Information Retrieval

Expanding classic IR

- Word Embeddings – part 1
- QA
- Recommender Systems
- Image and Audio Signals as data

Development:
Moshe Friedman

Credits:
Yoav Goldberg, Ido Dagan, Reut Tsarfaty , Moshe Koppel, Wei Song, David Bamman, Ed Grefenstette, Chris Manning, Tsvi Kuflik, Hinrich Schütze, Christina Lioma and more

Information Retrieval - administration

Moshe Friedman

Email: moshefr.teach@gmail.com

Reception time: before/after lesson/zoom with coordination

Query expansion – Recap

- Query expansion is another method for increasing recall.
- We use “global query expansion” to refer to “global methods for query reformulation”.
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.
- Main information we use: (near-)synonymy
- A publication or database that collects (near-)synonyms is called a thesaurus.
- We will look at two types of thesauri: manually created and automatically created.

3

Relevance feedback: Basic idea – Recap

- Relevance feedback: user feedback on relevance of docs in initial set of results
  - The user issues a (short, simple) query.
  - The search engine returns a set of documents.
  - User marks some docs as relevant, some as nonrelevant.
  - **Search engine computes a new representation of the information need. Hope: better than the initial query.**
  - Search engine runs new query and returns new results.
  - New results have (hopefully) better recall.
- Idea: it may be difficult to formulate a good query when you don't know the collection well, so iterate

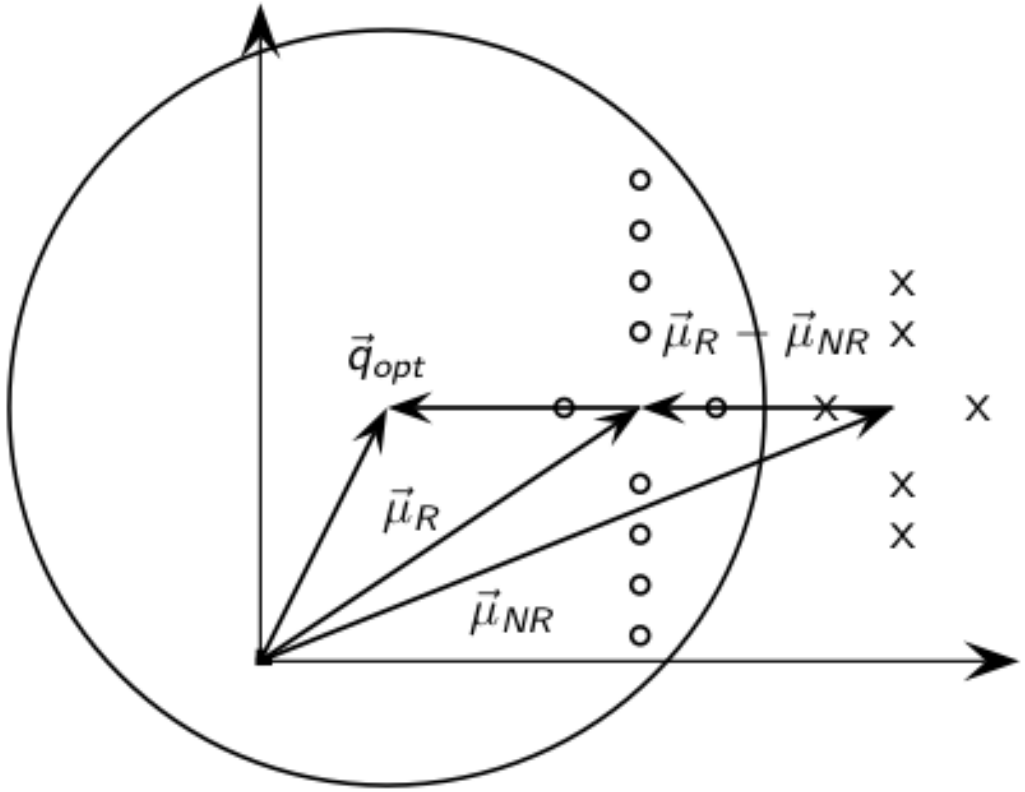
4

Pseudo-relevance feedback – Recap

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.
- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - Assume that the top *k* documents are relevant.
  - Do relevance feedback (e.g., Rocchio)
- Works very well on average
- But can go horribly wrong for some queries.
- Several iterations can cause *query drift*.

5

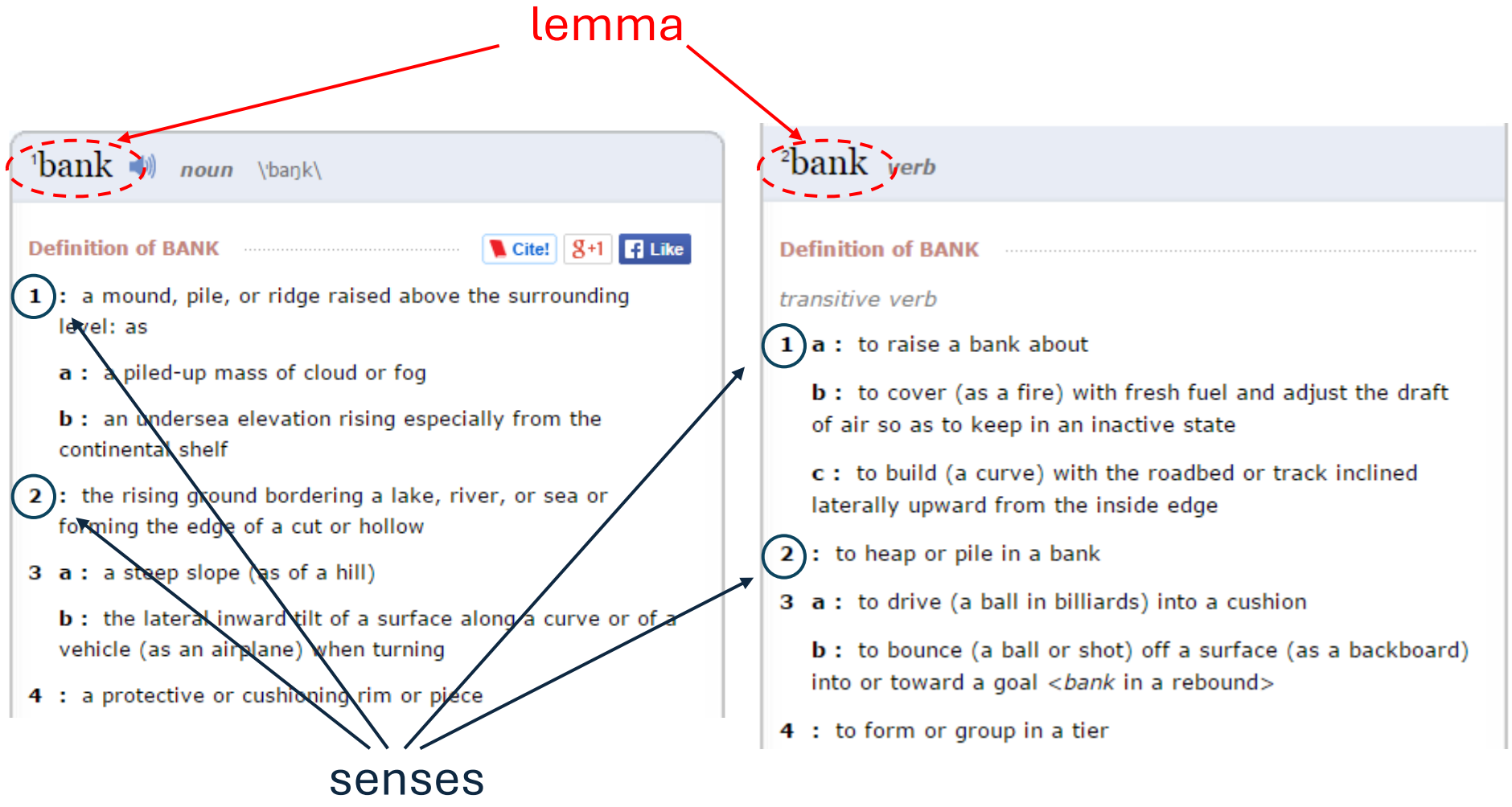
Rocchio' illustrated – Recap



$\vec{q}_{opt}$ separates relevant / nonrelevant perfectly.

