



Meaning: Hearst Patterns and Distributional Semantics

Instructor: Jackie CK Cheung

COMP-550

J&M Ch. 20.2, 6 (3rd)



Word Sense Disambiguation

Figuring out which word sense is expressed in context

*His **hands** were tired from hours of typing.*

→ *hand.n.01*

*Due to her superior education, her **hand** was flowing and graceful.*

→ *hand.n.03*

General idea: use words in the context to disambiguate.
Which words above would help with this?

Possible Computational Approaches

A heuristic algorithm

- **Lesk's algorithm**

Supervised machine learning

- Possible, but requires a lot of work to annotate word sense information that we want to avoid

Unsupervised, or minimally supervised machine learning

- **Yarowsky's algorithm**

Lesk's Algorithm (1986)

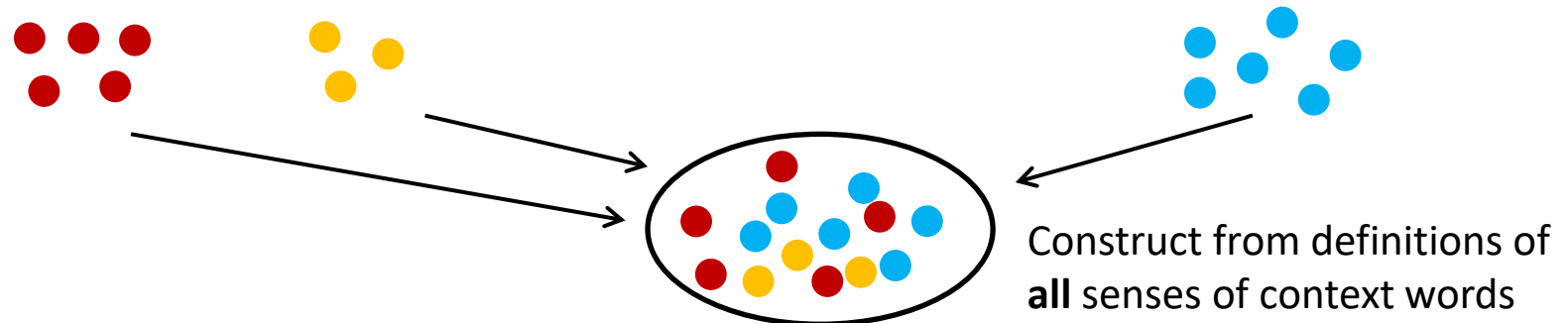
Use the dictionary definitions of a word's senses

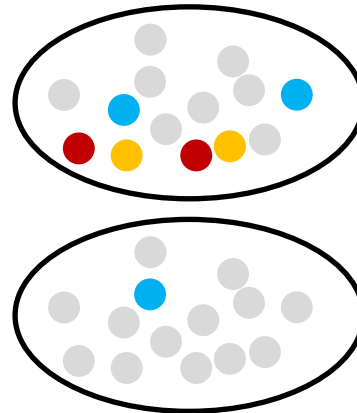
Steps to disambiguate word w :

1. Construct a bag of words representation of the context, B
2. For each candidate sense s_i of word w :
 - Calculate a signature of the sense by taking all of the words in the dictionary definition of s_i
 - Compute $\text{Overlap}(B, \text{signature}(s_i))$
3. Select the sense with the highest overlap score

Financial Bank or Riverbank?

... *deposit a cheque at the bank before it closed* ...

- B :
- $overlap(bank\#1, B)$
 - 6 overlaps found
- $overlap(bank\#2, B)$
 - 1 overlap found
- Decision: select sense 1.



Model Variations

Which dictionary to use? NLTK?

Use only dictionary definitions? Or include example sentences?

Ignore uninformative stopwords (e.g., *the, a, of*)?

Lemmatize when considering matches (*tomatoes* matches *tomato*)?

Exercise

Run the Lesk algorithm using NLTK/WordNet. Ignore stop words, include examples, count lemma overlap. Consider only the top two senses of bank.

