Hearst Patterns and Distributional Semantics

COMP-550

Oct 24, 2017

Reminder

Project proposal is due soon! Please submit on MyCourses.

Outline

Detecting lexical semantic relationships

Distributional semantics

- Count-based methods
- Singular-value decomposition
- word2vec

Assignment 1 feedback

Last Class

Word sense disambiguation

Another lexical semantic task: detecting words that are in a certain lexical semantic relation

e.g., a rabbit is a mammal

Hearst Patterns (1992)

Pairs of terms that are in hyponym-hypernym relationships tend to occur in certain **lexico-syntactic** patterns:

> *The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck for each string.*
>
> (Hearst, 1992)

What are the hyponym and hypernym in this passage?

Hearst's Original Patterns

NP such as {NP}* {and|or} NP

such NP as {NP ,}* {or|and} NP

NP {, NP}* {,} or other NP

NP {, NP}* {,} and other NP

NP {,} including {NP, }* {or|and} NP

NP {,} especially {NP ,}* {or|and} NP

Exercise: label each NP as indicating hyponym or hypernym

How To Find Patterns?

Be smart and just think of them?

Hint: Think about our idea of bootstrapping that we saw from last class

Other Relations

Using this approach, Hearst patterns have also been discovered and used for other relations between words, e.g., cause-effect relationships (Girju, 2002)

- e.g., *Earthquakes* ***cause*** *tidal waves*.
- NP-cause cause NP-effect

Other verbs:

- *induce, give rise (to), stem (from), etc.*

Synonymy

We've looked at the relationship between two words that co-occur, and their intervening words.

> ***Extinct birds****, such as* ***dodos****,* ***moas****, and* ***elephant birds***

What if the words don't tend to co-occur directly?

> e.g., synonyms are supposed to be substitutes of each other
>
> *The dodo* ***went extinct*** *in the 17th century.*
>
> *The dodo* ***died out*** *in the 17th century.*

Another signal: the words that tend to co-occur with the target words

Distributional Semantics

You shall know a word by the company it keeps.

Firth, 1957

Understand a term by the distribution of words that appear near the term

Basic Idea

Go through a corpus of text. For each word, keep a count of all of the words that appear in its context within a window of, say, 5 words.

John Firth was an English linguist and a leading figure in British linguistics during the 1950s.

Term-Context Matrix

Each row is a vector representation of a word

	the	was	and	British	linguist
Firth	5	7	12	6	9
figure	276	87	342	56	2
linguist	153	1	42	5	34
1950s	12	32	1	34	0
English	15	34	9	5	21

Context words

Co-occurrence counts

Target words

Cosine Similarity

Compare word vectors A and B by

$$\text{sim}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

This corresponds to the cosine of the angle between the two vectors.

Range of values:

- -1 Vectors point in opposite directions
- 0 Vectors are orthogonal
- 1 Vectors point in the same direction

If vectors are positive (e.g., they're count vectors), similarity score is between 0 and 1.

Reasons Words Can Be Related

Cosine similarity gives you a lot more than synonymy!

Any words that tend to share context words will have high cosine similarity. What are some reasons for this?

- Synonymy or near-synonymy
- others?

Similarity vs. Relatedness

Similarity:

- Specifically about synonymy, hypernymy, hyponymy
- e.g., *chair* is similar to *furniture*
- *cat* is not similar to *scratching post*

Relatedness:

- Includes anything that might be associated
- *good* is related to *bad* (antonyms mess things up!)

Confusingly, people often say similarity when they mean relatedness. e.g., what is cosine similarity a measure of?

Vector Space Evaluation

Word vectors have no objective inherent value

- Is the vector [0.4, 0.3, -0.2] better for the word *linguistics*, or [0.2, 0.5, 0.1]?
- Evaluate the similarity of vectors to each other instead
- Correlate against some gold standard. Many possible choices: http://wordvectors.org/suite.php

e.g., the WS-353 data set (Finkelstein et al., 2002)

monk	oracle	5
cemetery	woodland	2.08
food	rooster	4.42
coast	hill	4.38
forest	graveyard	1.85
shore	woodland	3.08
monk	slave	0.92

Constructing Better Word Vectors

Rescaling with PMI weighting

Singular value decomposition

Learning word vectors with neural networks: word2vec

Rescaling the Vectors

Instead of raw counts, people usually use a measure of how much two words are correlated with each other, above chance.