6

Lexicon entries



WordNet

George Miller, Cognitive Science Laboratory of Princeton University, 1985

- A very large lexical database of English:
  - 117K nouns, 11K verbs, 22K adjectives, 4.5K adverbs
- Word senses grouped into synonym sets (“synsets”) linked into a conceptual-semantic hierarchy
  - 82K noun synsets, 13K verb synsets, 18K adjectives synsets, 3.6K adverb synsets
  - Avg. # of senses: 1.23/noun, 2.16/verb, 1.41/adj, 1.24/adverb
- Conceptual-semantic relations
  - hypernym/hyponym

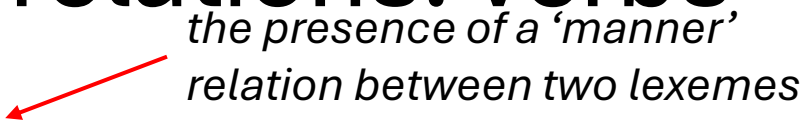
A WordNet example

- http://wordnet.princeton.edu/

Hierarchical synset relations: nouns

- **Hypernym/hyponym** (between concepts)
  - The more general ‘meal’ is a hypernym of the more specific ‘breakfast’
- **Instance hypernym/hyponym** (between concepts and instances)
  - Austen is an instance hyponym of author
- **Member holonym/meronym** (groups and members)
  - professor is a member meronym of (a university’s) faculty
- **Part holonym/meronym** (wholes and parts)
  - wheel is a part meronym of (is a part of) car.
- **Substance meronym/holonym** (substances and components)
  - flour is a substance meronym of (is made of) bread

Hierarchical synset relations: verbs



- Hypernym/troponym (between events)
  - travel/fly, walk/stroll
  - Flying is a troponym of traveling: it denotes a specific manner of traveling
- Entailment (between events):
  - snore/sleep
    - Snoring entails (presupposes) sleeping

Problems with Discrete Symbols

All vectors are **orthogonal**

No notion of similarity: words **hotel**, **motel** have nothing in common

Search Engines try to address the issue using WordNet synonyms

Still doesn't solve some problems

- E.g., Base form needed, small subset (Hebrew only around 5,500 synsets)

Alternative solution:

Encode similarity in the vector themselves

Relation: **Similarity**

Words with similar meanings. Not synonyms, but sharing some element of meaning

`car, bicycle`

`cow, horse`

Ask humans how similar 2 words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

But what about co-occurrences in large documents? A better solution

Distributional Hypothesis

- **A word's meaning is given by the words that frequently appear close-by**

  "You shall know a word by the company it keeps" (J. R. Firth 1957)
  Old idea but not quite successful until the rise of modern statistical NLP

- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window), aka **collocations**.
- Use the many contexts of w to build up a representation of w

...government debt problems turning into	**banking**	crises as happened in 2009...
...saying that Europe needs unified	**banking**	regulation to replace the hodgepodge...
..India has just given its	**banking**	system a shot in the arm...

What is Word Embedding?

- Natural language processing (NLP) models do not work with plain text. So, a numerical representation was required.
- Word embedding is a class of techniques where word is represented as a real value vectors.
- It is a representation of word in a continuous vector space.
- It is a dense representation in a vector space.
- It can be represented in smaller dimension compared to sparse representation like one-hot encoding.
- Most of the word embedding method is based on “distributional hypothesis” by Zelling Harris.

What is word embedding?

- The Distributional Hypothesis is that words that occur in the same contexts tend to have similar meanings. (Harris, 1954)
- Word embeddings are designed to capture the similarity between representation like: meaning, morphology, context etc.
- The captured relationship helps us to work on downstream NLP task like chat-bot, text summarization, information retrieval etc.
- It is generated by co-occurrence matrix, dimensionality reduction and neural networks.
- It can be broadly categorized in two parts: frequency-based embeddings and prediction-based embeddings.
- The earliest work to give a vector representation was vector space model used in information retrieval task.

Co-occurrence Matrix

One row per word with counts of co-occurrences with any other word

		Context words						
		…	cook	eat	…	changed	broke	…
Words	cake	…	10	20	…	0	0	…
	steak	…	12	22	…	0	0	…
	engine	…	0	0	…	3	10	…
	tire	…	0	0	…	10	1	…
	…	…	…	…	…	…	…	…

Pointwise Mutual Information

- **Pointwise mutual information**:
  Do events x and y co-occur more than 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)
Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

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

	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

- p(w=information, c=data) = 3982/111716 = .3399
- p(w=information) = 7703/11716 = .6575
- p(c=data) = 5673/11716 = .4842

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

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

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

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

- pmi(information, data) = .3399 / (.6575*.4842)) = .0944

Resulting PPMI matrix (negatives replaced by 0)

	computer	**data**	**result**	**pie**	**sugar**
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Curse of dimensionality

Solutions:

- Feature selection – for word vectors
- Dimensionality reduction
  - PCA – for word vectors
  - LSA
- Word Embeddings
  - Glove
  - Word2Vec

Dimensionality Reduction

What is the difference between “simple” feature selection and dimensionality reduction?

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d - k$
  - Subset selection algorithms
- Dimensionality reduction project the original x_i , i =1,…,d dimensions to new $k<d$ dimensions, z_j , j =1,…,k
  - Dimensionality reductio, represent the data in less dimensions, and losses less information
  - Principal Components Analysis (PCA) – explained later

Word-Context Matrix

A word-context (or word-word) matrix is a $|V| \times |V|$ matrix C that **counts** the frequencies of co-occurrence of words in a collection of contexts (i.e, text spans of a given length).

*You cook the **cake** twenty minutes in the oven at 220 C.*
*I eat my **steak** rare.*
*I'll throw the **steak** if you cook it too much.*
*The **engine** broke due to stress.*
*I broke a **tire** hitting a curb, I changed the **tire**.*

$\text{Context}_{-2,+2}$('cake') = {['cook', 'the', 'twenty', 'minutes']}
$\text{Context}_{-2,+2}$('tire') = {['broke', 'a', 'hitting', 'a'], ['changed', 'the']}

Sparse versus dense vectors

- 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)

Sparse versus dense vectors

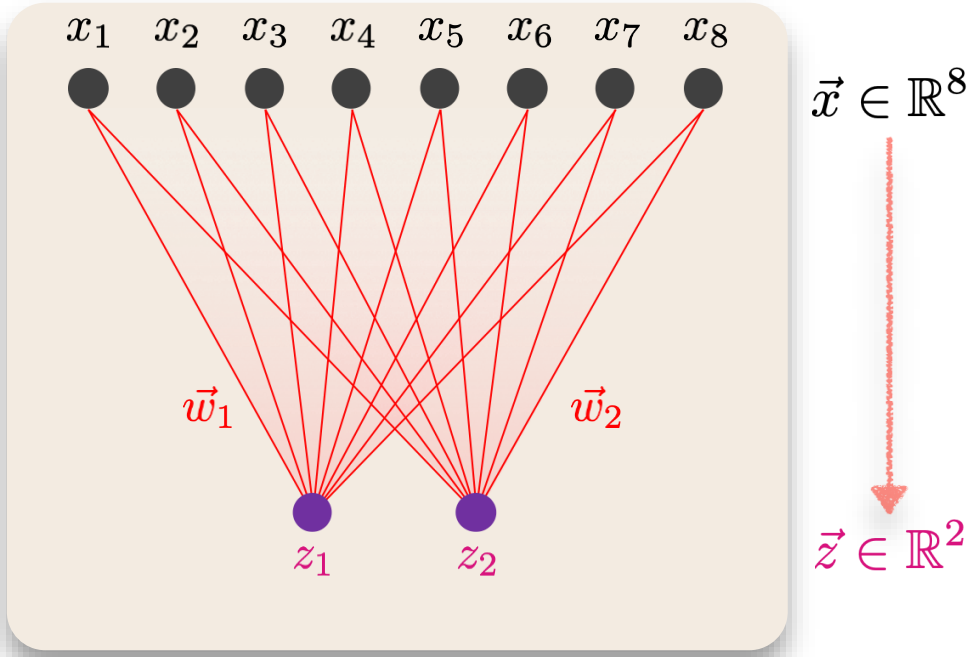
- Why dense vectors?
  - Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
  - Dense vectors may **generalize** better than explicit counts
  - Dense vectors may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are distinct dimensions
      - a word with *car* as a neighbor and a word with *automobile* as a neighbor should be similar, but aren't
  - **In practice, they work better**

Common methods for getting short dense vectors

- Singular Value Decomposition (SVD)
  - A special case of this is called LSA – Latent Semantic Analysis
- “Neural Language Model”-inspired models
  - Word2vec (skipgram, CBOW), GloVe
- Alternative to these "static embeddings":
  - Contextual Embeddings (ELMo, BERT)
  - Compute distinct embeddings for a word in its context
  - Separate embeddings for each token of a word

הורדת מימדים - PCA

- מחפשים ליצג את $\vec{x} \in \mathbb{R}^d$ באמצעות $\vec{z} \in \mathbb{R}^k$
- ע"י שימוש בקומבינציות לינאריות $\vec{w}_1, \ldots, \vec{w}_k$ של המאפיינים.



ש: איך נבחר את $\vec{w}_1, \ldots, \vec{w}_k$?

ת: שגיאת שחזור מינימלית.