1. I'll deposit the cheque at the **bank**.
2. The **bank** overflowed and water flooded the town.

Yarowsky's Algorithm (1995)

A method based on **bootstrapping**

Steps:

1. Gather a data set with target word to be disambiguated
2. Automatically label a small **seed set** of examples
3. Repeat the following for a while:
 - Train a supervised learning algorithm from the seed set
 - Apply the supervised model to the entire data set
 - Keep the highly confident classification outputs to be the new seed set
4. Use the last model as the final model

Yarowsky's Example

Step 1: Disambiguating *plant*

Sense	Training Examples (Keyword in Context)
?	... company said the <i>plant</i> is still operating
?	Although thousands of <i>plant</i> and animal species
?	... zonal distribution of <i>plant</i> life
?	... to strain microscopic <i>plant</i> life from the ...
?	vinyl chloride monomer <i>plant</i> , which is ...
?	and Golgi apparatus of <i>plant</i> and animal cells
?	... computer disk drive <i>plant</i> located in ...
?	... divide life into <i>plant</i> and animal kingdom
?	... close-up studies of <i>plant</i> life and natural
?	... Nissan car and truck <i>plant</i> in Japan is ...
?	... keep a manufacturing <i>plant</i> profitable without
?	... molecules found in <i>plant</i> and animal tissue
?	... union responses to <i>plant</i> closures
?	... animal rather than <i>plant</i> tissues can be
?	... many dangers to <i>plant</i> and animal life
?	company manufacturing <i>plant</i> is in Orlando ...
?	... growth of aquatic <i>plant</i> life in water ...
?	automated manufacturing <i>plant</i> in Fremont ,
?	... Animal and <i>plant</i> life are delicately
?	discovered at a St. Louis <i>plant</i> manufacturing
?	computer manufacturing <i>plant</i> and adjacent ...
?	... the proliferation of <i>plant</i> and animal life
?

Step 2: Initial Seed Set

Sense A:

- *plant* as in a lifeform

Other data

Sense B:

- *plant* as in a factory

Sense	Training Examples (Keyword in Context)
A	used to strain microscopic <i>plant life</i> from the ...
A	... zonal distribution of <i>plant life</i>
A	close-up studies of <i>plant life</i> and natural ...
A	too rapid growth of aquatic <i>plant life</i> in water ...
A	... the proliferation of <i>plant</i> and animal <i>life</i> ...
A	establishment phase of the <i>plant</i> virus <i>life</i> cycle ...
A	... that divide <i>life</i> into <i>plant</i> and animal kingdom
A	... many dangers to <i>plant</i> and animal <i>life</i> ...
A	mammals . Animal and <i>plant life</i> are delicately
A	beds too salty to support <i>plant life</i> . River ...
A	heavy seas, damage , and <i>plant life</i> growing on ...
A
?	... vinyl chloride monomer <i>plant</i> , which is ...
?	... molecules found in <i>plant</i> and animal tissue
?	... Nissan car and truck <i>plant</i> in Japan is ...
?	... and Golgi apparatus of <i>plant</i> and animal cells ...
?	... union responses to <i>plant</i> closures
?
?
?	... cell types found in the <i>plant</i> kingdom are ...
?	... company said the <i>plant</i> is still operating ...
?	... Although thousands of <i>plant</i> and animal species
?	... animal rather than <i>plant</i> tissues can be ...
?	... computer disk drive <i>plant</i> located in ...
B
B	automated manufacturing <i>plant</i> in Fremont ...
B	... vast manufacturing <i>plant</i> and distribution ...
B	chemical manufacturing <i>plant</i> , producing viscose
B	... keep a manufacturing <i>plant</i> profitable without
B	computer manufacturing <i>plant</i> and adjacent ...
B	discovered at a St. Louis <i>plant</i> manufacturing
B	... copper manufacturing <i>plant</i> found that they
B	copper wire manufacturing <i>plant</i> , for example ...
B	's cement manufacturing <i>plant</i> in Alpena ...
B	polystyrene manufacturing <i>plant</i> at its Dow ...
B	company manufacturing <i>plant</i> is in Orlando ...

Step 3: Train a Classifier

He went with a **decision-list** classifier (we didn't cover this one in class)

Initial decision list for <i>plant</i> (abbreviated)		
LogL	Collocation	Sense
8.10	<i>plant</i> life	⇒ A
7.58	manufacturing <i>plant</i>	⇒ B
7.39	life (within ±2-10 words)	⇒ A
7.20	manufacturing (in ±2-10 words)	⇒ B
6.27	animal (within ±2-10 words)	⇒ A
4.70	equipment (within ±2-10 words)	⇒ B
4.39	employee (within ±2-10 words)	⇒ B
4.30	assembly <i>plant</i>	⇒ B
4.10	<i>plant</i> closure	⇒ B
3.52	<i>plant</i> species	⇒ A
3.48	automate (within ±2-10 words)	⇒ B
3.45	microscopic <i>plant</i>	⇒ A
	...	