Pointwise mutual information (PMI)

$$\text{pmi}(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- Numerator: probability of both words occurring (i.e., in each other's context)
- Denominator: probability of each word occurring in general

Pointwise Mutual Information Example

the occurs 100,000 times in a corpus with 1,000,000 tokens, of which it co-occurs with *linguistics* 300 times. *linguistics* occurs 2,500 times in total.

$$P(the, linguistics) = 0.0003$$
$$P(the) = 0.1$$
$$P(linguistics) = 0.0025$$

$$\text{pmi}(the, linguistics) = \log \frac{0.0003}{0.00025} = 0.26303 \text{ (base 2)}$$

If ratio is < 1, PMI is negative

People often discard negative values → positive pointwise mutual information (PPMI)

Sparsity in Term-Context Matrices

Term-context matrices are **sparse** (many zeros)

Can we compress the sparse matrix into some **dense** representation?

- Dense matrix allows small number of dimensions which are all well used; easier to use as features in downstream applications!
- Number of dimensions can be pre-specified, and does not grow with the number of context words

Singular Value Decomposition (SVD)

SVD: a matrix factorization algorithm

- Let's apply it to our term-context matrix, X

$$X = W \times \Sigma \times C^T$$

$|V| \times c \quad |V| \times m \quad m \times m \quad m \times c$

- m is the **rank** of matrix X
- Rows of W are the new word vectors
- Rows of C (columns of C^T) are the new context vectors
- Σ is a diagonal matrix of the **singular values** of X (the square root of the **eigenvalues** of $X^T X$, arranged from highest to lowest)

Truncated SVD

Idea: throw out some of the singular values in Σ

Latent semantic analysis

- Apply SVD to compress the term-context matrix while minimizing reconstruction loss
- Removes noise and prevents overfitting of model

$$\underset{|V| \text{ x } c}{X_k} \cong \underset{|V| \text{ x } k}{W_k} \times \underset{k \text{ x } k}{\Sigma_k} \times \underset{k \text{ x } c}{C_k^T} \quad , k < m$$

- Use rows of $W_k \times \Sigma_k$ as new word representations

Views of Truncated SVD

It can be shown that for any matrix B of at most rank k,

$$\|X - X_k\|_2 \leq \|X - B\|_2$$

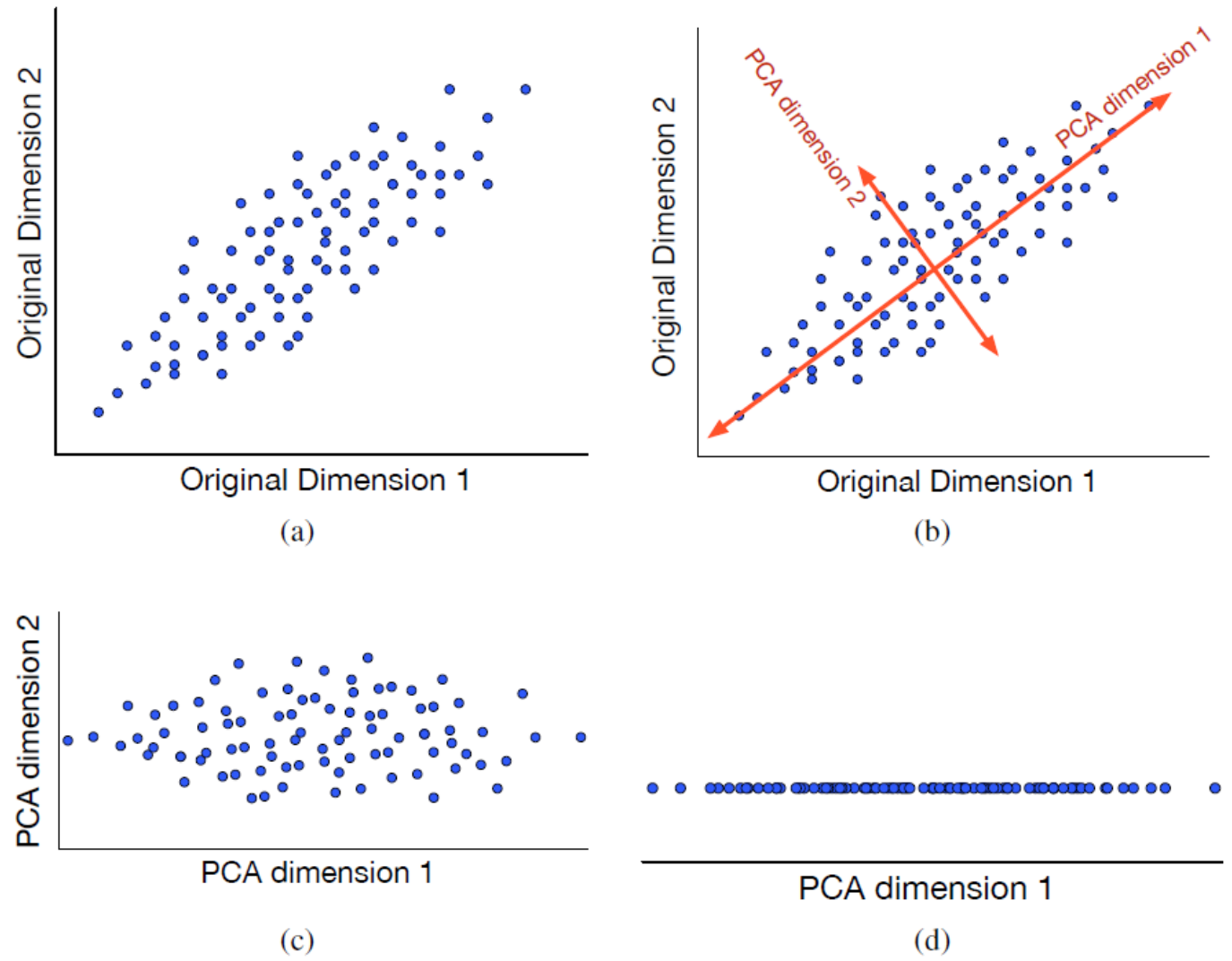
- i.e., X_k is the best possible approximation among any matrix of this rank, according to squared error.

