

# UNIT IV

# Vector Semantics and Embeddings

By

Dr. S. S. Gharde

Dept. of Information Technology/ AIML  
Government Polytechnic Nagpur

# Contents...

- Lexical Semantics, Vector Semantics, Words and Vectors
- Cosine for measuring similarity, TF-IDF: Weighing terms in the vector, Pointwise Mutual Information (PMI)
- Applications of the tf-idf or PPMI vector models, Word2vec, Visualizing Embeddings, Semantic properties of embeddings

# Lexical Semantics      cont...

- Linguistic study of word meaning, which is called **Lexical Semantics**.
- Dictionary meaning: **mouse (N)**
  - 1. any of numerous small rodents...
  - 2. a hand-operated device that controls a cursor...
- Here the form *mouse* is the **lemma**, also called the **citation form**.
- The form *mouse* would also be the lemma for the word *mice*
- Similarly *sing* is the lemma for *sing, sang, sung*.
- The specific forms *sung or carpets or sing* are called **wordforms**.

# Lexical Semantics

cont...

- Each lemma can have multiple meanings.
- Each of these aspects of the meaning is called **word sense**.
- Example: the lemma *mouse* can refer to the *rodent* or the *cursor control device*.
- The fact that lemmas can be **polysemous** (have multiple senses) can make interpretation difficult.

# Lexical Semantics cont...

- **Synonymy** One important component of word meaning is the relationship between word senses.
- When one word has a sense whose meaning is identical to a sense of another word, or nearly identical, we say the two senses of those two words are **synonyms**.
  - Synonyms include such pairs as couch/sofa, vomit/throw up, car/automobile
- **Definition:** **Synonymy is that two words are synonymous if they are substitutable for one another** in any sentence without changing the *truth conditions* of the sentence, the situations in which the sentence would be true.
- In this case that the two words have the same **propositional meaning**.

# Lexical Semantics cont...

- Principle of contrast states that a difference in linguistic form is always associated with some difference in meaning.
- Example: the word  $H_2O$  is used in scientific contexts and would be inappropriate in a drinking  $H_2O$ .
- Here, *water* would be more appropriate.

# Lexical Semantics cont...

- **Word Similarity:** While words don't have many synonyms, most words do have lots of similar words.
- *Cat* is not a synonym of *dog*, but *cats* and *dogs* are certainly similar words.
- In moving from synonymy to similarity, it will be useful to shift from talking about relations between **word senses (like synonymy)** to relations between **words (like similarity)**.
- The notion of word **similarity** is very useful in larger semantic tasks.
- Knowing how similar two words are can help in computing how similar the meaning of two phrases or sentences are, a very important component of tasks like **question answering, paraphrasing, and summarization**.

# Lexical Semantics cont...

- **Word Relatedness:** The meaning of two words can be related in ways other than similarity.
- One such class of connections is called **word relatedness**, also traditionally called **word association** in psychology.
  - Example: Consider the meanings of the words *coffee* and *cup*, *scalpel* and *surgeon*.
- One common kind of relatedness between words is if they belong to the same **semantic field**.
- A **semantic field** is a set of words which cover a particular semantic domain and bear structured relations with each other.
  - For example, words might be related by being in the semantic field of
    - hospitals (surgeon, scalpel, nurse, anesthetic, hospital)
    - restaurants (waiter, menu, plate, food, chef)
    - houses (door, roof, kitchen, family, bed)

# Lexical Semantics

- **Semantic Frames and Roles:** Closely related to semantic fields is the idea of a semantic frame.
- A **semantic frame** is a set of words that denote perspectives or participants in a particular type of event.
  - Example: *Buy, Sell, Pay*
- **Connotation:** Finally, words have affective meanings or connotations.
- The word **connotation** has different meanings in different fields, but here we use it to mean the aspects of a word's meaning that are related to a writer or reader's emotions, sentiment, opinions, or evaluations.
  - For example:
  - positive connotations (**happy**), negative connotations (**sad**)
  - **innocent** (positive connotation) and **naive** (negative connotation)
  - positive evaluation (**great, love**) and negative evaluation (**terrible, hate**)
  - fake, knockoff, forgery
  - copy, replica, reproduction

# Vector Semantics

- Vector semantics is the standard way to represent word meaning in NLP.
- The idea of vector semantics is to represent a word as a point in a multidimensional semantic space that is derived from the distributions of word neighbors.
- Vectors for representing words are called embeddings.