Note how new collocations are found for each sense

Step 3: Change Seed Set

Use only the cases where classifier is highly confident

Labeling previously untagged contexts
using the one-sense-per-discourse property

Change in tag	Disc. Numb.	Training Examples (from same discourse)
A → A	724	... the existence of <i>plant</i> and animal life ...
A → A	724	... classified as either <i>plant</i> or animal ...
? → A	724	Although bacterial and <i>plant</i> cells are enclosed
A → A	348	... the life of the <i>plant</i> , producing stem
A → A	348	... an aspect of <i>plant</i> life , for example
? → A	348	... tissues ; because <i>plant</i> egg cells have
? → A	348	photosynthesis, and so <i>plant</i> growth is attuned

Results

96% on binary word sense distinctions

Same result as with supervised methods, but with minimal amounts of annotation effort!

Outline

Detecting lexical semantic relationships

Distributional semantics

- Count-based methods

- Singular-value decomposition

- word2vec

Another Lexical Semantic Task

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	Context words
<i>Firth</i>	5	7	12	6	9	
<i>figure</i>	276	87	342	56	2	
<i>linguist</i>	153	1	42	5	34	
<i>1950s</i>	12	32	1	34	0	
<i>English</i>	15	34	9	5	21	
Target words	Co-occurrence counts					

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.

$$P(\textit{the}, \textit{linguistics}) = 0.0003$$

$$P(\textit{the}) = 0.1$$

$$P(\textit{linguistics}) = 0.0025$$

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

If ratio is < 1 , PMI is negative

People often discard negative values \rightarrow 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

$$\begin{matrix} X & = & W & \times & \Sigma & \times & C^T \\ |V| \times c & & |V| \times m & & m \times m & & m \times c \end{matrix}$$

- 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

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

- Use rows of W_k as new word representations

Views of Truncated SVD

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

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

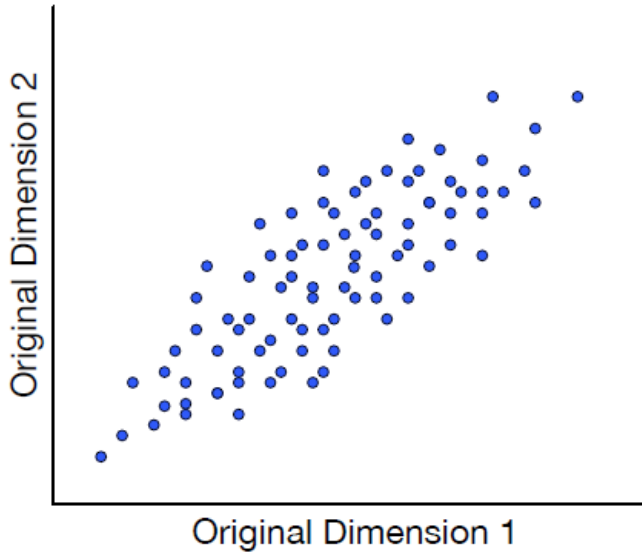
- i.e., X_k is the best possible approximation among any matrix of this rank, according to squared error.

Truncated SVD also corresponds to

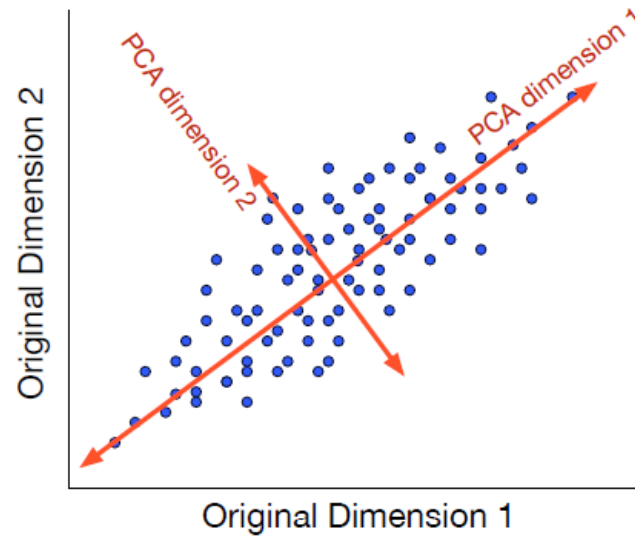
1. finding the principal components of the data, then
2. projecting down to the lower-dimensional subspace spanned by these principal components

