UNIT IV
Information Retrieval

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 "sauteed"

Vector Semantics

- 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

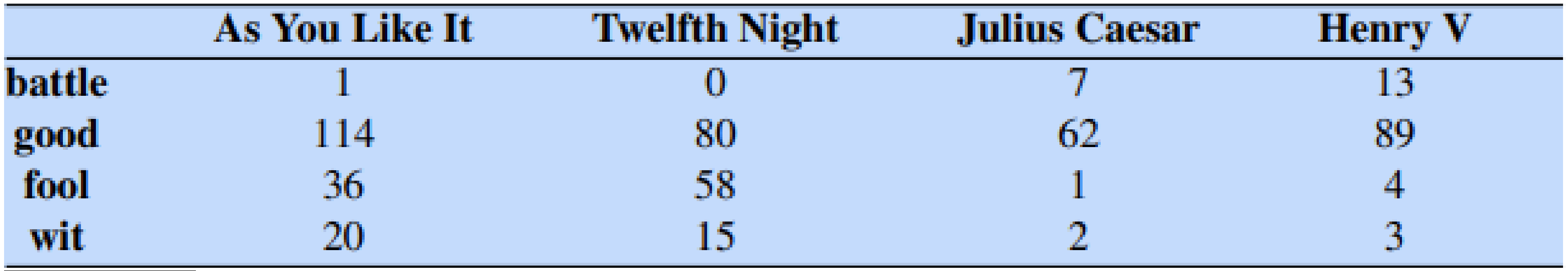
Words and Vectors

- **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.

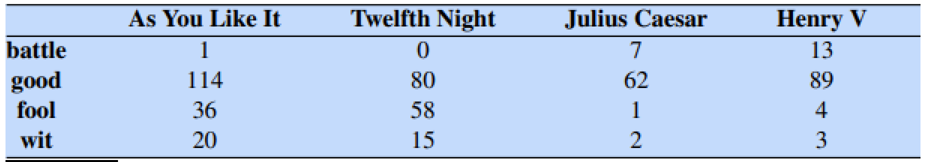
- Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column)



	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



	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.

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]

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)

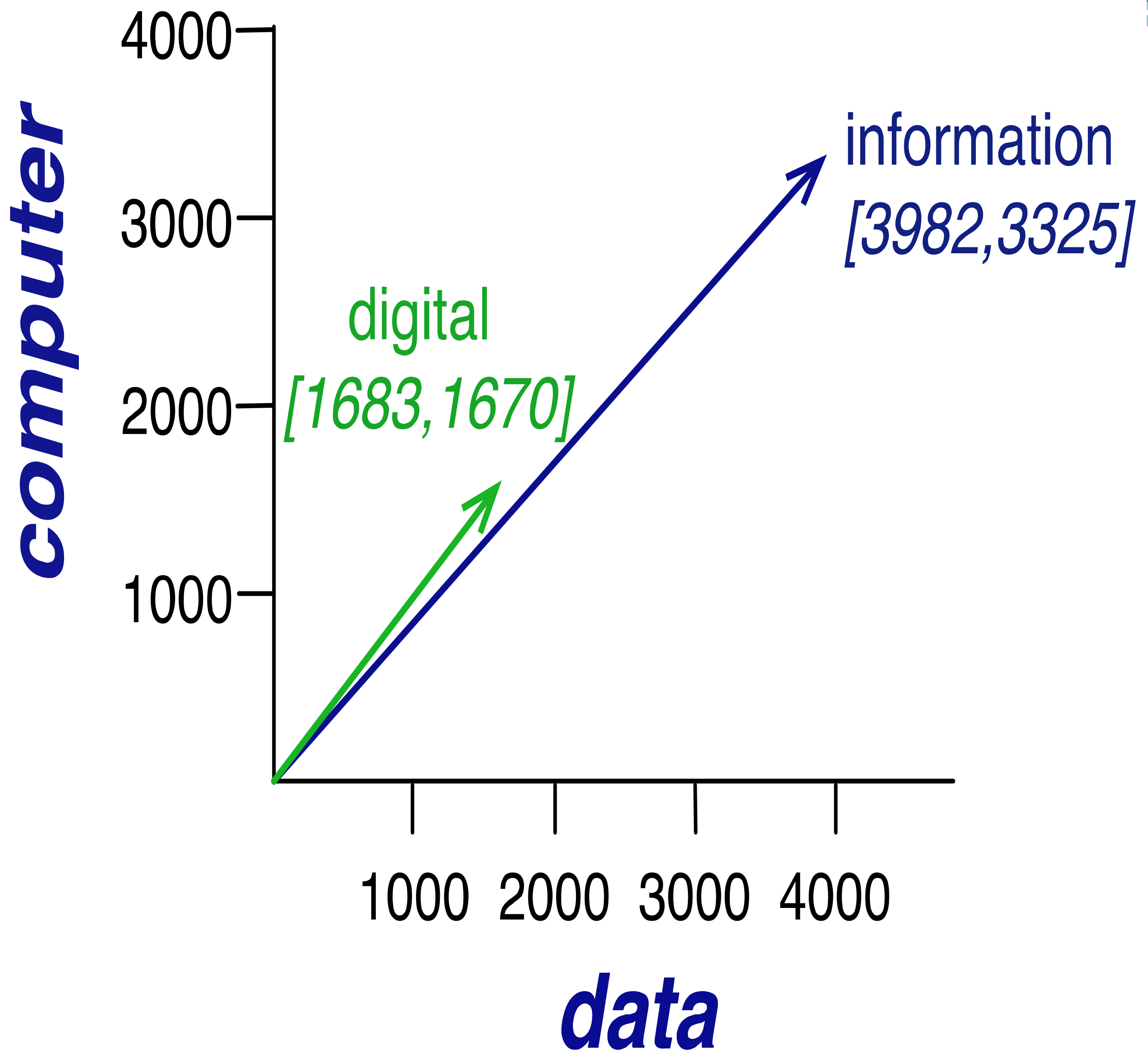
- **Information retrieval (IR)** is the task of finding the document d from the D documents in some collection that best matches a query q.