# Vector Semantics

- What does recent English borrowing *ongchoi* mean?
- Suppose you see these sentences:
  - Ong choi is delicious **sautéed with garlic**.
  - Ong choi is superb **over rice**
  - Ong choi **leaves** with salty sauces
- And you've also seen these:
  - ...spinach **sautéed with garlic over rice**
  - Chard stems and **leaves** are **delicious**
  - Collard greens and other **salty leafy greens**
- Conclusion:
  - **Ongchoi is a leafy green like spinach, chard, or collard greens**
  - We could conclude this based on words like "leaves" and "delicious" and "sautéed"

# Vector Semantics

NLP UNIT IV (C)SSG

- Idea 1: Defining meaning by linguistic distribution

Idea 2: Meaning as a point in multidimensional space

- We define meaning of a word as a vector
- Called an "embedding" because it's embedded into a space
- The standard way to represent meaning in NLP

**Every modern NLP algorithm uses embeddings as the representation of word meaning**

- Fine-grained model of meaning for similarity of vector semantics offers enormous power to NLP applications.

# Vector Semantics

- Most commonly used models
- **tf-idf Model**
  - the meaning of a word is defined by a simple function of the counts of nearby words.
- **word2vec Model**
  - Constructing short, dense vectors that have useful semantic properties.
- **The cosine Model**
  - The standard way to use embeddings to compute semantic similarity, between two words, two sentences, or two documents, an important tool in practical applications like question answering, summarization, or automatic essay grading.

# Words and Vectors

- Vector or distributional models of meaning are generally based on a **co-occurrence matrix**, a way of representing how often words co-occur.
- Two popular matrices:
  - term-document matrix
  - term-term matrix

# Words and Vectors

- **Vectors and documents (term-document matrix)**
- In a term-document matrix, each **row** represents a **word** in the vocabulary and each **column** represents a **document** from some collection of documents.
- Fig. 6.2 shows a small selection from a term-document matrix showing the occurrence of four words in four plays by Shakespeare.
- Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column)

# Words and Vectors

- **Vectors and documents (term-document matrix)**
- Fig. 6.2 shows a small selection from a term-document matrix showing the occurrence of four words in four plays by Shakespeare.
- Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column)

|        | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|--------|----------------|---------------|---------------|---------|
| battle | 1              | 0             | 7             | 13      |
| good   | 114            | 80            | 62            | 89      |
| fool   | 36             | 58            | 1             | 4       |
| wit    | 20             | 15            | 2             | 3       |

**Figure 6.2** The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

# Words and Vectors:

## term-document matrix

- The term-document matrix of Fig. 6.2 was first defined as part of the **vector space model** of information retrieval.
- Term-document matrices were originally defined as a means of finding similar documents for the task of document information retrieval.
- Two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar.
- A **vector space** is a collection of vectors, characterized by their dimension.
- A **vector** is, at heart, just a list or array of numbers.
- So **As You Like It** is represented as the list [1,114,36,20] and **Julius Caesar** is represented as the list [7,62,1,2]

|        | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|--------|----------------|---------------|---------------|---------|
| battle | 1              | 0             | 7             | 13      |
| good   | 114            | 80            | 62            | 89      |
| fool   | 36             | 58            | 1             | 4       |
| wit    | 20             | 15            | 2             | 3       |

**Figure 6.2** The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

# Words and Vectors

## **term-document matrix**

- the term-document matrix has  $|V|$  rows (one for each word type in the vocabulary) and  $D$  columns (one for each document in the collection)
- **Information retrieval (IR)** is the task of finding the document  $d$  from the  $D$  documents in some collection that best matches a query  $q$ .