PCA via SVD

$\mathrm{Cov}(\boldsymbol{x}) = \boldsymbol{\Sigma}$ - Covariance Centered data matrix x - $\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$

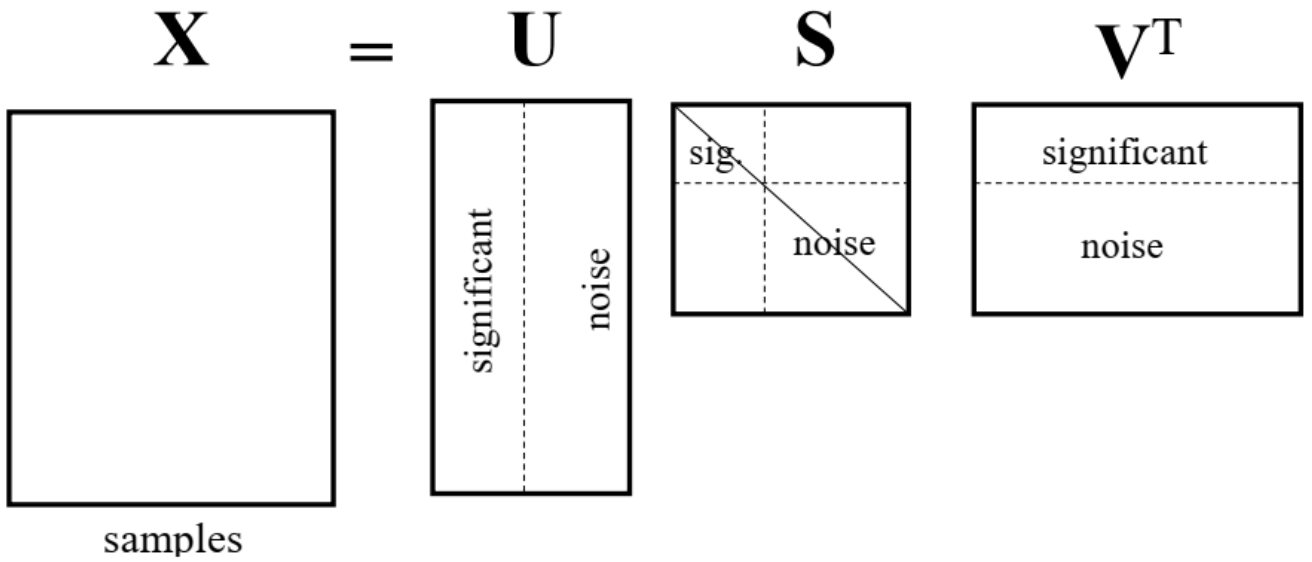
Singular Value Decomposition of the **centered** data matrix **X**.

$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m}$,

m: number of instances,

N: dimension

$$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{USV}^T$$



PCA – פעולות מרכזיות

PCA does the following:

- finds orthonormal basis for data
- Sorts dimensions in order of “importance”
- Discard low significance dimensions

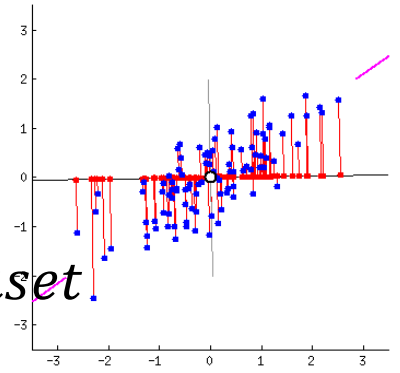
Explanations:

- Principal components – the W_i vectors
- Singular values – the coefficients of the principal components
  - higher coefficients mean more important principal components
- λ_i - eigenvalues – square of singular values

Using PCA

Notations

- Reduced dataset – Z
- W – principal components
- X^{scaled} = $standartized\ original\ dataset$



PCA Flow

- Calculate symmetric covariance matrix
- Calculate eigen vectors & eigen values (representing the variance) of the covariance matrix - **out of scope**
- Sort eigen vectors, by their eigen values
- Select *principal components* - the most significant eigen vectors

Transfer dataset in the following way:

- Z = $W^T * X^{scaled^T}$

Latent Semantic Analysis (LSA)
Basic terminology

- Symmetric matrix
  - $C = C^T$
- Rank of a matrix
  - The number of linearly independent rows (columns) in a matrix $C_{M \times N}$
  - $rank(C_{M \times N}) \leq \min(M, N)$

Latent Semantic Analysis (LSA)

- Low rank approximation of term-document matrix $C_{M\times N}$
  - Goal: remove noise in the observed term-document association data
  - Solution: find a matrix with rank k which is closest to the original matrix in terms of Frobenius norm

$$\hat{Z} = \underset{Z|rank(Z)=k}{\operatorname{argmin}} \|C - Z\|_F$$

$$= \underset{Z|rank(Z)=k}{\operatorname{argmin}} \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(C_{ij} - Z_{ij}\right)^2}$$

הורדת מימדים (dimension reduction) – מוטיבציה – דוגמה - Deriving new data

- Vector Representation

We can define a word by a vector of counts over contexts, For Example:

	song	cucumber	meal	black
tomato	**0**	**6**	**5**	**0**
book	2	0	2	3
pizza	**0**	**2**	**4**	**1**

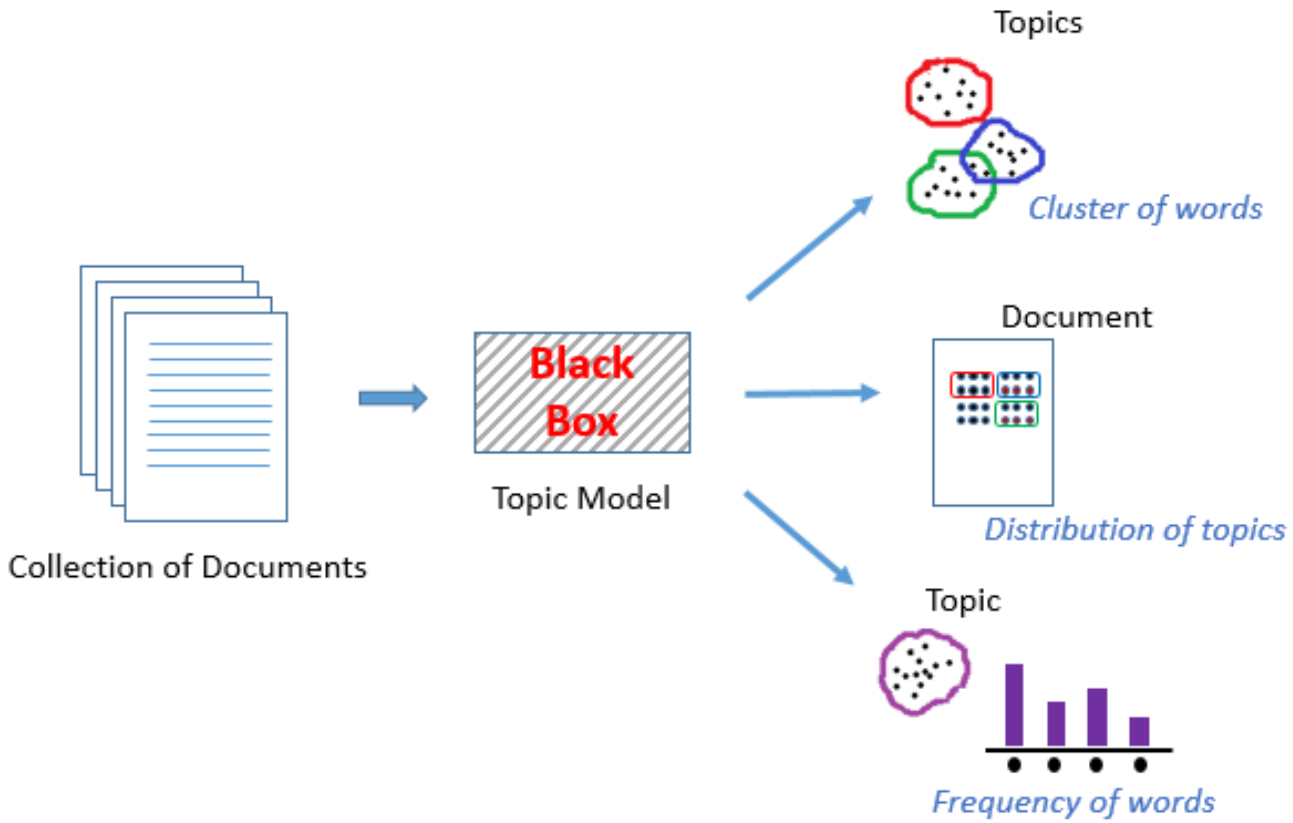
- Each word is associated with a vector of dimension |V| (the size of the vocabulary)
- We expect similar words to have similar vectors
- Given the vectors of two words, we can determine their similarity (more about that later)

These vectors are:

- huge – each of dimension |V| (the size of the vocabulary ~ $100K$ +)

