CS 6501 Natural Language Processing

Letent Semantic Analysis

Yangfeng Ji

October 22, 2019

Department of Computer Science
University of Virginia

UNIVERSITY of VIRGINIA | ENGINEERING

Overview

1. Distributional Hypothesis
2. Latent Semantic Analysis
3. Word Embeddings

Distributional Hypothesis

Distributional Hypothesis

Distributional hypothesis

Words that occur in the similar contexts tend to have similar meanings

[Jurafsky and Martin, 2019, Chap 06]

Distributional Hypothesis

Distributional hypothesis

Words that occur in the similar contexts tend to have similar meanings

Examples

- `to have a splendid time in Rome`
- `to have a wonderful time in Rome`

[Jurafsky and Martin, 2019, Chap 06]

Another Example

Another example

- `_____ is delicious sauteed with garlic.`
- `_____ is superb over rice.`
- `... _____ leaves with salty sauces ...`

[Jurafsky and Martin, 2019]

Generalized Hypotheses

- **Statistical semantics hypothesis**: Statistical patterns of human word usage can be used to figure out what people mean.

[Turney and Pantel, 2010]

Generalized Hypotheses

- **Statistical semantics hypothesis**: Statistical patterns of human word usage can be used to figure out what people mean.
- **Bag of words hypothesis**: The frequencies of words in a document tend to indicate the relevance of the document to a query

[Turney and Pantel, 2010]

Latent Semantic Analysis

Word-document Matrix

For a corpus of d documents over a vocabulary $\mathcal{V}$, the cooccurence matrix is defined as $\mathbf{C}$,

$$\begin{aligned} \mathbf{C} &= [c_{ij}] \in \mathbb{R}^{v \times d} \\ &= \begin{bmatrix} c_{1,1} & \dots & c_{1,d} \\ \vdots & \ddots & \vdots \\ c_{v,1} & \dots & c_{v,d} \end{bmatrix} \end{aligned} \tag{1}$$

where

- $v = |\mathcal{V}|$ is the size of vocab, and
- c_{ij} is the count of word i in document j

Word-document Matrix

Word	Documents							
	1	2	3	4	5	6	7	8
w_1	0	1	0	0	0	0	0	0
w_2	0	0	1	0	0	3	0	0
w_3	1	0	0	2	0	0	5	0
w_4	3	0	0	1	1	0	2	0
w_5	0	1	3	0	1	2	1	0
w_6	1	2	0	0	0	0	1	0
w_7	0	1	0	1	0	1	0	1
w_8	0	0	0	0	0	7	0	0

Also called *term-document* matrix

Two Views of Word-Document Matrix

- Each row $\boldsymbol{d}_i$ is a document (BoW) representation
- Each column $\boldsymbol{w}_k$ is a word representation

Word	Documents							
	1	2	3	4	5	6	7	8
w_1	0	1	0	0	0	0	0	0
w_2	0	0	1	0	0	3	0	0
w_3	1	0	0	2	0	0	5	0
w_4	3	0	0	1	1	0	2	0
w_5	0	1	3	0	1	2	1	0
w_6	1	2	0	0	0	0	1	0
w_7	0	1	0	1	0	1	0	1
w_8	0	0	0	0	0	7	0	0

Similarity

If we consider a document as a context, we can use row vectors $\{\boldsymbol{w}_k\}$ to represent words and hence measure similarity between words as

$$\text{cos-sim}(\boldsymbol{w}_k, \boldsymbol{w}_{k'}) = \frac{\boldsymbol{w}_k^\top \boldsymbol{w}_{k'}}{\|\boldsymbol{w}_k\|_2 \cdot \|\boldsymbol{w}_{k'}\|_2} \tag{2}$$

- $\boldsymbol{w}_k^\top \boldsymbol{w}_{k'} = \sum_{i=1} w_{k,i} w_{k',i}$
- $\|\boldsymbol{w}_k\|_2 = \sqrt{\sum_{i=1} w_{k,i}^2}$

Data Sparsity

Compute the dot product of the following two pairs

- $w_1^\top w_2$
- $w_2^\top w_3$

Word	Documents							
	1	2	3	4	5	6	7	8
w_1	0	1	0	0	0	0	0	0
w_2	0	0	1	0	0	3	0	0
w_3	1	0	0	2	0	0	5	0
w_4	3	0	0	1	1	0	2	0
w_5	0	1	3	0	1	2	1	0
w_6	1	2	0	0	0	0	1	0
w_7	0	1	0	1	0	1	0	1
w_8	0	0	0	0	0	7	0	0