(Principal component analysis)

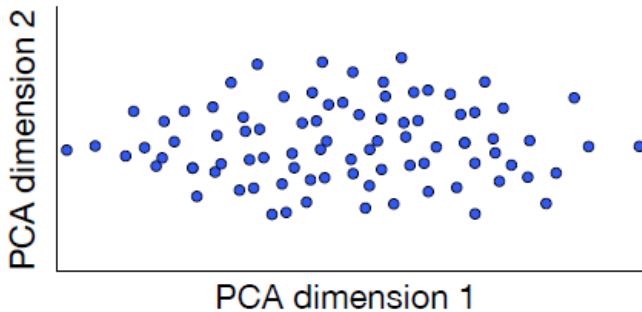
Principal Components Graphically



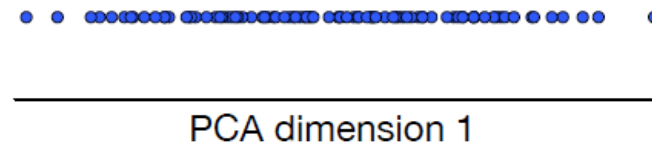
(a)



(b)



(c)



(d)

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_j s and c_j s 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

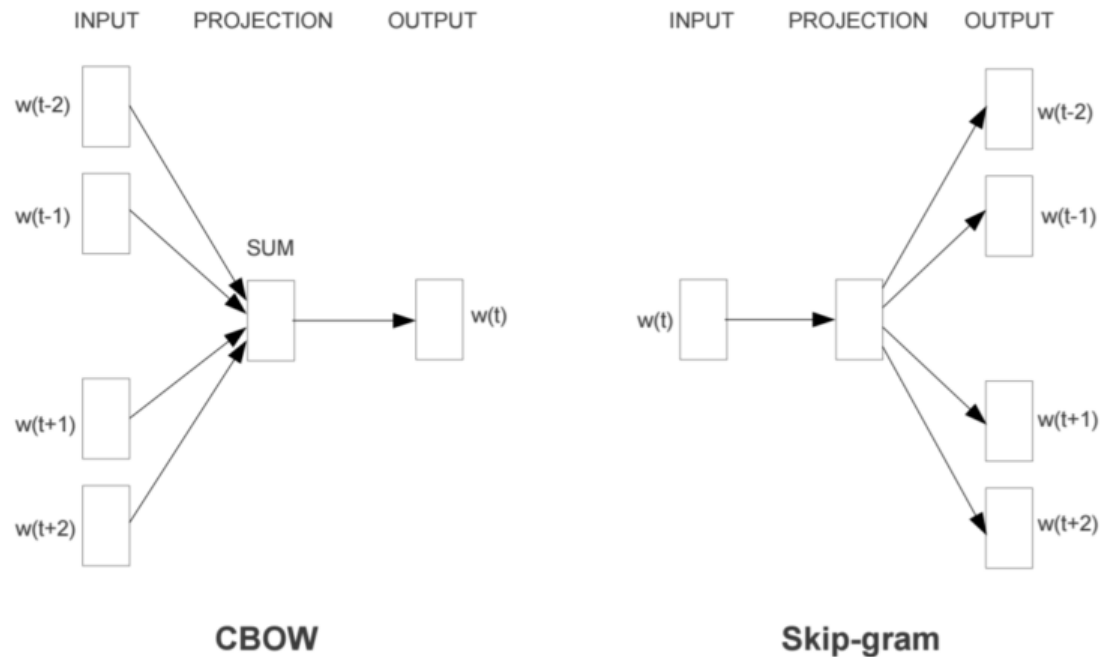


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

(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.

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

Note:

- Denominator is too expensive to compute!
- There is a way to speed this up using an approximation algorithm (**contrastive estimation**)

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:

$$v_a - v_b = v_c - v_d$$

$$\text{or } v_d = v_b - v_a + v_c$$

$$(? = \text{king} - \text{man} + \text{queen})$$

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

Pretrained word2vec embeddings available online

Use word vectors as features in other NLP tasks

- Now very widespread
- Use it to initialize a LSTM's input layer

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)