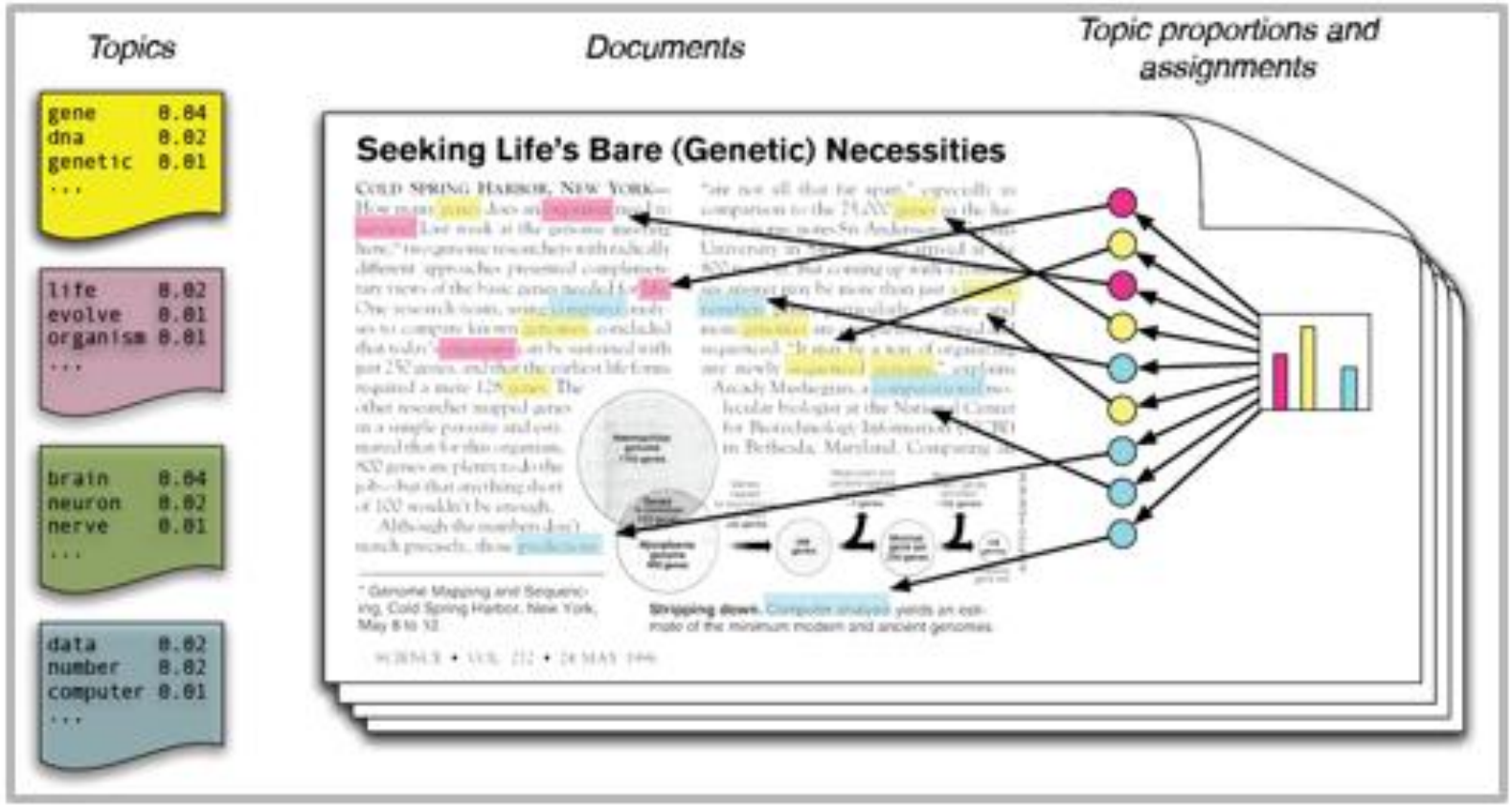
הורדת מימדים (dimension reduction) – מוטיבציה – דוגמה - Deriving new data

Example: Topic modeling (using vector representation for similarity)



הורדת מימדים (dimension reduction) – מוטיבציה – דוגמה - Deriving new data

Example: Topic modeling (using vector representation for similarity)



word vectors - LSA

LSI – SVD like PCA, but done on **term-document matrix** (rather than the covariance matrix)

In LSI we decompose the **term-document matrix** *A* into a product of three matrices: **The term matrix, The document matrix, The singular value matrix**

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

$\times$

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

term-document matrix

term matrix - one (row) vector for each term

singular value matrix – diagonal matrix with singular values, reflecting importance of each dimension

document matrix - one (column) vector for each document

LSA-A Small Example

- To see how this works let’s look at a small example
- This example is taken from: Deerwester, S.,Dumais, S.T., Landauer, T.K.,Furnas, G.W. and Harshman, R.A. (1990). "Indexing by latent semantic analysis." *Journal of the Society for Information Science, 41(6)*, 391-407.
- Slides are from a presentation by Tom Landauer and Peter Foltz

A Small Example

Technical Memo Titles

c1: *Human* machine *interface* for ABC *computer* applications

c2: A *survey* of *user* opinion of *computer system response time*

c3: The *EPS user interface* management *system*

c4: *System* and *human system* engineering testing of *EPS*

c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*

m2: The intersection *graph* of paths in *trees*

m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering

m4: *Graph minors*: A *survey*

A Small Example – 2

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

r (human.user) = -.38 r (human.minors) = -.29

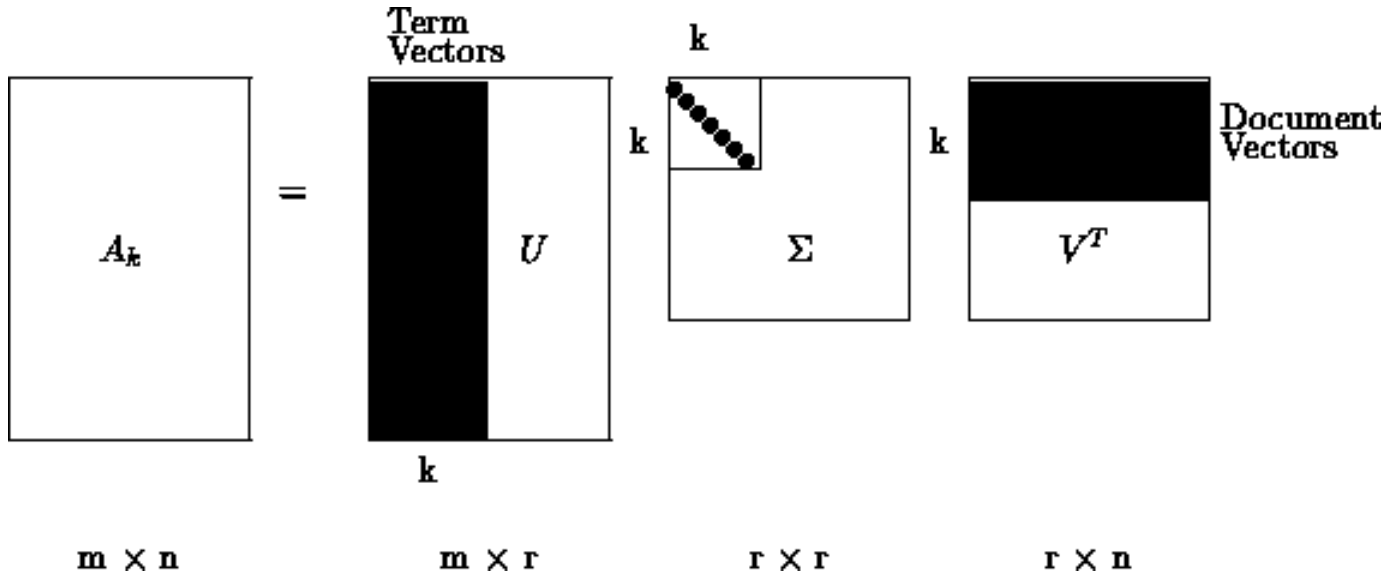
A Small Example – 3

- Singular Value Decomposition

$$\{A\}=\{U\}\{S\}\{V\}^T$$

- Dimension Reduction

$$\{\sim A\}\sim=\{\sim U\}\{\sim S\}\{\sim V\}^T$$



A Small Example – 4

- {U} =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

A Small Example – 5

- {S} =

$$\begin{bmatrix} 3.34 & & & & & & & & \\ & 2.54 & & & & & & & \\ & & 2.35 & & & & & & \\ & & & 1.64 & & & & & \\ & & & & 1.50 & & & & \\ & & & & & 1.31 & & & \\ & & & & & & 0.85 & & \\ & & & & & & & 0.56 & \\ & & & & & & & & 0.36 \end{bmatrix}$$

A Small Example – 6

- {V} =

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

A Small Example – 7

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

r (human.user) = .94 r (human.minors) = -.83

A Small Example – 2 reprise

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

r (human.user) = -.38 r (human.minors) = -.29

Correlation

Raw data

	c1	*c2*	*c3*	*c4*	*c5*	*m 1*	*m 2*	*m 3*
c2	- 019							
c3	0.00	0.00						
c4	0.00	0.00	0.47					
c5	- 033	0.58	0.00	- 031				
m 1	- 017	- 030	- 021	- 016	- 017			
m 2	- 026	- 045	- 032	- 024	- 026	0.67		
m 3	- 033	- 058	- 041	- 031	- 033	0.52	0.77	
m 4	- 033	- 019	- 041	- 031	- 033	- 017	0.26	0.56

0.02	
- 030	**0.44**

Correlations in first-two dimension space

c2	0.91							
c3	1.00	0.91						
c4	1.00	0.88	1.00					
c5	0.85	0.99	0.85	0.81				
m 1	- 085	- 056	- 085	- 088	- 045			
m 2	- 085	- 056	- 085	- 088	- 044	1.00		
m 3	- 085	- 056	- 085	- 088	- 044	1.00	1.00	
m 4	- 081	- 050	- 081	- 084	- 037	1.00	1.00	1.00

0.92	
-0.72	1.00

A Small Example

- A note about notation:
  - Here we called our matrices
    - $\{A\}=\{U\}\{S\}\{V\}^T$
  - You may also see them called
    - $\{W\}\{S\}\{P\}^T$
    - $\{T\}\{S\}\{D\}^T$
  - The last one is easy to remember
    - T = term
    - S = singular
    - D = document