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	...


4000
3000
2000
1000
computer
digital
[1683,1670]
information
[3982,3325]
1000
2000
3000
4000
data

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 a and 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 v and 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

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

	pie	data	computer
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:

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

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

- **PMI:** (Pointwise mutual information)
  - $\mathrm{PMI}(w_1, w_2) = \log \frac{p(w_1,w_2)}{p(w_1)p(w_2)}$

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)
  - $\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$

  Instead of using raw count, we squash a bit:
  - $\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1)$

- Document frequency (df)
  - 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)

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

  - N is the total number of documents in the collection

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

- The tf-idf weighted value wt,d for word t in document d thus combines term frequency tft, d with idf.

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

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.

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

- **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:

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

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 w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) 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

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

	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

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{W \quad C}$$

$$f_{ij}$$

$$i{=}1 \quad j{=}1$$

	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

- p(w=information, c=data) = 3982/11716 = 0.3399
- p(w=information) = 7703/11716 = 0.6575
- p(c=data) = 5673/11716 = 0.4842

$$p(w_i) = \frac{\overset{C}{\underset{j=1}{}} f_{ij}}{N} \qquad p(c_j) = \frac{\overset{W}{\underset{i=1}{}} f_{ij}}{N}$$

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

- PPMI(information, 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 50-1000)
  - **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.