# More common: word-word matrix (or "term-context matrix")

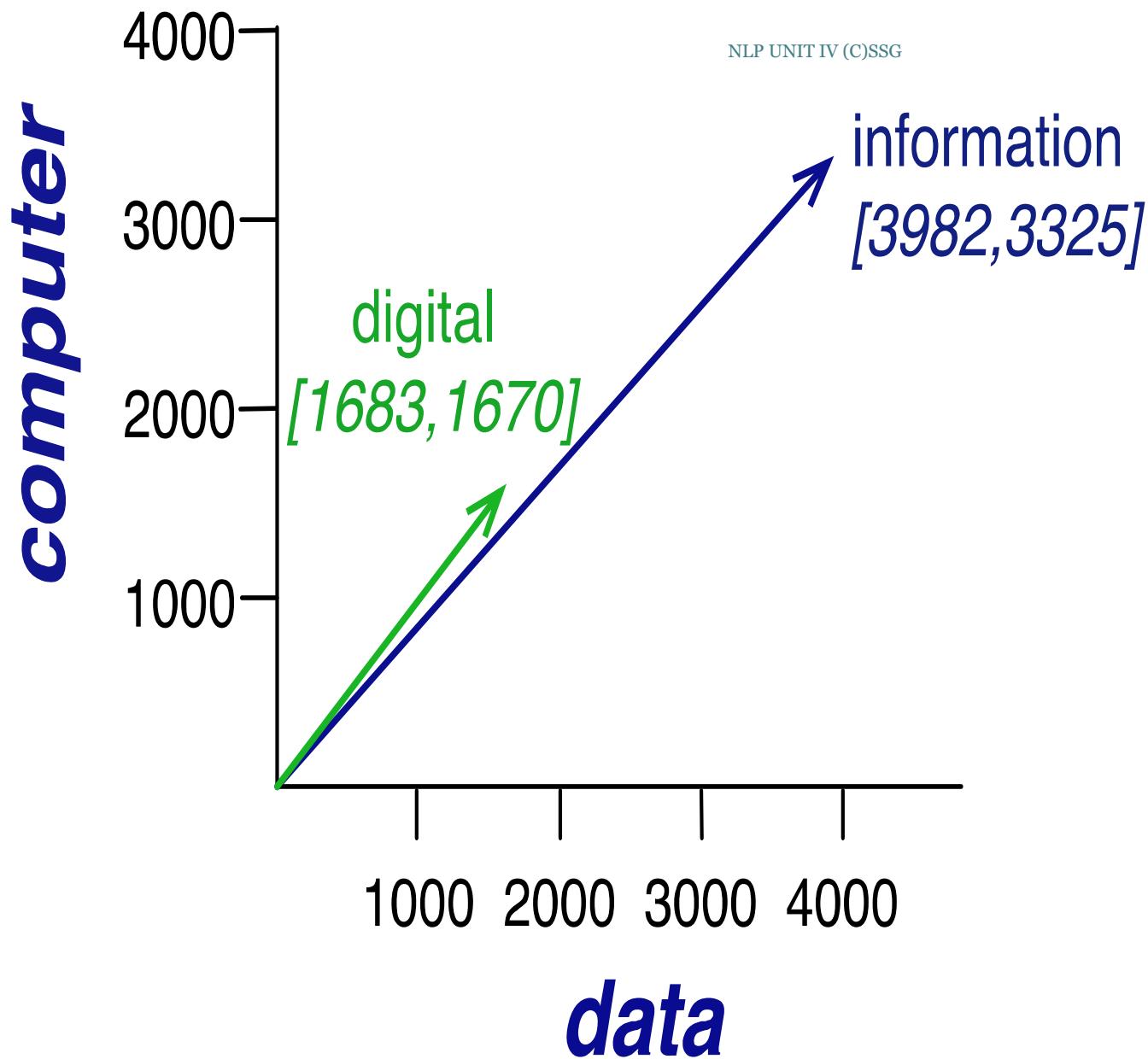
- Two **words** are similar in meaning if their context vectors are similar

is traditionally followed by **cherry** pie, a traditional dessert

often mixed, such as **strawberry** rhubarb pie. Apple pie

computer peripherals and personal **digital** assistants. These devices usually  
a computer. This includes **information** available on the internet

|             | aardvark | ... | computer | data | result | pie | sugar | ... |
|-------------|----------|-----|----------|------|--------|-----|-------|-----|
| cherry      | 0        | ... | 2        | 8    | 9      | 442 | 25    | ... |
| strawberry  | 0        | ... | 0        | 0    | 1      | 60  | 19    | ... |
| digital     | 0        | ... | 1670     | 1683 | 85     | 5   | 4     | ... |
| information | 0        | ... | 3325     | 3982 | 378    | 5   | 13    | ... |