Summary

- Has proved to be a valuable tool in many areas of NLP as well as IR
  - summarization
  - cross-language IR
  - topics segmentation
  - text classification
  - question answering
  - more

Common methods for getting short dense vectors

- Singular Value Decomposition (SVD)
  - A special case of this is called LSA – Latent Semantic Analysis
- “Neural Language Model”-inspired models
  - Word2vec (skipgram, CBOW), GloVe
- Alternative to these "static embeddings":
  - Contextual Embeddings (ELMo, BERT)
  - Compute distinct embeddings for a word in its context
  - Separate embeddings for each token of a word

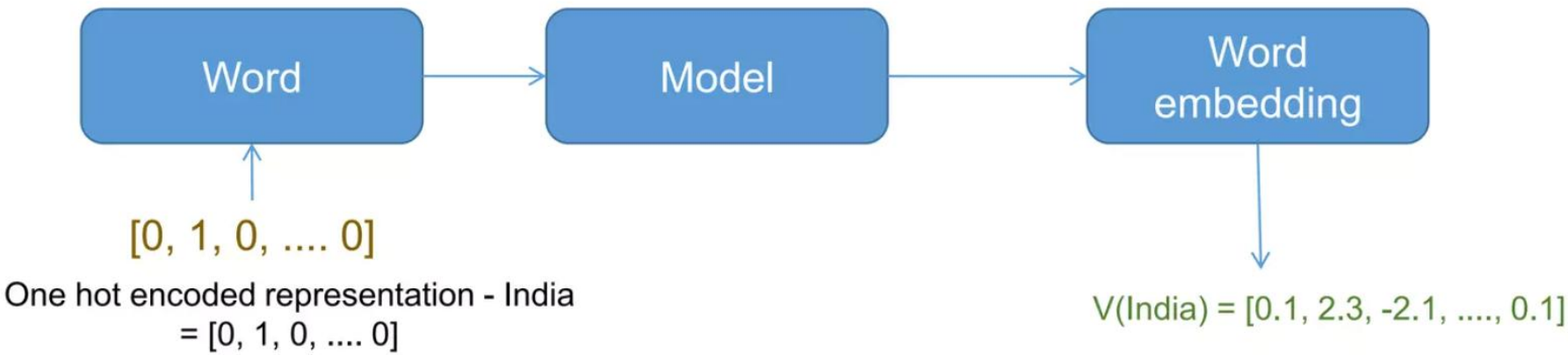
Simple static embeddings you can download!

- Word2vec (Mikolov et al)
- https://code.google.com/archive/p/word2vec/

- GloVe (Pennington, Socher, Manning)
- http://nlp.stanford.edu/projects/glove/

Prediction based word embeddings

- It is a method to learn dense representation of word from a very high dimensional representation.



- It is a modular representation, where a sparse vector is fed to generate a dense representation

Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: **predict** rather than **count**
- Word2vec provides various options. We'll do:
- **skip-gram with negative sampling (SGNS)**
-
-

Word2Vec

- It was proposed by Mikolov et.al. in 2013.
- It is a two layer shallow neural network trained to learn the contextual relationship.
- It places contextually similar word near to each other.
- It is a co-occurrence based model.
- Two variants of the model was proposed
  - Continuous bag of words model (CBOW)
    - Given the context word, predict the center word
    - Order of context words are not considered, so this representation is similar to BOW.
  - Skip-gram model

What does context mean?

- Context is co-occurrence of the words. It is a sliding window around the word under the consideration.

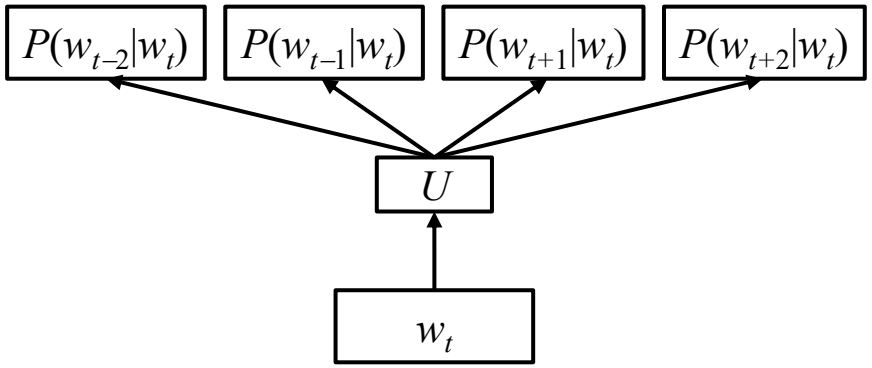
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state
India	is	now	inching	towards	a	self	reliant	state

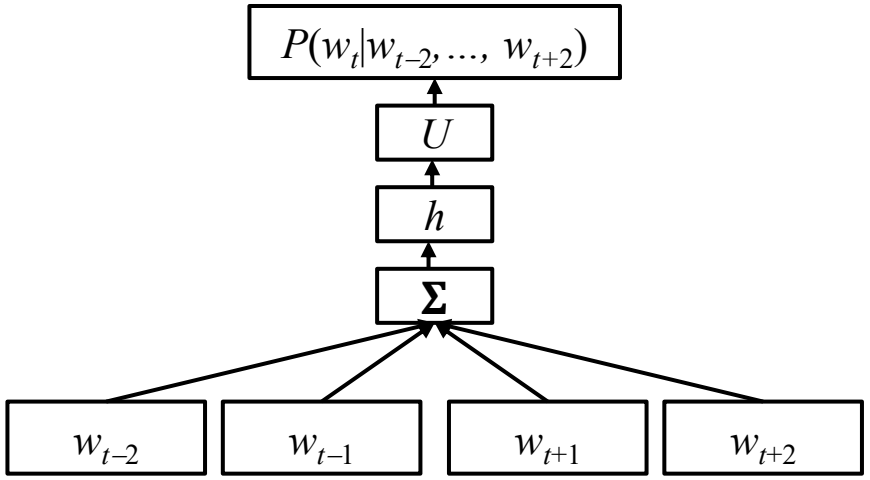
Window size = 2, Yellow patches are words are in consideration, orange box are the context window

Word2Vec Models

Skip-gram: **predict context** words within window of width m, surrounding center word w_t

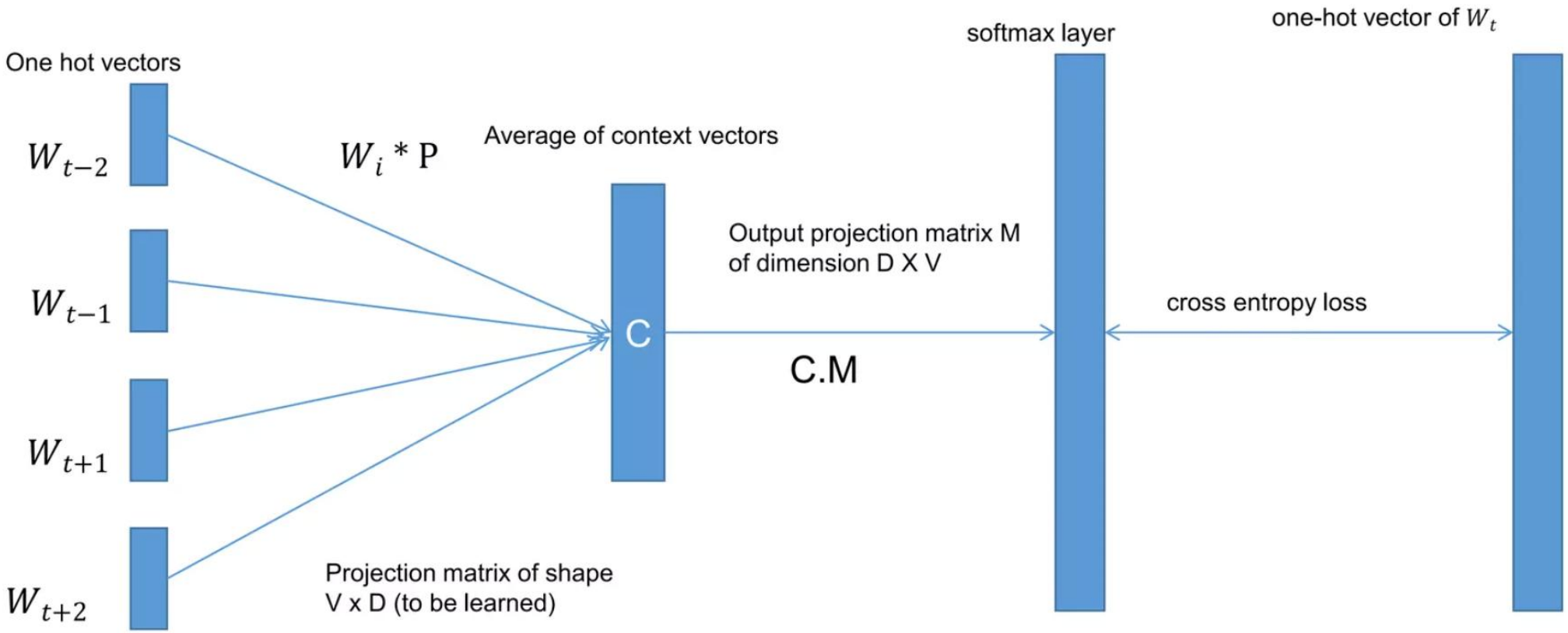
CBoW: **predict center** word w_t given context words within window of width m





CBOW

- Goal: Predict the center word, given the context words.