Truncated SVD also corresponds to

1. finding the principal components of the data, then
2. projecting down to the lower-dimensional subspace spanned by these principal components

(**Principal component analysis**)

Principal Components Graphically



J&M 3rd ed., Figure 16.1

Word Embeddings

Neural network models – train vector space representation of word to predict words in context

- e.g., word2vec (Mikolov et al., 2013)
- These vector representations of words are called **word embeddings**

- We have a vector of parameters for each word j as a target word v_j, and another vector of parameters for each word as a context word c_j.
- Learn all the v_js and c_js using some auxiliary task

word2vec (Mikolov et al., 2013)

Learn vector representations of words

Actually two models:

- Continuous bag of words (CBOW) – use context words to predict a target word
- Skip-gram – use target word to predict context words

In both cases, the representation that is associated with the target word is the embedding that is learned.

word2vec Architectures

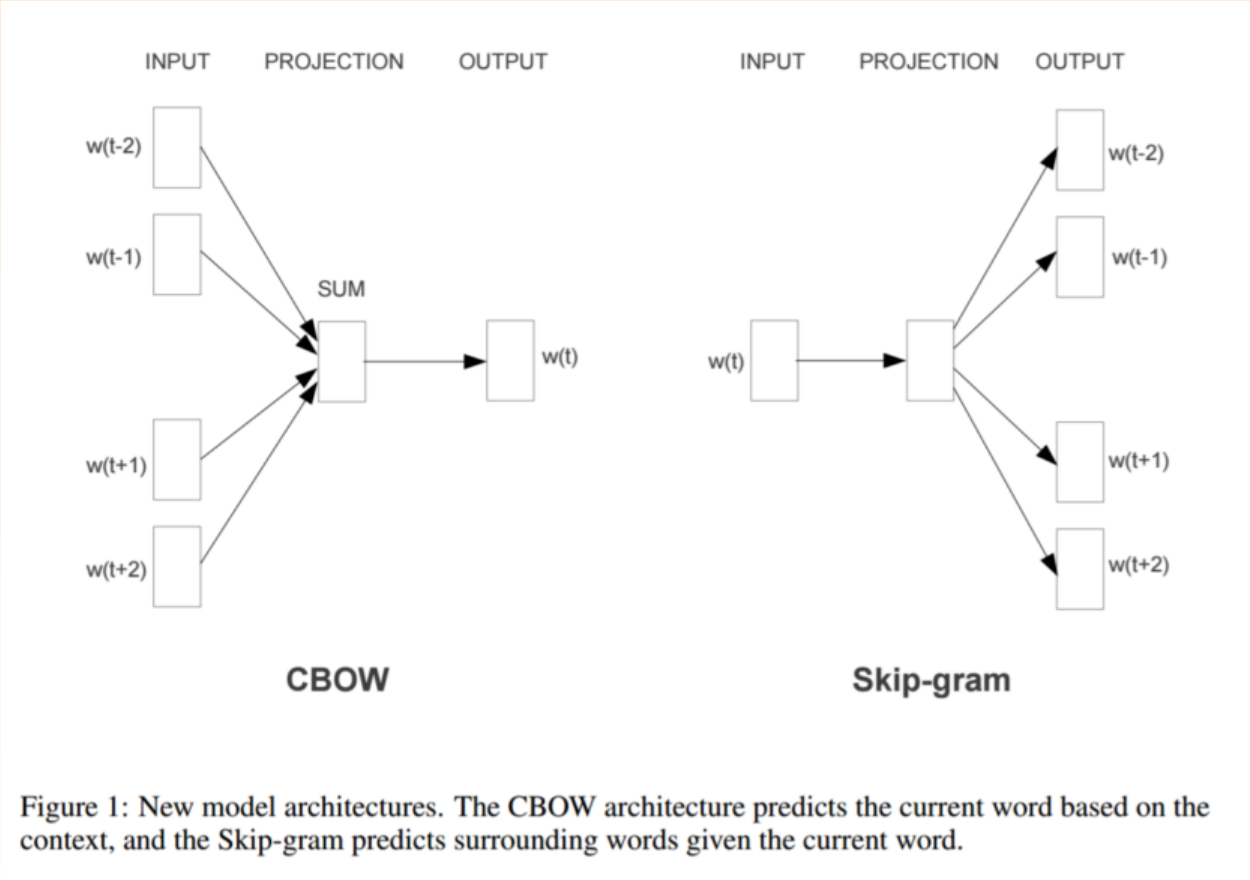


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

(Mikolov et al., 2013)

Ideal Skip-gram Objective

What we would like to learn:

- target word vectors v_j
- context word vectors c_k
- s.t.

$$p(w_k|w_j) = \frac{\exp(c_k \cdot v_j)}{\sum_{i \in |V|} \exp(c_i \cdot v_j)}$$

Problem:

- Denominator is too expensive to compute!

Skip-gram in More Detail

Actual training procedure:

- Sample a target word, j
- Sample a **true** context word, c, and some **negative** context words from the entire vocabulary
- Optimize:

$$\log \sigma(c \cdot v_j) + \sum_i \mathrm{E}_{w_i \sim p(w)} [\log \sigma(-c_i \cdot v_j)]$$