# Cosine for measuring similarity

- To measure similarity between two target words  $v$  and  $w$ , we need a metric that takes two vectors and gives a measure of their similarity.
- The most common similarity metric is the cosine of the angle between the vectors.
- The cosine is based on dot product operator from linear algebra, also called the inner product.
- The dot product acts as a similarity metric because it will tend to be high just when the two vectors have large values in the same dimensions.
- The vector length is defined as

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^N v_i^2}$$

# Cosine for measuring similarity

- Normalized dot product turns out to be the same as the cosine of the angle between the two vectors, following from the definition of the dot product between two vectors  $\mathbf{a}$  and  $\mathbf{b}$ :

$$\begin{aligned}\mathbf{a} \cdot \mathbf{b} &= |\mathbf{a}| |\mathbf{b}| \cos \theta \\ \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} &= \cos \theta\end{aligned}$$

- The cosine similarity metric between two vectors  $\mathbf{v}$  and  $\mathbf{w}$  thus can be computed as:

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^N v_i^2}$$

# Cosine example

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\vec{v}}{\|\vec{v}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

|             | <b>pie</b> | <b>data</b> | <b>computer</b> |
|-------------|------------|-------------|-----------------|
| cherry      | 442        | 8           | 2               |
| digital     | 5          | 1683        | 1670            |
| information | 5          | 3982        | 3325            |

$$\cos(\text{cherry}, \text{information}) =$$

$$\frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital}, \text{information}) =$$

$$\frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

The model decides that *information* is way closer to *digital* than it is to *cherry*, a result that seems sensible.

# TF-IDF: Weighing terms in the vector

- The co-occurrence matrices we have seen represent each cell by word frequencies.
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it*, or *they* are not very informative about the context
- How can we balance these two conflicting constraints?
- There are two common solutions to this problem:
  - tf-idf weighting (usually used when the dimensions are documents.)
  - PPMI algorithm (usually used when the dimensions are words).

# TF-IDF: Weighing terms in the vector

- **tf-idf:** tf-idf value for word t in document d:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

Words like "the" or "it" have very low idf

- **PMI:** (Pointwise mutual information)

$$\square \text{PMI}(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

See if words like "good" appear more often with "great" than we would expect by chance

# TF-IDF: Weighing terms in the vector

- The tf-idf weighting is the product of two terms Term frequency (tf) and Inverse document frequency (idf)
- Term frequency (tf)
  - $\text{tf}_{t,d} = \text{count}(t,d)$   
Instead of using raw count, we squash a bit:
    - $\text{tf}_{t,d} = \log_{10}(\text{count}(t,d)+1)$
- Document frequency (df)
  - $\text{df}_t$  is the number of documents  $t$  occurs in.
  - (note this is not collection frequency: total count across all documents)
  - "Romeo" is very distinctive for one Shakespeare play:

|        | Collection Frequency | Document Frequency |
|--------|----------------------|--------------------|
| Romeo  | 113                  | 1                  |
| action | 113                  | 31                 |

# TF-IDF: Weighing terms in the vector

- Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left( \frac{N}{\text{df}_t} \right)$$

- N is the total number of documents in the collection
- The tf-idf weighted value  $\text{wt}_{t,d}$  for word t in document d thus combines term frequency  $t_{ft,d}$  with idf.

| Word     | df | idf   |
|----------|----|-------|
| Romeo    | 1  | 1.57  |
| salad    | 2  | 1.27  |
| Falstaff | 4  | 0.967 |
| forest   | 12 | 0.489 |
| battle   | 21 | 0.246 |
| wit      | 34 | 0.037 |
| fool     | 36 | 0.012 |
| good     | 37 | 0     |
| sweet    | 37 | 0     |

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

# Pointwise mutual information (PMI)

- An alternative weighting function to tf-idf, positive pointwise mutual information (PPMI).
- It is used for term-term-matrices, when the vector dimensions correspond to words rather than documents.
- 
- PPMI draws on the intuition that the best way to weigh the association between two words is to ask how much more the two words co-occur in our corpus than we would have a priori expected them to appear by chance.