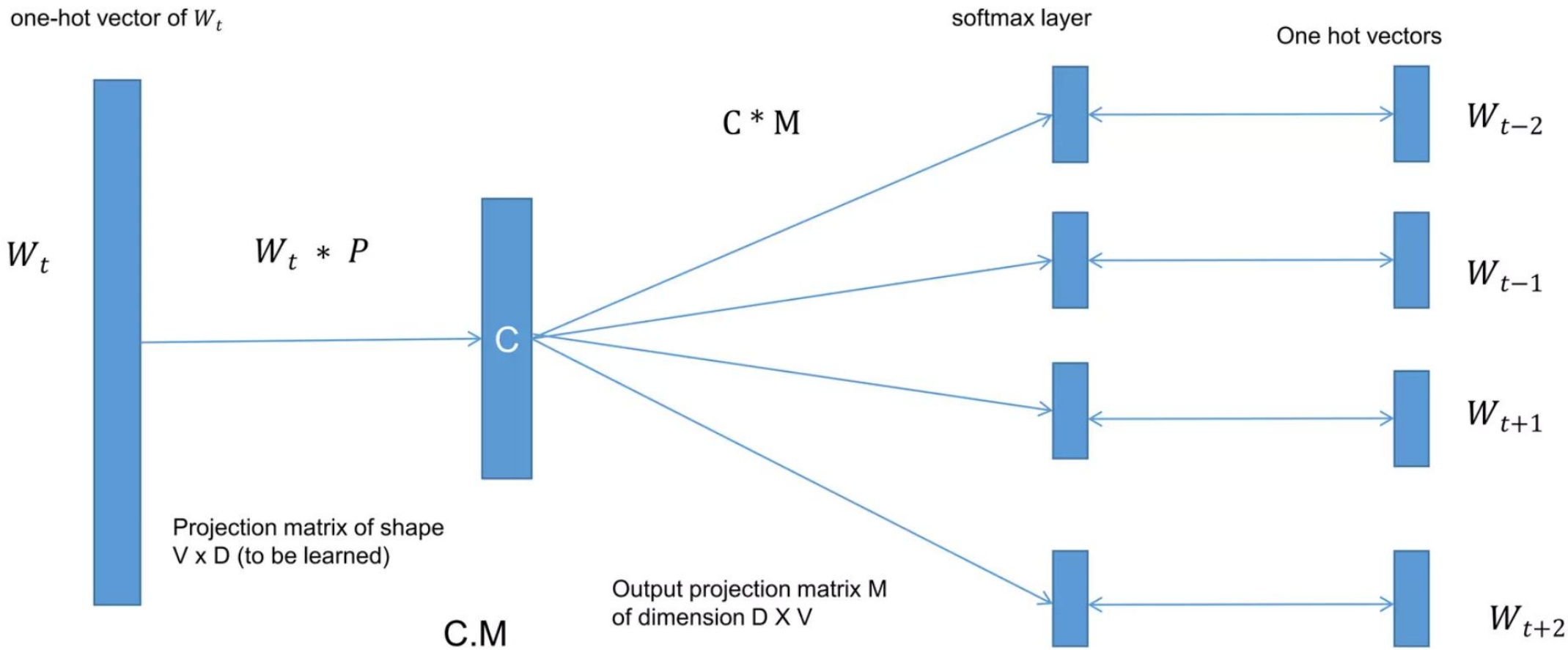


CBOW

- One hot encoded of the context words W_{t-2} , ... W_{t+2} is input to the model.
- Projection matrix of shape V x D, where is V is the total no of unique words in the corpus and D is dimension of the dense representation, to project one-hot encoded vector into D-dimension vector.
- Averaged context vector is projected back to V-dimensional space. Softmax layer converts the representation into proablities for W_t.
- The model is trained using cross-entropy loss between the softmax layer output and the one-hot encoded representation of W_t.

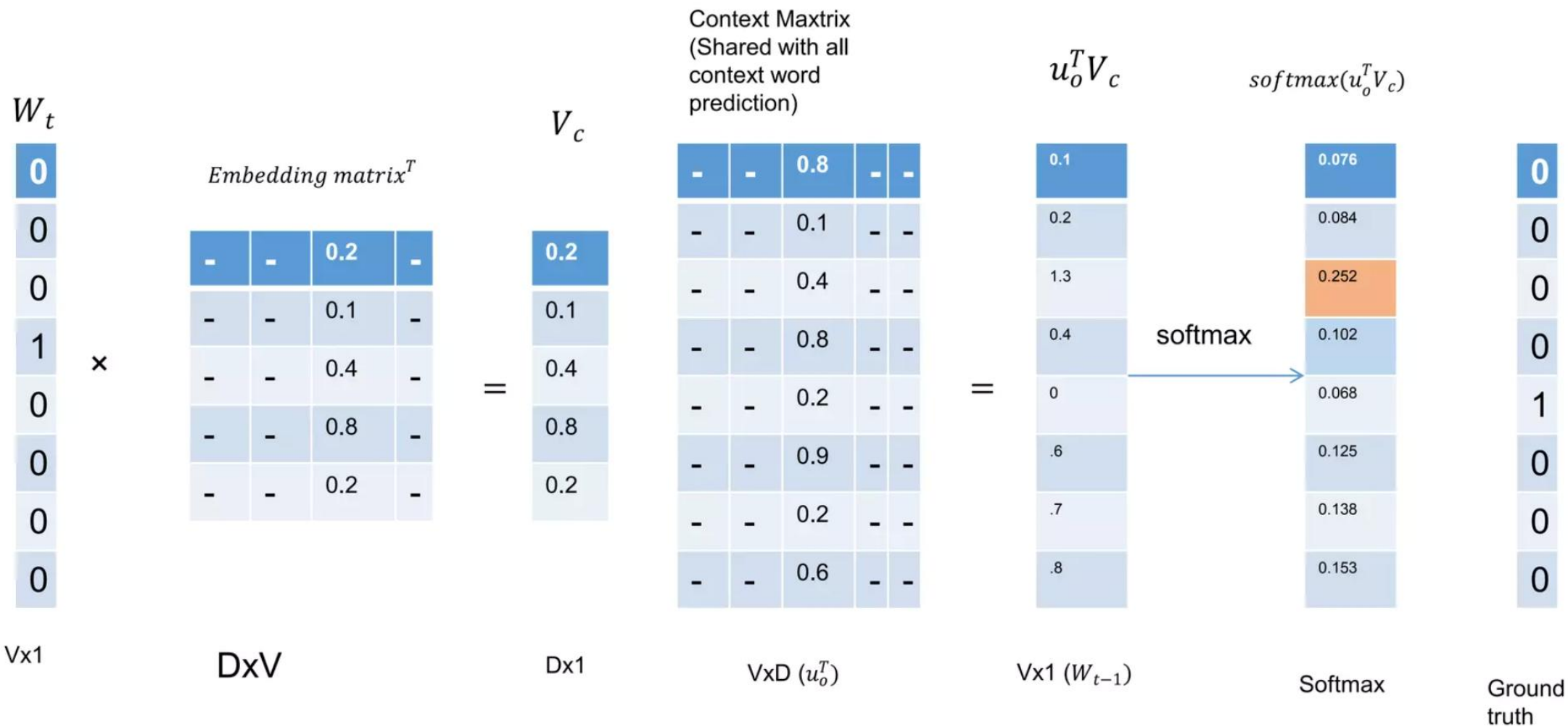
Skip-gram model: High level

- Goal: To predict the context word W_{t-2} , …. W_{t+2} given the word W_t



Skip-gram

- A end to end flow of training:



Skip-gram

- It focus on optimization of loss for each word:
  - $P(o|c) = \frac{\exp(u_0^T v_c)}{\sum_{k=1}^{V} \exp(u_w^T v_c)}$
  - It calculates probability of output context word given the center word c.
  - The loss function which it tries to minimize is:

