keep adjusting the word vectors to maximize this probability.

Word2vec: Overview

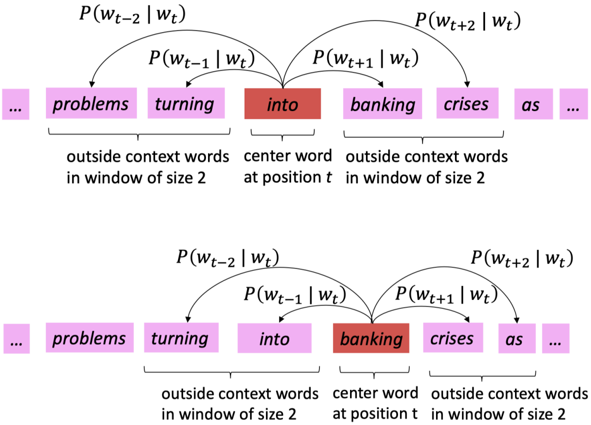

Figure 2: Example windows and process for computing $P(w_{t+j}|w_t)$

Word2vec: Objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_j. The **likelihood function** $L(\theta)$ is

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

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

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

Then

$$\hat{\theta} = \arg\max_{\theta} L(\theta) = \arg\min_{\theta} J(\theta).$$

Word2vec: Objective function

We want to minimize the objective function

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

but how to calculate $P(w_{t+j} \mid w_t;\theta)$? 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, the probability is

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

Notes

- A dot product $u_o^T v_c$ compares similarity of o and c:

$$\text{similarity} = \cos\theta = \frac{u^T v}{\|u\|\|v\|}$$

- $\sum_{w \in V} \exp(u_w^T v_c)$ normalizes over entire vocabulary to give probability distribution.
- The softmax function maps arbitrary values $x_i \in \mathbb{R}^n$ to a probability distribution $p_i \in (0,1)^n$

Word2vec: Optimization

- θ represents *all* the model parameters, in one vector:

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

- We optimize these parameters by gradient descent with respect to all vector gradients.
- Every word has two vectors; we use the average of v_w and u_w as the word vector for w.

Word2vec: Optimization

The idea of **gradient descent** is for current value of θ, calculate the gradient of $J(\theta)$ and then take small step α (learning rate) in direction of *negative gradient*. Repeat until convergence.

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

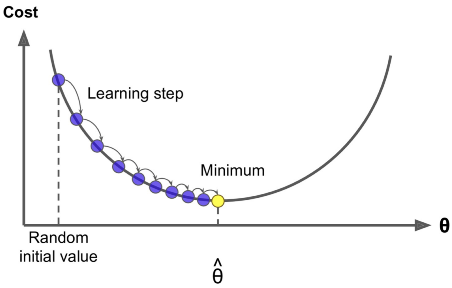

Figure 3: Illustration of gradient descent.

Word2vec: Optimization

We need to calculate partial derivative of $J(\theta)$ with respect to all word vectors. Note that

$$\begin{aligned}
\frac{\partial}{\partial v_c} \log p(o|c) &= \frac{\partial}{\partial v_c}\left(\log \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}\right) \\
&= \frac{\partial}{\partial v_c}\log\left(\exp(u_o^T v_c)\right) - \frac{\partial}{\partial v_c}\log\left(\sum_{w\in V}\exp(u_w^T v_c)\right) \\
&= u_o - \frac{\sum_{w\in V}\exp(u_w^T v_c)\cdot u_w}{\sum_{w\in V}\exp(u_w^T v_c)} \\
&= u_o - \sum_{w\in V}\frac{\exp(u_w^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}\cdot u_w \\
&= \underbrace{u_o}_{\text{observed representaion}} - \underbrace{\sum_{w\in V} p(w|c)\cdot u_w}_{\text{expected context word}}
\end{aligned}$$

Word2vec: How to train the model?

- Word2vec is a prediction-based word representation.
- There are two variants: **Continuous Bag of Words** (CBOW) and **Skip-Gram**.
- CBOW predicts a center word from context words, while skip-gram predicts context words from a center word.
- It is known that skip-gram model performs better than CBOW model.

The Skip-Gram Model

The goal is to predict the context words from the given a center word. Suppose we have a sentence: *The fat cat sat on the mat* and the size of window is 2.

The fat cat sat on the mat	[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
The fat cat sat on the mat	[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
The fat cat sat on the mat	[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
The fat cat sat on the mat	[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
The fat cat sat on the mat	[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1]
The fat cat sat on the mat	[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
The fat cat sat on the mat	[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

Figure 4: The red words are center words and the blues words are context words.

The Skip-Gram Model

1. Generate one hot input vector $x \in \mathbb{R}^{|V|}$ of the center word.
2. Get embedded word vector for the center word $v_c = \mathcal{V}x \in \mathbb{R}^N$.
3. Generate a score vector $z = \mathcal{U}v_c$.
4. Turn the score vector into probabilities,

   $$\hat{y} = \text{softmax}(z) = (\hat{y}_1, \ldots, \hat{y}_{c-m}, \ldots, \hat{y}_{c+m}, \ldots, \hat{y}_{|V|})^T,$$

   and compute loss,

   $$\mathcal{L}(\hat{y}, y) = \sum_{j=0, j\neq m}^{2m} H(\hat{y}, y_{c-m+j})$$

   where $H(\hat{y}, y)$ is the cross entropy.
5. Repeat step 1 to 4 until convergence.

