Word Embedding

Ninghao Liu

University of Georgia

March 20, 2024

1. How to Represent A Word?

Treating each word as a token.

- In previous methods, each word is represented as an index. Different terms are mutually independent.
- Another perspective: Each word is a **one-hot vector** in the $|\mathcal{V}|$ dimensional space.

$$\boldsymbol{w}^{a} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \boldsymbol{w}^{at} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots, \quad \boldsymbol{w}^{zoo} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}. \tag{1}$$

- The similarity issue: Ideally we would want similar words like "learn" and "study" should be similar in the feature space.

1. How to Represent the Meaning of A Word?

Traditional NLP solution: Maintaining synonym or hypernyms sets, e.g., WordNet.

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

1. How to Represent the Meaning of A Word?

Issues with WordNet-like solution.

- Missing nuance.
  - E.g., “proficient” is listed as a synonym for “good”.
  - This is only correct in certain contexts.
- Missing new meanings of words.
- Requires human labor to create and maintain.
  - Subjective.
- Cannot compute word similarity.
  - "good", "great", "proficient", ...

2. Word Embeddings

- We still assign a **vector** to each word.
- The vector is **dense**, but not one-hot.

$$
\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}
$$

- The word vectors are also called word **embeddings** or (neural) word **representations**.

2. Word Embeddings

- **Word embeddings** – towards dense, semantically-meaningful representation of words.

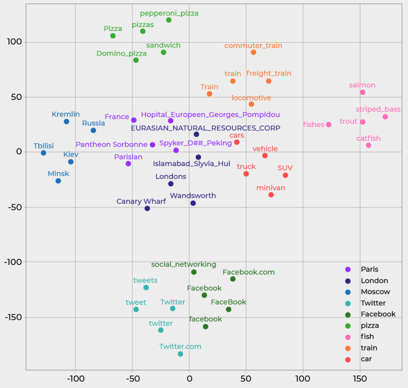


Figure 1: Visualization of embeddings of words and phrases [1].

[1] https://openclassrooms.com/en/courses/6532301-introduction-to-natural-language-processing/

2. Word Embeddings

Why are they called “embeddings”?
“embed”: enroot, implant, ...

2. Word Embeddings

The overall idea to compute the embeddings of words.

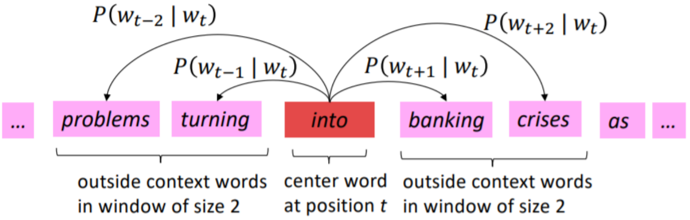
- A word's meaning is decided by the words that frequently appear close-by.
  - "You shall know a word by the company it keeps" (J. R. Firth 1957: 11).
  - One of the most successful ideas of modern statistical NLP!
  - Representing words by their context.
- Given a word w, its context is the set of its nearby words (within a fixed-size window).

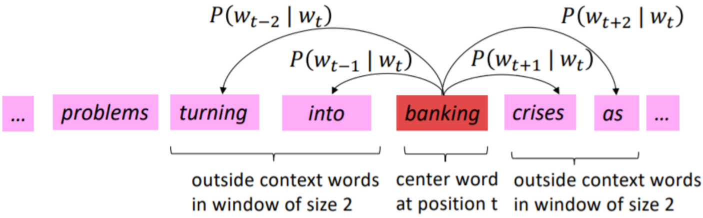
3. Word2Vec

Word2vec (Mikolov et al. 2013) is a computational framework for learning word embeddings.

- Input: A large corpus of texts.
- Output: Embeddings or word vectors. 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 ("outside") words o.
- Calculate the probability of o given c (or vice versa).
- Adjust the vectors to maximize this probability.

3. Word2Vec


$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2


$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2

3. Word2Vec

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

The likelihood of observing the given text:

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

θ is all variables to be optimized

The objective function (cost function) is:

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

We want to minimize the objective function.

3. Word2Vec

We want to minimize the cost function:

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

Question: How to calculate $P(w_{t+j}|w_t;\theta)$?

Solution: We assign a vector $\boldsymbol{v}_w$ to each word w.

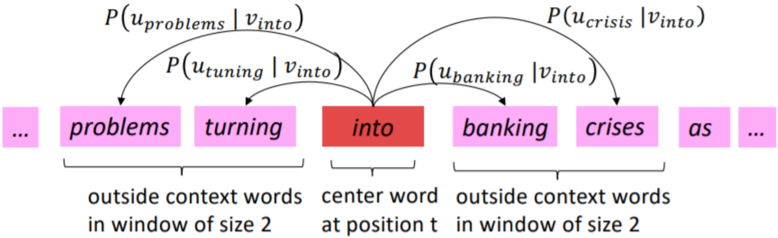
Given a center word c and a context word o,

$$P(o|c) = \frac{\exp\left(\boldsymbol{v}_o^\intercal \boldsymbol{v}_c\right)}{\sum_{w\in\mathcal{V}} \exp\left(\boldsymbol{v}_w^\intercal \boldsymbol{v}_c\right)}\,. \tag{2}$$