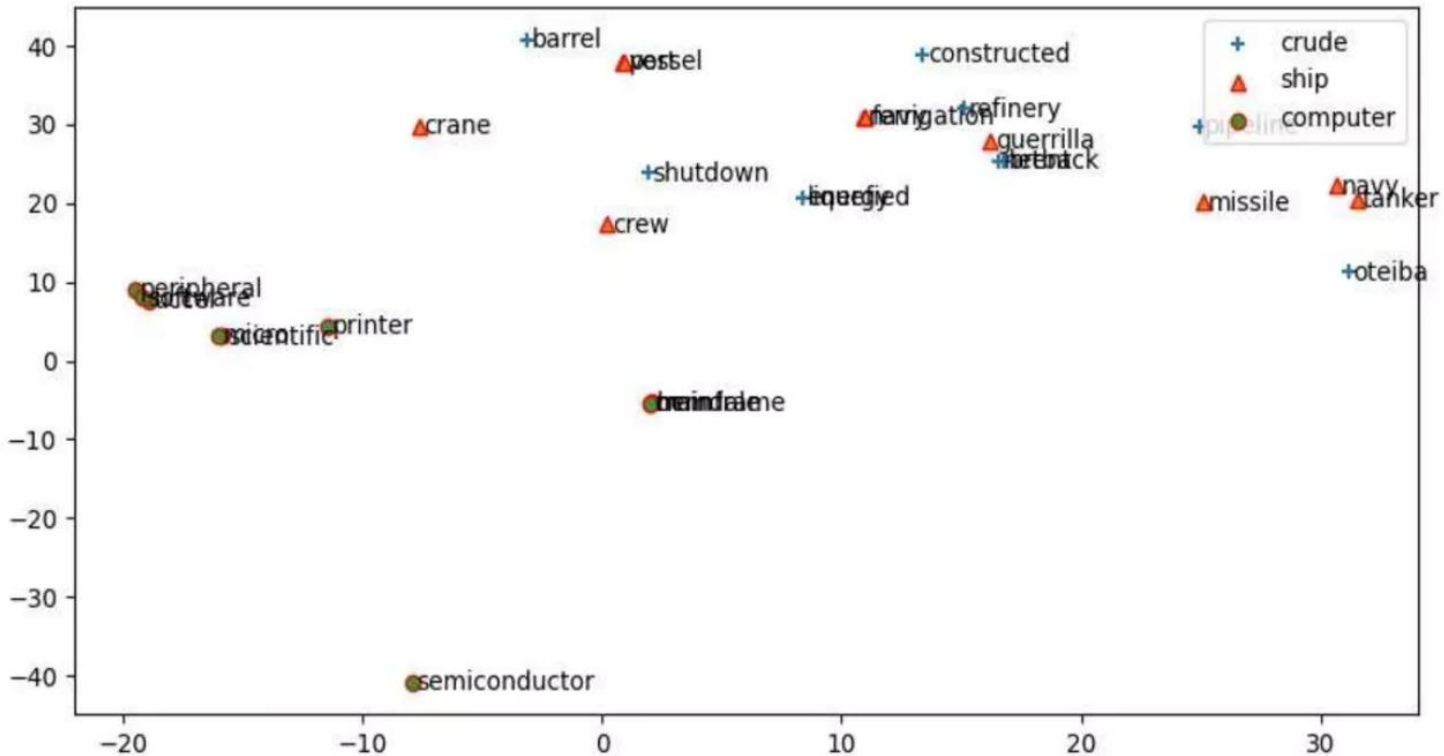
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \neq 0}} logp(W_{t+j}/W_t)$$

  - Log value is calculated similar to depicted in first equation.
  - Naive Training is costly because gradient calculation is of order $|V|$
- Two computationally efficient methods are proposed:
  - Hierarchical Softmax
  - Negative sampling

Skip-gram

- Use of negative sampling is to train is more prevalent.
- In early example, tuple like (India, the), (India, now) are the example to true cases.
- Any corrupted tuple is called as negative sample. like (India, reliant), (India, state)
- This process with modified objective function results it in a logistic regression to classify a tuple as a true combination or a corrupt ones.
- The corrupt tuple is generated by sampling such that less frequent words are picked up more often as a corrupt tuple.

Word Embedding visualization



t-SNE 2. D projection of Word2vec (gensim implementation) embeddings of top 10 similar words, trained for 50 epoch on Reuters news corpus from NLTK, with context len 15, vector dimension 100

Top 5 similar words

crude barrel 0.548
crude oteiba 0.464
crude netback 0.45
crude refinery 0.438
crude pipeline 0.421
---
ship vessel 0.623
ship port 0.575
ship tanker 0.496
ship navigation 0.471
ship crane 0.463
---
computer software 0.602
computer micro 0.559
computer printer 0.542
computer mainframe 0.538
computer hemdale 0.527

- Even with a smaller corpus it can capture semanticallly relevant words.

Word2vec results

- The top 5 similar words when:

Context length is 30

---

crude barrel 0.475
crude refinery 0.438
crude stockdraws 0.427
crude yates 0.408
crude utilized 0.382

---

ship vessel 0.557
ship tanker 0.506
ship port 0.5
ship icebreaker 0.461
ship loaded 0.453

---

computer software 0.569
computer micro 0.517
computer memory 0.498
computer disk 0.495
computer printer 0.476

Embedding dimension is 300

---

crude refinery 0.27
crude stockdraws 0.254
crude barrel 0.244
crude utilized 0.242
crude liquefied 0.239

---

ship vessel 0.468
ship crew 0.318
ship tanker 0.308
ship shipbuilder 0.302
ship yard 0.288

---

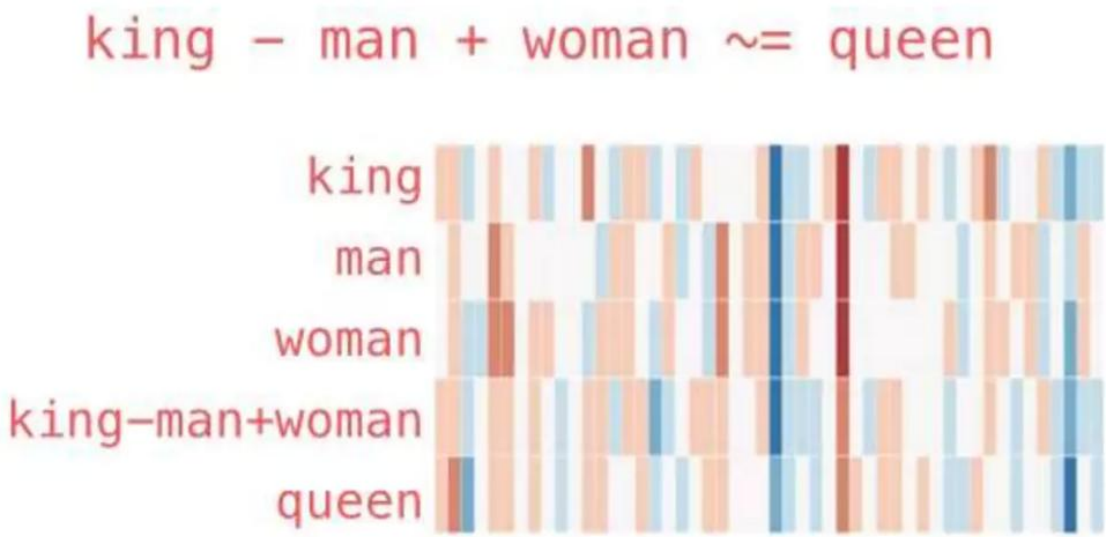
computer software 0.441
computer disk 0.345
computer printer 0.345
computer uccel 0.338
computer scientific 0.335

Analogies

- Representation of analogy in vector space using word2vec vectors:

  - Vector representation of “King-man+woman” is roughly equivalent to the vector representation of queen
  - Using gensim and pretrained word2vec the analogy vector generated for “King-man+woman” 5 most closer relationships are

queen	0.7118
monarch	0.619
princess	0.5902
crown_prince	0.55
prince	0.54



Limitations: Word Ordering or N-Grams

“it was not good, it was actually quite bad”

== or !=

“it was not bad, it was actually quite good”

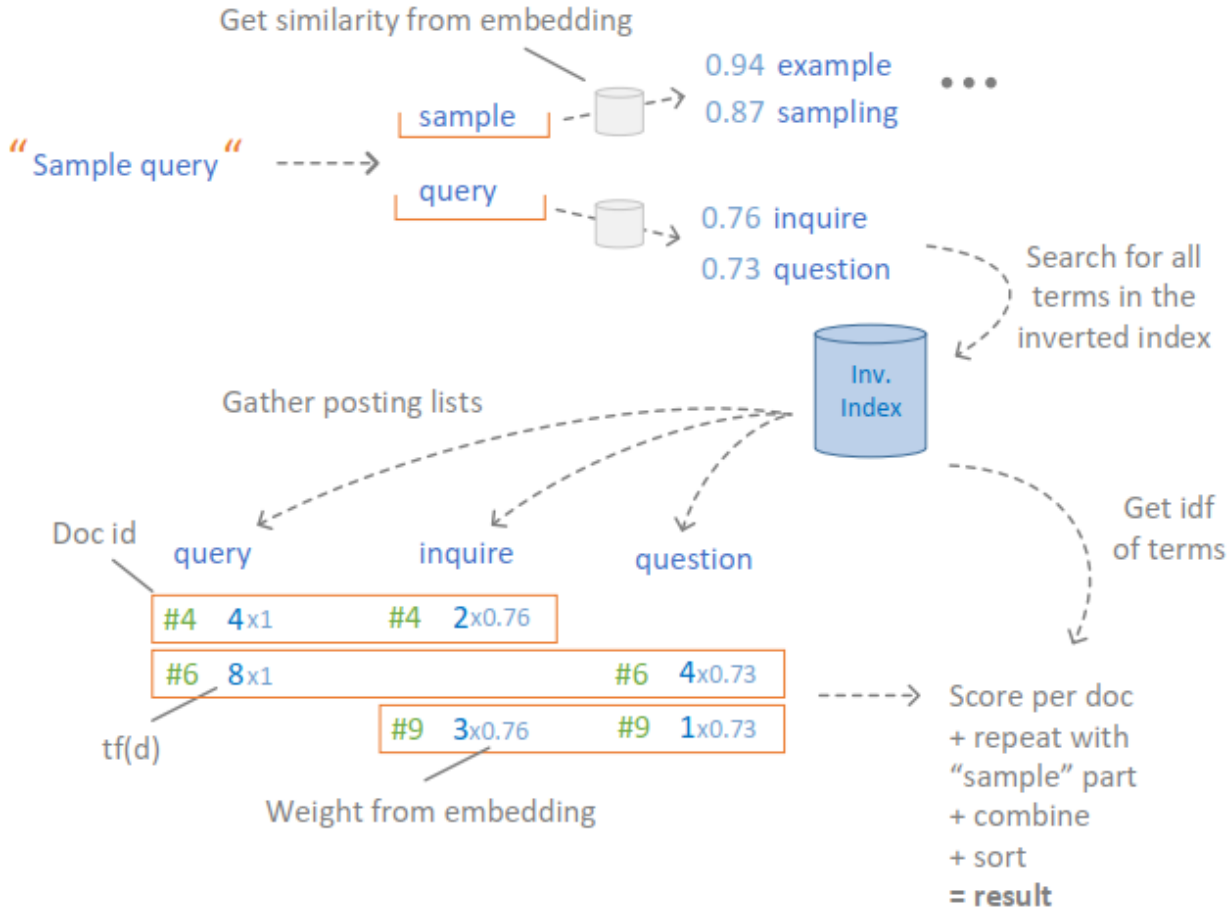
- The ordering & local context is important: “not good” vs. “not bad”
- Looking at N words at a time is called N-gram
- Creating bi-gram (2) or tri-gram (3) embeddings is not feasible
- Sparsity problem
- Not enough training data: no connection between “quite good” and “very good