- It will get even worse when we have a large vocab, say, 10K or 50K words

Singular Value Decomposition (SVD)

Using SVD, the word-document matrix $\mathbf{C}$ is decomposed into a multiplication of three matrices

$$\mathbf{C} = \mathbf{U}_0 \cdot \boldsymbol{\Sigma}_0 \cdot \mathbf{V}_0^\top. \tag{3}$$

- $\mathbf{U}_0 \in \mathbb{R}^{v \times v}$ is orthonormal
- $\mathbf{V}_0 \in \mathbb{R}^{d \times d}$ is orthonormal
- $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{v \times d}$ is diagonal — each component on the diagonal is called a singular value

Singular Values

A real example

- $\mathbf{C} \in \mathbb{R}^{40K \times 20K}$: from the Yalp dataset used in homework 01

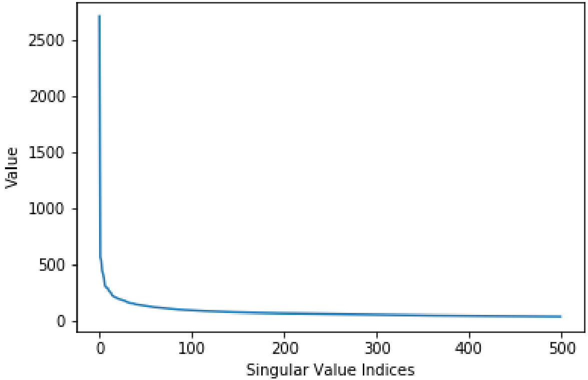


Figure: Plot of the (first 500) singular values in decreasing order.

SVD

Approximate $\mathbf{C}$ with a lower-dimensional factorization

$$\mathbf{C} \approx \underbrace{\begin{bmatrix} | & & | \\ \boldsymbol{u}_1 & \dots & \boldsymbol{u}_k \\ | & & | \end{bmatrix}}_{\mathbf{U}} \cdot \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix}}_{\boldsymbol{\Sigma}} \cdot \underbrace{\begin{bmatrix} — & \boldsymbol{v}_1 & — \\ & \vdots & \\ — & \boldsymbol{v}_k & — \end{bmatrix}}_{\mathbf{V}^\top} \tag{4}$$

where $\mathbf{U} \in \mathbb{R}^{v \times k}$, $\mathbf{V} \in \mathbb{R}^{d \times k}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}$.

Lower-dimensional Word/Doc Representations

Given

$$\mathbf{C} \approx \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \tag{5}$$

Based on 5, the word representation can be constructed as

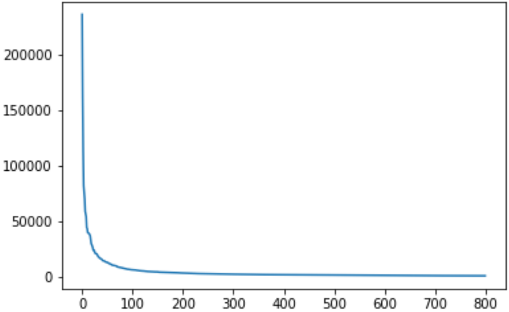
$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma} \in \mathcal{R}^{v \times k} \tag{6}$$

and document representation as

$$\mathbf{D} = \mathbf{V}\boldsymbol{\Sigma} \in \mathcal{R}^{d \times k} \tag{7}$$

Re-weighting: Motivation

Word frequency in the descreasing order



Top words: `the`, `and`, `to`, `was`, `it`

Re-weighting: TF-IDF

- Term frequency $\mathrm{tf}_{w,d}$: the frequency of the word w in the document d

$$\mathrm{tf}_{w,d} = \#(w, d) \tag{8}$$

Re-weighting: TF-IDF

- Term frequency $\mathrm{tf}_{w,d}$: the frequency of the word w in the document d

$$\mathrm{tf}_{w,d} = \#(w, d) \tag{8}$$

- Document frequency df_w: the number of documents that the word w occurs in
- Inverse document frequency

$$\mathrm{idf}_w = \log_{10} \frac{N}{\mathrm{df}_w} \tag{9}$$

where N is the total number of documents

Re-weighting: TF-IDF

- Term frequency $\mathrm{tf}_{w,d}$: the frequency of the word w in the document d

$$\mathrm{tf}_{w,d} = \#(w, d) \tag{8}$$

- Document frequency df_w: the number of documents that the word w occurs in
- Inverse document frequency