3. Word2Vec

Example of computing $P(w_{t+j}|w_t;\theta)$

- $P(turning|into;\theta) = P(\mathbf{v}_{turning}|\mathbf{v}_{into})$



3. Word2Vec

A closer look at

$$P(o|c) = \frac{\exp\left(\boldsymbol{v}_o^\intercal \boldsymbol{v}_c\right)}{\sum_{w \in \mathcal{V}} \exp\left(\boldsymbol{v}_w^\intercal \boldsymbol{v}_c\right)}\,. \tag{3}$$

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

$$softmax(x_i) = \frac{\exp\left(x_i\right)}{\sum_{j=1}^{n} \exp\left(x_j\right)} = p_i \tag{4}$$

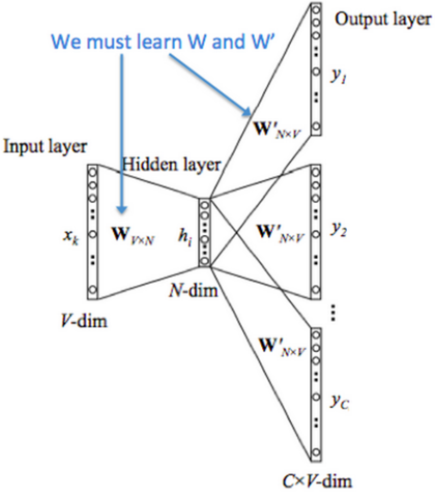
- Maps arbitrary values x_i to a probability distribution p_i .
    - "max": it amplifies the probability of largest x_i;
    - "soft": it still assigns some probability to smaller x_i.
    - A widely used technique in deep learning.

3. Word2Vec

Why is it a neural network?

3. Word2Vec

Why is it a neural network?



3. Word2Vec

Another look at

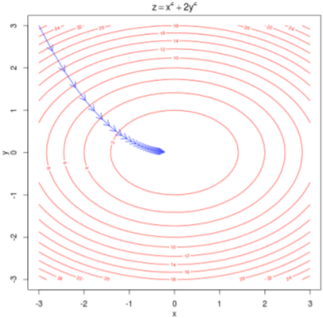
$$P(o|c) = \frac{\exp\left(\boldsymbol{v}_o^\intercal \boldsymbol{v}_c\right)}{\sum_{w\in\mathcal{V}} \exp\left(\boldsymbol{v}_w^\intercal \boldsymbol{v}_c\right)} . \tag{5}$$

- Computation cost?
- Solution: Negative sampling.

3. Word2Vec

- How to train our model?
  - Gradient descent.
  - Gradually adjust parameters, by walking down the gradient, to minimize the loss.
- What are the parameters θ?
  - Embedding vectors.
  - Each word owns two vectors.

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



3. Word2Vec

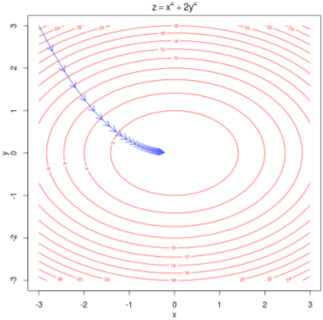
Stochastic Gradient Descent

- $J(\theta)$ considers all windows (potentially billions) in the corpus.

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

- Each parameter update is expensive.
- Solution: Stochastic gradient descent (SGD)
    - Repeatedly sample windows, and update part of the parameters for each mini-batch of windows.
    - Will it converge?

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



In SGD, for each mini-batch of data, the parameters update "somehow" moves towards the solution.

4. The "Real" Word2Vec

We want to minimize the cost function:

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

When calculating $P(w_{t+j}|w_t;\theta)$, we assign two vectors to each word.

- $\boldsymbol{v}_w$, when w is the center word,
- $\boldsymbol{u}_w$, when w is the context word.

Given a center word c and a context word o,

$$P(o|c) = \frac{\exp\left(\boldsymbol{u}_o^\top \boldsymbol{v}_c\right)}{\sum_{w\in\mathcal{V}} \exp\left(\boldsymbol{u}_w^\top \boldsymbol{v}_c\right)}. \tag{6}$$

Q: Which embedding to be used in downstream applications?

5. How to evaluate word embeddings?

- Intrinsic evaluation.
  - Evaluate on a specific/intermediate subtask.
  - Helps to understand the embeddings.
  - Limitations?
- Extrinsic evaluation.
  - Evaluate on a real task.
  - Text classification, QA, ...
  - Limitations?

5. Intrinsic evaluation

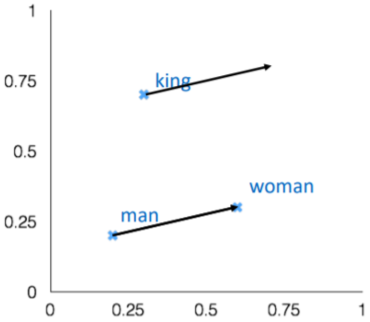
- Word Vector Analogies

a:b :: c:?

man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



5. Intrinsic evaluation