# Pointwise mutual information (PMI)

- **Pointwise mutual information:** Pointwise mutual information is a measure of how often two events x and y occur, compared with what we would expect if they were independent.

$$\text{PMI}(X, Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- **PMI between two words:** (Church & Hanks 1989)  
The pointwise mutual information between a target word w and a context word c is then defined as:

$$\text{PMI}(w, c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

# Positive Pointwise Mutual Information

- PMI ranges from  $-\infty$  to  $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine  $w_1$  and  $w_2$  whose probability is each  $10^{-6}$
    - Hard to be sure  $p(w_1, w_2)$  is significantly different than  $10^{-12}$
    - Plus it's not clear people are good at "unrelatedness"
- So we just replace negative PMI values by 0
- Positive PMI (**PPMI**) between  $w$  and  $c$ :

$$\text{PPMI}(w, c) = \max\left(\log_2 \frac{P(w, c)}{P(w)P(c)}, 0\right)$$

# Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- $f_{ij}$  is number of times  $w_i$  occurs in context  $c_j$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{i^*} = \frac{\sum_{j=1}^C f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^W f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

|                | computer | data | result | pie | sugar | count(w) |
|----------------|----------|------|--------|-----|-------|----------|
| cherry         | 2        | 8    | 9      | 442 | 25    | 486      |
| strawberry     | 0        | 0    | 1      | 60  | 19    | 80       |
| digital        | 1670     | 1683 | 85     | 5   | 4     | 3447     |
| information    | 3325     | 3982 | 378    | 5   | 13    | 7703     |
| count(context) | 4997     | 5673 | 473    | 512 | 61    | 11716    |

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i^*} p_{*j}}$$

$$ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

|                    | computer | <u>data</u> | result | pie | sugar | count(w) |
|--------------------|----------|-------------|--------|-----|-------|----------|
| cherry             | 2        | 8           | 9      | 442 | 25    | 486      |
| strawberry         | 0        | 0           | 1      | 60  | 19    | 80       |
| digital            | 1670     | 1683        | 85     | 5   | 4     | 3447     |
| <u>information</u> | 3325     | 3982        | 378    | 5   | 13    | 7703     |
| count(context)     | 4997     | 5673        | 473    | 512 | 61    | 11716    |

$$p_{ij} = \frac{f_{ij}}{\sum_{j=1}^C f_{ij}}$$

- $p(w=\text{information}, c=\text{data}) = 3982/11716 = 0.3399$

- $p(w=\text{information}) = 7703/11716 = 0.6575$

- $p(c=\text{data}) = 5673/11716 = 0.4842$

$$\hat{p}_{ij} = \frac{\sum_{i=1}^W f_{ij}}{N}$$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N} \quad p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*} p_{*j}}$$

- $PPMI(\text{information}, \text{data}) = \log_2 (0.3399 / (0.6575 * 0.4842)) = 0.0944$

# Applications of the tf-idf or PPMI vector models

- The **tf-idf model** of meaning is often used for document functions like deciding if two documents are similar.
- Document similarity is also useful for all sorts of applications:
  - Information retrieval
  - Plagiarism detection
  - News recommender systems
  - Even for digital humanities tasks like comparing different versions of a text to see which are similar to each other.
- Either the **PPMI model** or the tf-idf model can be used to compute word similarity, for tasks like:
  - Finding word paraphrases
  - Tracking changes in word meaning
  - Automatically discovering meanings of words in different corpora.