The Skip-Gram Model

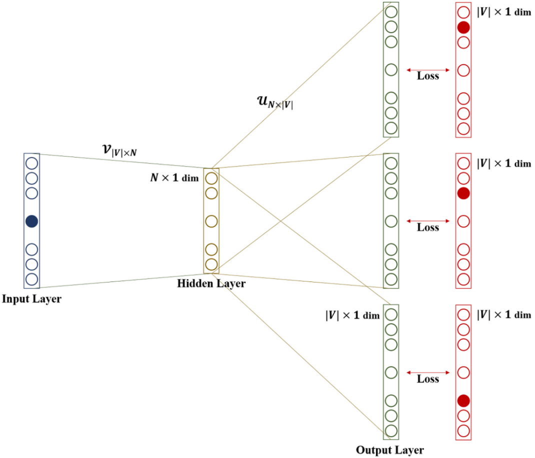

Figure 5: The skip-gram model.

Remarks

- Only one probability vector $\hat{y}$ is computed.
- Skip-gram treats each context word equally: the model computes the probability for each word of appearing in the context independently of its distance to the center word.
- $\hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m}$ are the probabilities of observing each context word.
- We desire $\hat{y}$ to match the true probabilities which is $y_{c-m}, \ldots, y_{c-1}, y_{c+1}, \ldots, y_{c+m}$, the one hot vectors of the actual output.
- The summation over $|V|$ is computationally huge. Any update we do would take $O(|V|)$ time.
- A simple idea is we could instead just approximate it, e.g., **negative sampling**.

Contents

Comparison with Previous Methods

- Count based methods (e.g. DTM, TF-IDF, LSA) effectively leverage global statistical information, but they do poorly on tasks such as word analogy.
- Prediction based methods (e.g. CBOW, skig-gram) fail to make use of the global co-occurence statistics.
- **GloVe** (Global Vectors for Word Representation, Pennington et al., 2014) uses global statistics to predict the probability of word j appearing in the context of word i with a least squares objective.

Co-occurence Matrix

- There are 2 options to build a co-occurence matrix X: window based vs. full document.
- Window based: Similar to word2vec, use window around each word.
- Full document: Give general topics leading to *Latent Semantic Analysis* (LSA).

Example: Window based Co-occurence Matrix

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

Table 1: Window based co-occurence matrix.

Co-occurence Probability

Probability and Ratio	k = solid	k = gas	k = water	k = fashion
$P(k\|$ ice $)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|$ steam $)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|$ ice $)/P(k\|$ steam $)$	8.9	8.5×10^{-2}	1.36	0.96

Table 2: Co-occurence probabilities for target words *ice* and *steam* with selected context words from a corpus.

GloVe: Notations

- X: co-occurence matrix
- X_{ij}: count of appearing word j given a center word i
- $X_i = \sum_j X_{ij}$
- $P_{ik} : P(k \mid i) = X_{ik}/X_i$
- w_i: word vector of center word i
- $\tilde{w}_k$: word vector of context word k

GloVe: Loss Function

The key idea is to make the dot product of center word and context word equal to the co-occurence probability, $w_i^T \tilde{w}_j = \log P(i \mid j)$. Suppose there is a function F satisfying

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

A function F needs to encode the ratio of co-occurence probabilities of two words to vector space:

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}},$$

or equivalently,

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

GloVe: Loss Function

We require that F be a homomorphism between the groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{++}, \times)$, i.e.,

$$F(v_1^T v_2 + v_3^T v_4) = F(v_1^T v_2)F(v_3^T v_4), \quad \forall v_1, v_2, v_3, v_4 \in V.$$

It becomes to

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}} \tag{1}$$

and then we get

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{2}$$

The solution to Eqn. (1) and (2) is $F = \exp(\cdot)$, or

$$w_i^T \tilde{w}_k = \log P_{ij} = \log X_{ik} - \log X_i. \tag{3}$$

GloVe: Loss function

Note that Eqn. (3) would exhibit the exchage symmetry if not for the $\log X_i$ on the right-hand side. This term is independent of k so it can be absorbed into a bias b_i for w_i. Finally, adding an additional bias $\tilde{b}_k$ for $\tilde{w}_k$ restores the symmetry,

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}.$$

However, the logarithm diverges whenever X_{ik} is zero, so a new weighted least squares regreesion model is proposed,

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

where $f(X_{ij}) = \min(1, (x/x_{\text{max}})^{\alpha})$ is a weighting function.

GloVe: Weighting function

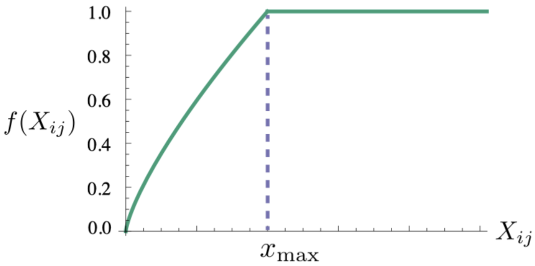

Figure 6: Weighting function f with $x_{\max} = 100$ and $\alpha = 3/4$.

GloVe: Optimization

With the cost calculated, we now need to compute gradients. From our original cost function J we derive gradients with respect to parameters $w_i, \tilde{w}_j, b_i$ and $\tilde{b}_j$. The operator $\odot$ denotes elementwise vector multiplication.

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

$$\nabla_{w_i} J = \sum_{j=1}^{V} f(X_{ij}) w_j \odot (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

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

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality, 2013.

[3] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[4] X. Rong. word2vec parameter learning explained, 2016.