$$\mathrm{idf}_w = \log_{10} \frac{N}{\mathrm{df}_w} \tag{9}$$

  where N is the total number of documents
- TF-IDF weighted value: for word w in document d, the corresponding value in the matrix $\mathbf{C}$ is

$$c_{w,d} = \mathrm{tf}_{w,d} \cdot \mathrm{idf}_w \tag{10}$$

Re-weighting: TF-IDF

- Term frequency $\mathrm{tf}_{w,d}$: the frequency of the word w in the document d

$$\mathrm{tf}_{w,d} = \#(w, d) \tag{8}$$

- Document frequency df_w: the number of documents that the word w occurs in
- Inverse document frequency

$$\mathrm{idf}_w = \log_{10} \frac{N}{\mathrm{df}_w} \tag{9}$$

  where N is the total number of documents
- TF-IDF weighted value: for word w in document d, the corresponding value in the matrix $\mathbf{C}$ is

$$c_{w,d} = \mathrm{tf}_{w,d} \cdot \mathrm{idf}_w \tag{10}$$

- Factorize the weighted matrix using SVD

Context Window Size

Distributional hypothesis

Words that occur in the similar contexts tend to have similar meanings

Word	Documents							
	1	2	3	4	5	6	7	8
w_1	0	1	0	0	0	0	0	0
w_2	0	0	1	0	0	3	0	0
w_3	1	0	0	2	0	0	5	0
w_4	3	0	0	1	1	0	2	0
w_5	0	1	3	0	1	2	1	0
w_6	1	2	0	0	0	0	1	0
w_7	0	1	0	1	0	1	0	1
w_8	0	0	0	0	0	7	0	0

Are w_i and w_j similar to each other, when they appear in the same documents but far away from each other?

Context Window Size (II)

Just under a week ago, Apple released a supplemental update to macOS Catalina with various bug fixes and performance improvements. Now, Apple has made a revised version of that same supplemental update available to users.

On its developer website, Apple says that a new version of the macOS Catalina supplemental update has been released today. If you installed the original supplemental update released last week, you might not even receive today's revised version with Apple focusing on people who hadn't yet installed the initial supplemental update.

The release notes for today's update, build 19A603, are exactly the same as last week's:

- Improves installation reliability of macOS Catalina on Macs with low disk space
- Fixes an issue that prevented Setup Assistant from completing during some installations
- Resolves an issue that prevents accepting iCloud Terms and Conditions when multiple iCloud accounts are logged in
- Improves the reliability of saving Game Center data when playing Apple Arcade games offline

The revised version of the macOS Catalina supplemental update likely includes very minor changes and fixes. Apple is also currently beta testing macOS Catalina 10.15.1, which may have provided our first look at the forthcoming 16-inch MacBook Pro.

Word Embeddings

Skip-gram

One way of finding a better word representation is to make sure it has the potential to predict its surrounding words

$$P(w_{t+i} \mid w_t; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t})} \tag{11}$$

where $i \in \{-c, \ldots, -1, 1, \ldots, c\}$ and c is the window size.

Skip-gram

One way of finding a better word representation is to make sure it has the potential to predict its surrounding words

$$P(w_{t+i} \mid w_t; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t})} \tag{11}$$

where $i \in \{-c, \ldots, -1, 1, \ldots, c\}$ and c is the window size.

- $t = 6, c = 2$

Skip-gram

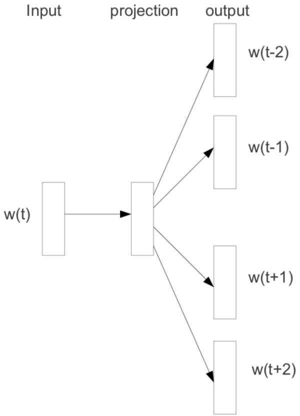
One way of finding a better word representation is to make sure it has the potential to predict its surrounding words

$$P(w_{t+i} \mid w_t; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t})} \tag{11}$$

where $i \in \{-c, \ldots, -1, 1, \ldots, c\}$ and c is the window size.

- $t = 6, c = 2$
- Usually, larger window size c gives better quality of word representations, but it also causes large computational complexity.

The Skip-gram Model



[Mikolov et al., 2013]

Word Vectors vs. Context Vectors

Distinguish a word as target (input) and context (output):

$$p(w_{t+i} \mid w_t; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{u}_{w_{t+i}}^{\top} \boldsymbol{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^{\top} \boldsymbol{v}_{w_t})} \tag{12}$$

- $\boldsymbol{v}_w$: word vector (as input)
- $\boldsymbol{u}_w$: context vector (as output)

Word Vectors vs. Context Vectors

Distinguish a word as target (input) and context (output):

$$p(w_{t+i} \mid w_t; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t})} \tag{12}$$

- $\boldsymbol{v}_w$: word vector (as input)
- $\boldsymbol{u}_w$: context vector (as output)

Question

Why we need two vectors for a word?