# Word2vec

- tf-idf (or PMI) vectors are
  - **long** (length  $|V| = 20,000$  to  $50,000$ )
  - **sparse** (most elements are zero)
- Alternative: learn vectors which are
  - **short** (length  $50\text{-}1000$ )
  - **dense** (most elements are non-zero)
- Dense vectors work better in every NLP task than sparse vectors.
- Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
- Idea: **predict** rather than **count**

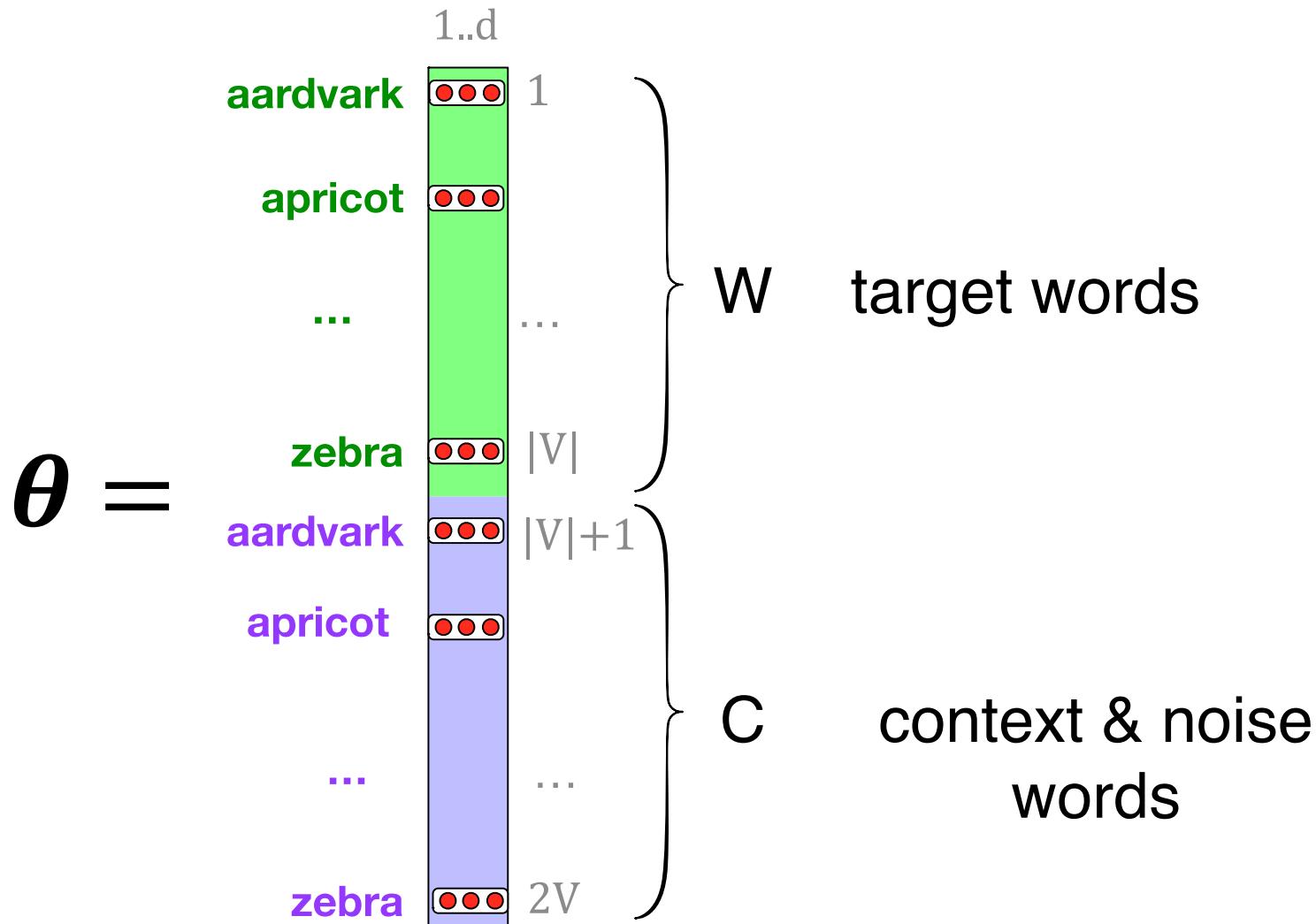
# Word2vec

- One method for computing embeddings: **skip-gram with negative sampling**, sometimes called SGNS.
- The skip-gram algorithm is one of two algorithms in a software package called word2vec
- Sometimes the algorithm is loosely referred to as word2vec.
- The word2vec methods are fast, efficient to train, and easily available online with code and pretrained embeddings.
- Word2vec embeddings are static embeddings.