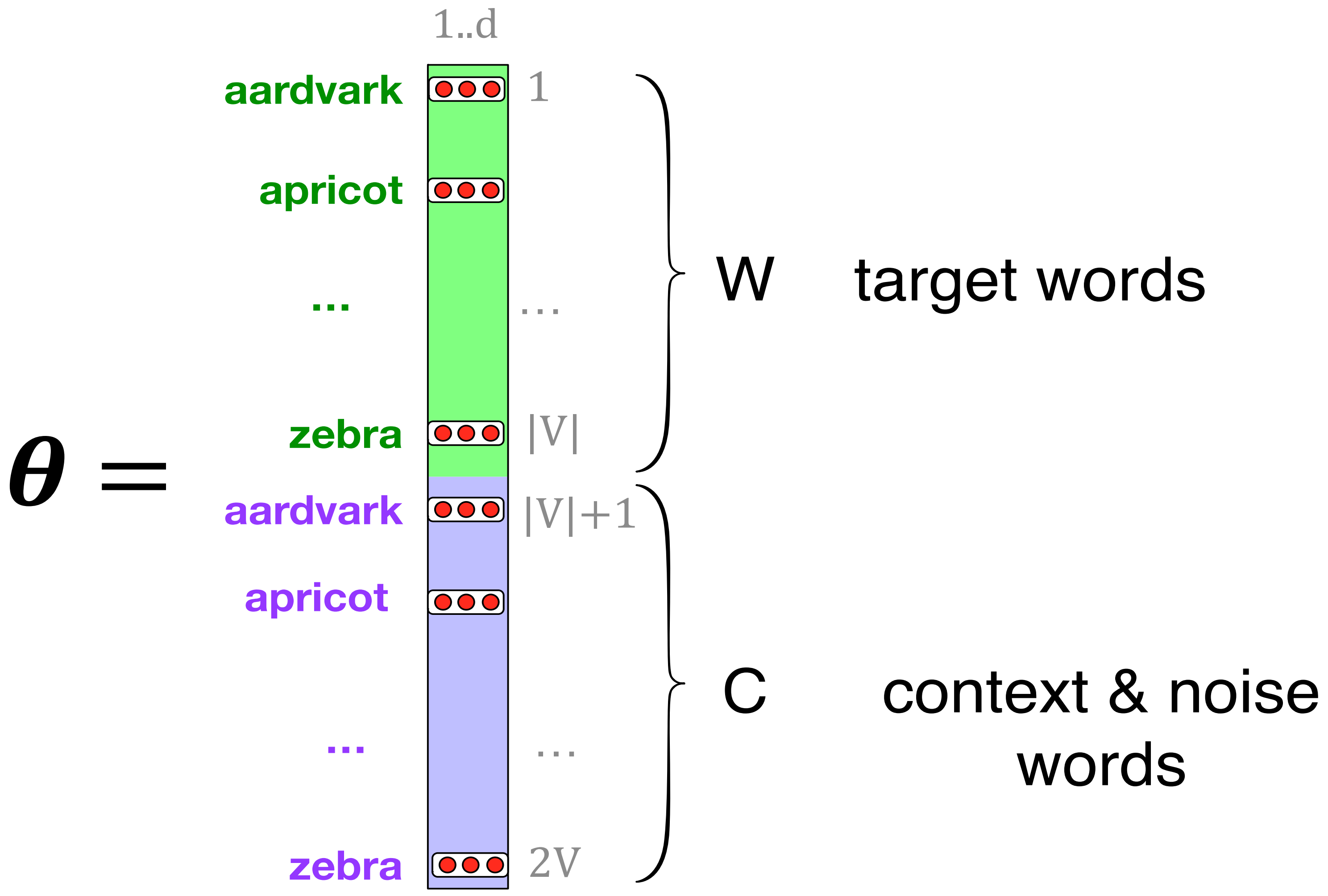
Word2vec: Skip-gram classifier

- 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 $\boldsymbol{c}_{1:L}$ (embeddings).

- To compute this, we just need embeddings for all the words.

Word2vec: Skip-gram classifier



Visualizing Embeddings

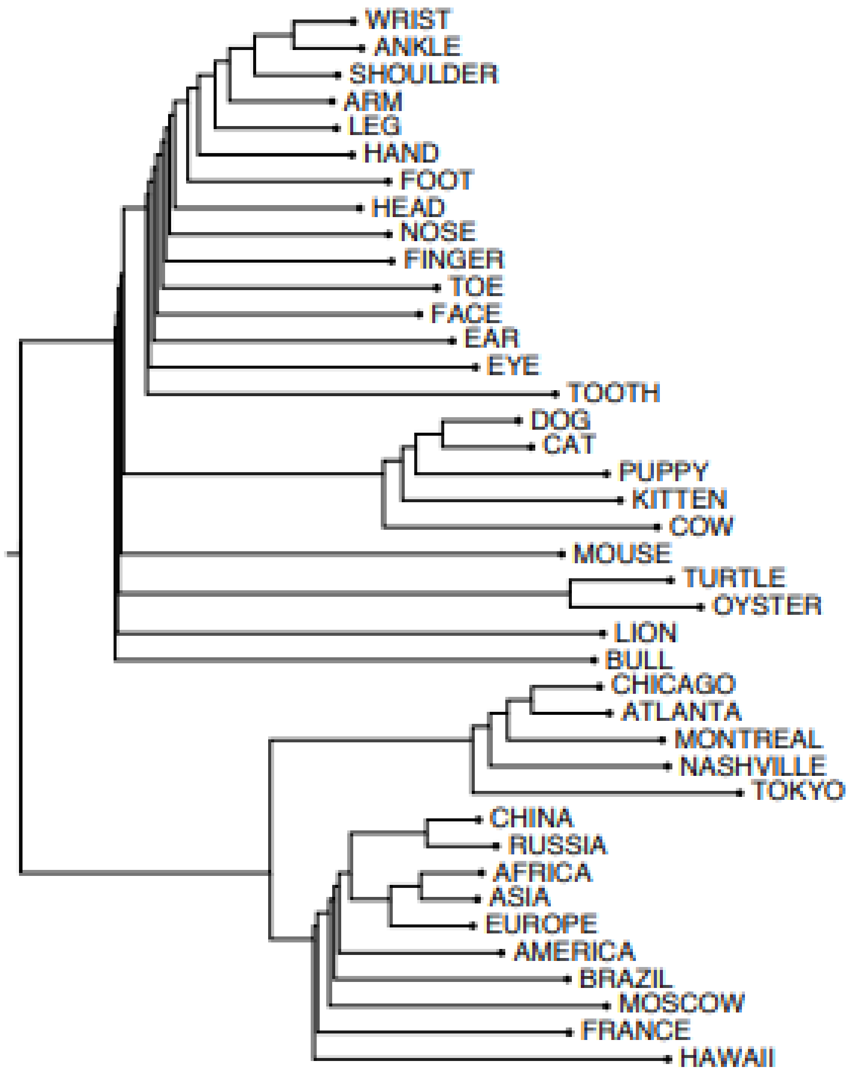
- Visualizing embeddings is an important goal in helping understand, apply, and improve these models of word meaning.