Objective Function

The objective function of a skip-gram model is defined as

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq i\leq c; i\neq 0}\log p(w_{t+i}\mid w_t) \tag{13}$$

- $\log p(w_{t+i}\mid w_t) = \boldsymbol{u}_{w_{t+i}}^{\top}\boldsymbol{v}_{w_t} - \log\sum_{w'\in\mathcal{V}}\exp(\boldsymbol{u}_{w'}^{\top}\boldsymbol{v}_{w_t})$
- In practice, the vocab size could be 10K, 50K or even bigger, the computation of the log-sum-exp is prohibitively expensive

Negative Sampling

Replace

$$\log p(w_{t+i} \mid w_t) = \boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t} - \log \sum_{w' \in \mathcal{V}} \exp(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t})$$

with the following function as objective

$$\log \sigma(\boldsymbol{u}_{w_{t+i}}^\top \boldsymbol{v}_{w_t}) - \sum_{i=1}^{k} E_{w' \sim p_n(w)} \Big[\log \sigma(\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{w_t}) \Big] \tag{14}$$

where k is the number of negative samples

Basic Training Procedure

Example with $t = 6$, $i = 1$, and $k = 3$

```
... finding a better word representation ...
```

w_6	w_7	negative samples
better	word	larger
		cause
		window

Basic Training Procedure

Example with $t = 6$, $i = 1$, and $k = 3$

```
... finding a better word representation ...
```

w_6	w_7	negative samples
better	word	larger
		cause
		window

For a given word w_t and i

1. Treat its neighboring context word w_{t+i} as positive example
2. Randomly sample k other words from the vocab as negative examples
3. Optimize Equation 14 to update both $\boldsymbol{v}_{\cdot}$ and $\boldsymbol{u}_{\cdot}$

Two Factors in Negative Sampling

$$\log \sigma(\boldsymbol{u}_{w_{t+i}}^{\top} \boldsymbol{v}_{w_t}) - \sum_{i=1}^{k} E_{w' \sim p_n(w)} \Big[\log \sigma(\boldsymbol{u}_{w'}^{\top} \boldsymbol{v}_{w_t}) \Big] \tag{15}$$

Two Factors in Negative Sampling

$$\log \sigma(\boldsymbol{u}_{w_{t+i}}^{\top} \boldsymbol{v}_{w_t}) - \sum_{i=1}^{k} E_{w' \sim p_n(w)} \Big[\log \sigma(\boldsymbol{u}_{w'}^{\top} \boldsymbol{v}_{w_t}) \Big] \tag{15}$$

Two factors [Mikolov et al., 2013]

- $k = ?$
  - $5 \leq k \leq 20$ works better for small datasets
  - $2 \leq k \leq 5$ is enough for large datasets

Two Factors in Negative Sampling

$$\log \sigma(\boldsymbol{u}_{w_{t+i}}^{\top} \boldsymbol{v}_{w_t}) - \sum_{i=1}^{k} E_{w' \sim p_n(w)} \Big[\log \sigma(\boldsymbol{u}_{w'}^{\top} \boldsymbol{v}_{w_t}) \Big] \tag{15}$$

Two factors [Mikolov et al., 2013]

- $k = ?$
  - $5 \leq k \leq 20$ works better for small datasets
  - $2 \leq k \leq 5$ is enough for large datasets
- Noisy distribution $p_n(w)$
  - $p_n(w) \propto \text{unigram-distribution}(w)^{\frac{3}{4}}$

Examples: Words and their Neighbors

The same Yalp dataset, with $k = 50$

yummy	horrible
delicious	terrible
tasty	poor
delish	awful
yum	*customer*
incredible	exceptional
superb	bad
phenomenal	astonished
fantastic	*pleasant*
disappoint	*happier*
awesome	*zero*

Summary

1. Distributional Hypothesis
2. Latent Semantic Analysis
3. Word Embeddings

Reference

Jurafsky, D. and Martin, J. (2019).
Speech and language processing.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013).
Distributed representations of words and phrases and their compositionality.
In *Advances in neural information processing systems*, pages 3111–3119.

Turney, P. D. and Pantel, P. (2010).
From frequency to meaning: Vector space models of semantics.
Journal of artificial intelligence research, 37:141–188.