# Word2vec: Skip-gram classifier

- A probabilistic classifier, given
  - a test target word  $w$
  - its context window of  $L$  words  $c_{1:L}$
- Estimates probability that  $w$  occurs in this window based on similarity of  $w$  (embeddings) to  $\mathcal{C}_{1:L}$  (embeddings).
- To compute this, we just need embeddings for all the words.

# Word2vec: Skip-gram classifier

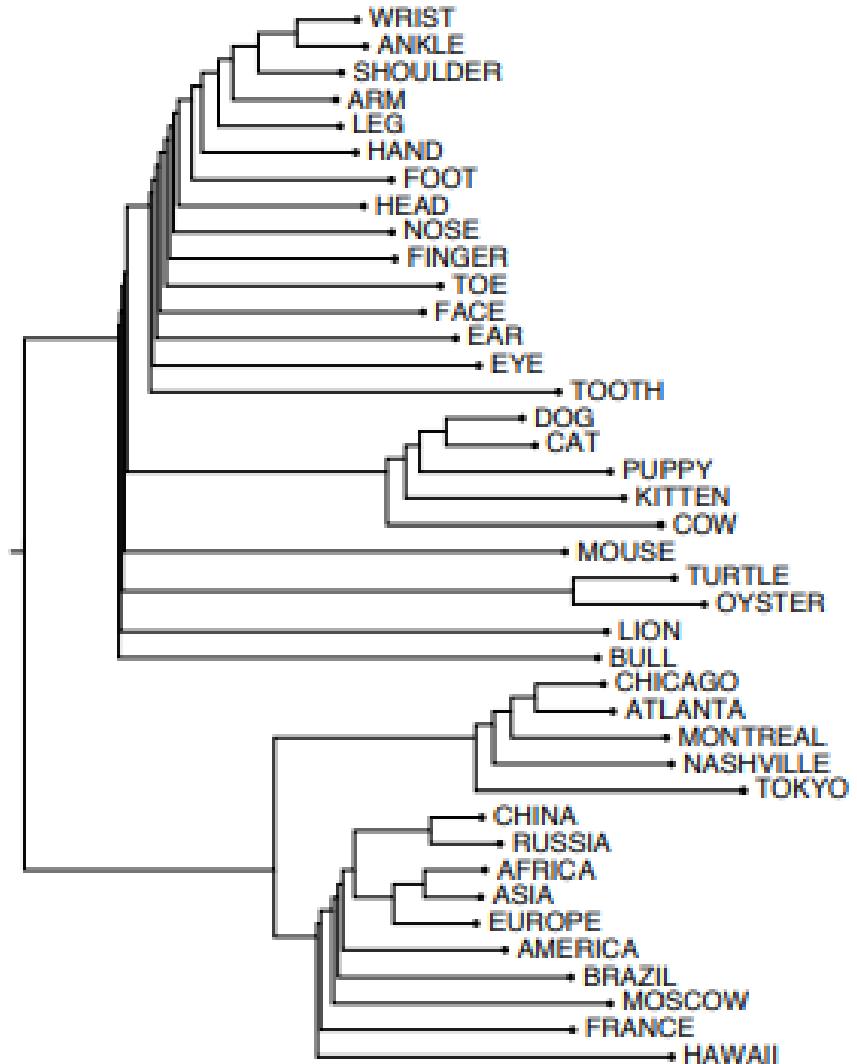


# Visualizing Embeddings

- Visualizing embeddings is an important goal in helping understand, apply, and improve these models of word meaning.
- The simplest way to visualize the meaning of a word  $w$  embedded in a space is to list the most similar words to  $w$  by sorting the vectors for all words in the vocabulary by their cosine with the vector for  $w$ .
- For example the 7 closest words to frog using the GloVe embeddings are:
  - frogs, toad, litoria, leptodactylidae, rana, lizard, and eleutherodactylus