- 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

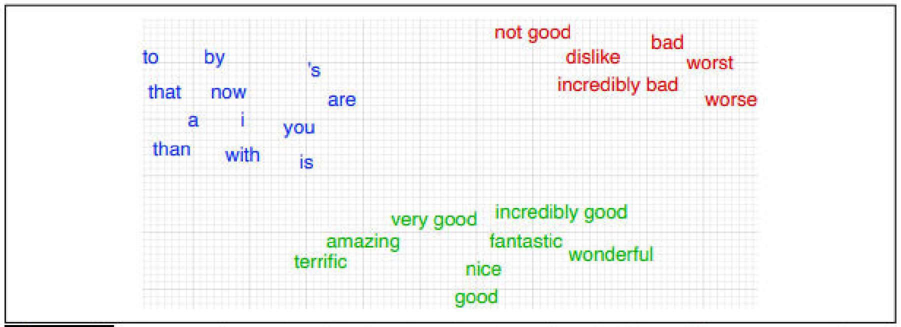
Another visualization method is to use a clustering algorithm to show a hierarchical representation of which words are similar to others in the embedding space.



Visualizing Embeddings

The most common visualization method is

to project the 100 dimensions of a word down into 2 dimensions



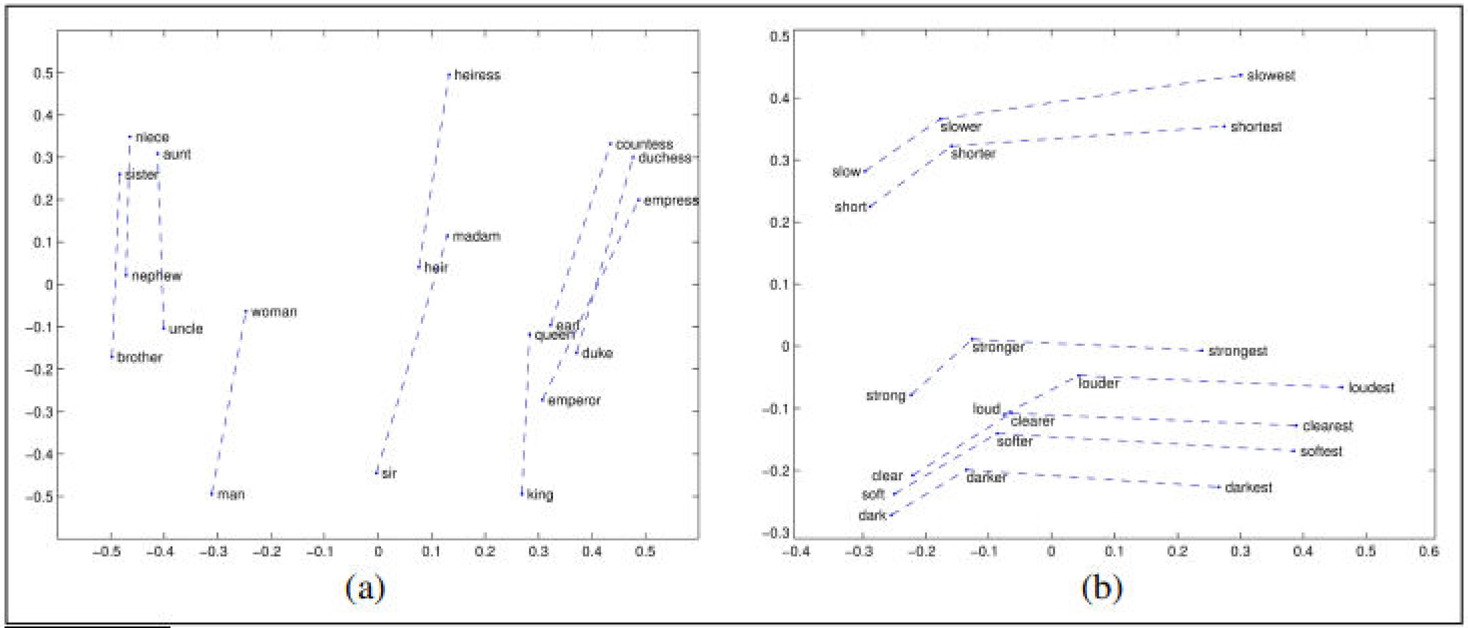


Figure 6.16 Relational properties of the GloVe vector space, shown by projecting vectors onto two dimensions. (a) $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$. (b) offsets seem to capture comparative and superlative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