where σ is the logistic function

- Use gradient descent to learn the parameters (what are the parameters here?)
- We've turned this into a different problem of separating true context words from fake ones!

Validation

Word similarity

- **Analogical reasoning task**

a : b :: c : d – what is *d*?

New York : New York Times :: Baltimore : ?

(Answer: Baltimore Sun)

man : king :: woman : ? (Answer: queen)

Solve by ranking vocabulary items according to assumption:

$$v_a - v_b = v_c - v_d$$

or $v_d = v_b - v_a + v_c$

(? = king – man + queen)

Impact of Hyperparameters

There are many hyperparameters that are important to the performance of word2vec:

- Weighting relative importance of context words
- Sampling procedure of negatives during training of target and context words ($p^{\frac{3}{4}}(w)$ works much better than standard $p(w)$!)

These have a large impact on performance (Levy et al., 2015)

- Applying analogous changes to previous approaches such as Singular Value Decomposition results in similar performance on word similarity tasks

Use of Word Embeddings in NLP

Can download pretrained word2vec embeddings from the web

Use word vectors as features in other NLP tasks

- Now very widespread

Advantages:

- Vectors are trained from a large amount of general-domain text: proxy of background knowledge!
- Cheap and easy to try

Disadvantages:

- Doesn't always work (especially if you have a lot of task-specific data)

Challenge Problem

Distributional similarity gives us a measure of relatedness which often works well, but it suffers from the antonym problem – synonyms and antonyms both share similar distributional properties!

How can we fix this? Brainstorm some suggestions with your neighbours.