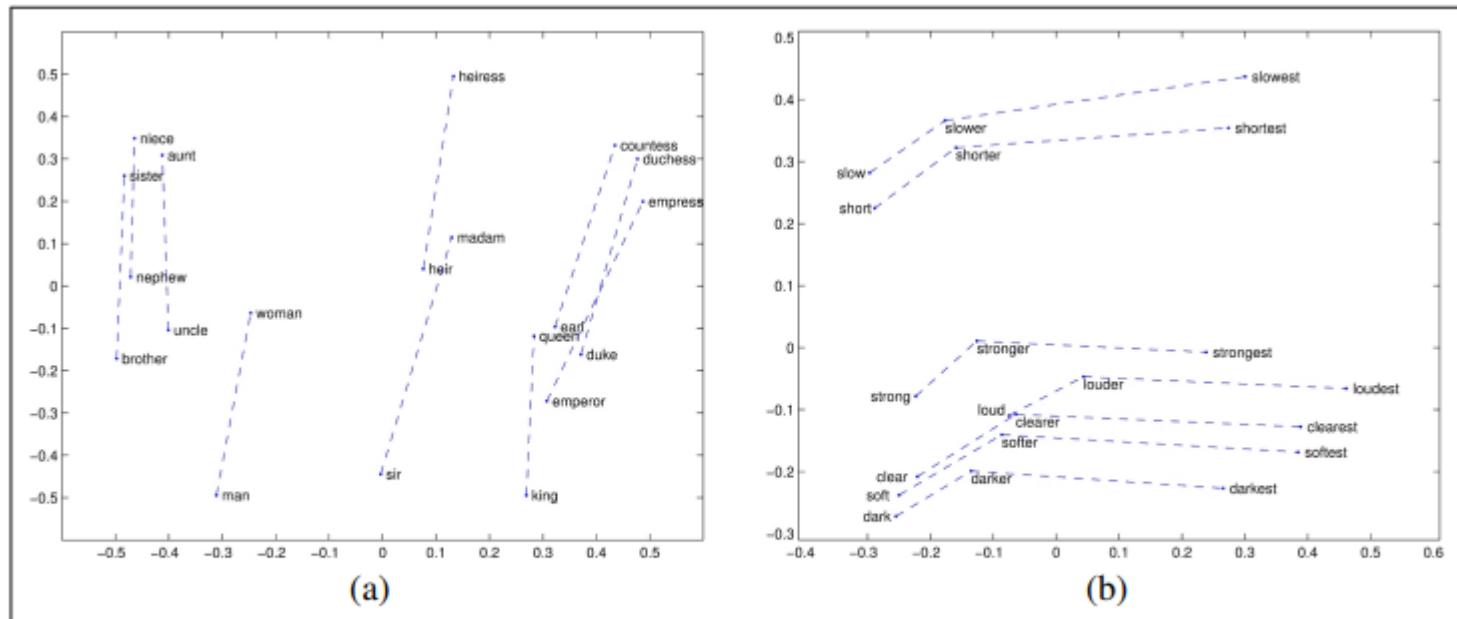
# Visualizing Embeddings

Another visualization method is to use a clustering algorithm to show a **hierarchical representation** of which words are similar to others in the embedding space.



# Visualizing Embeddings

The most common visualization method is to project the 100 dimensions of a word down into 2 dimensions



**Figure 6.16** Relational properties of the GloVe vector space, shown by projecting vectors onto two dimensions. (a)  $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$  is close to  $\overrightarrow{\text{queen}}$ . (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

# Semantic properties of embeddings

- **Different types of similarity or association:**
  - Two kinds of similarity or association between words
  - Two words have **first-order co-occurrence** (sometimes called syntagmatic association) if they are typically nearby each other.
    - Thus wrote is a first-order associate of book or poem.
  - Two words have **second-order co-occurrence** (sometimes called paradigmatic association) if they have similar neighbors.
    - Thus wrote is a second-order associate of words like said or remarked.

# Semantic properties of embeddings

- **Analogy/Relational Similarity:**
  - Another semantic property of embeddings is their ability to capture relational meanings.
  - In an important early vector space model of cognition, proposed the parallelogram model for solving simple analogy problems of the form  $a$  is to  $b$  as  $a^*$  is to what?.

# End of UNIT IV