Limitations: Multiple Senses per Word

- Word2Vec, Glove & FastText always map 1 word to 1 vector
  - This is good for analysis purposes and constrained resource environments
- There is no contextualization in the vector after training
  - The vector of 1 word does not change based on other words in the sentence around it
  - Words with multiple senses depending on context are squashed together in an average or most common sense in the training data
- But many words do have many senses based on the context
  - Missed opportunity for improved effectiveness

Query Expansion with Word Embeddings

- Expand the search space of a search query with similar words
- Update collection statistics
- Adapted relevance model to score 1 document with multiple similar words together

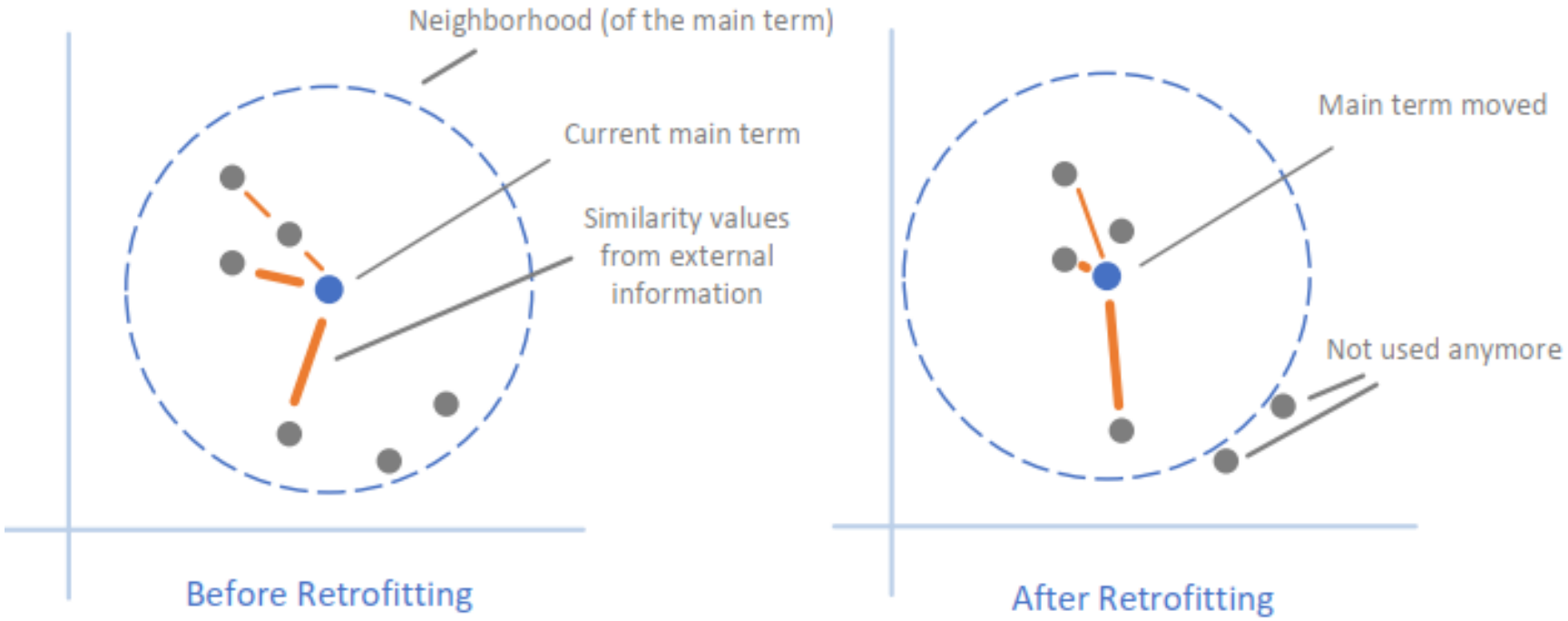


Problem: Topic Shifting

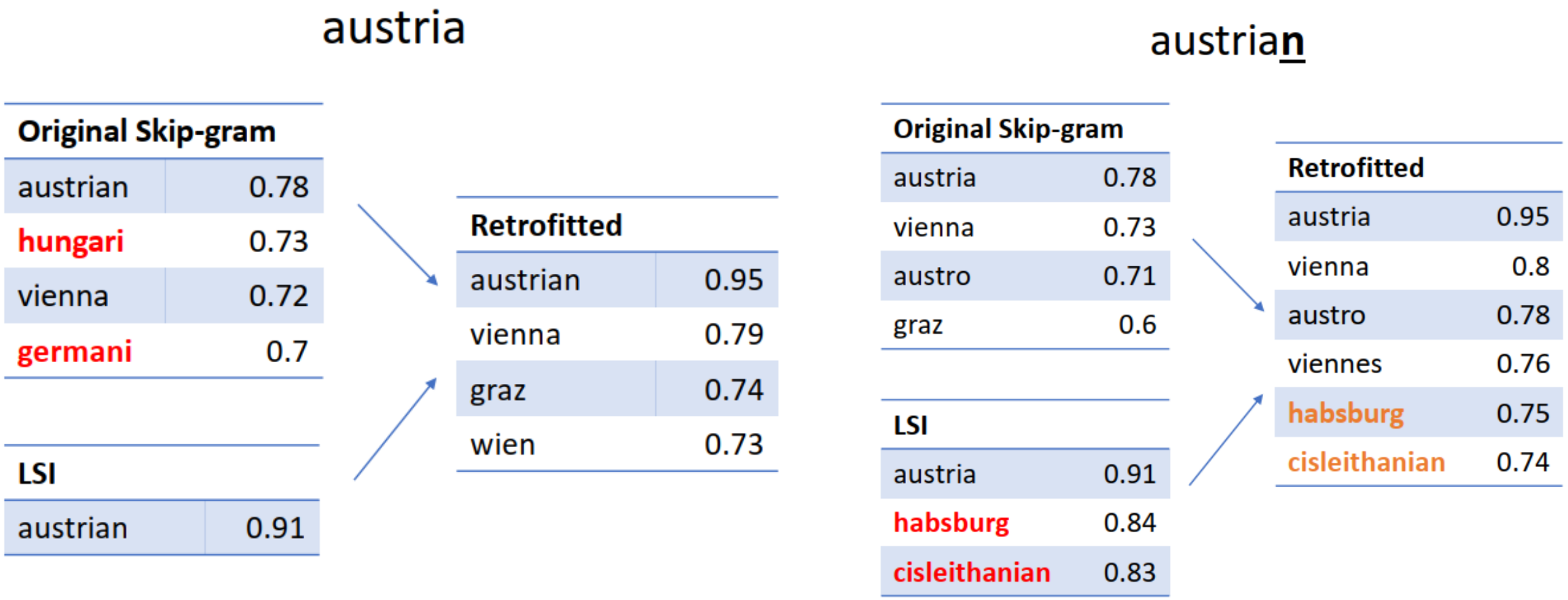
- Word Embeddings trained on Wikipedia context
- Words with similar context close together
  - Can be a different topic -> bad for IR
- Word “Austria” has close neighbors that correspond to physical
- neighboring countries
  - Germany, Hungary, …
  - Searching for “Hotels in Austria” -> you probably only want results located in “Austria” not the 25 best hotels in Germa

Retrofitting

- Incorporate external resources into an existing word embedding
- Moves the vectors of the existing word embedding
- Iterative update function



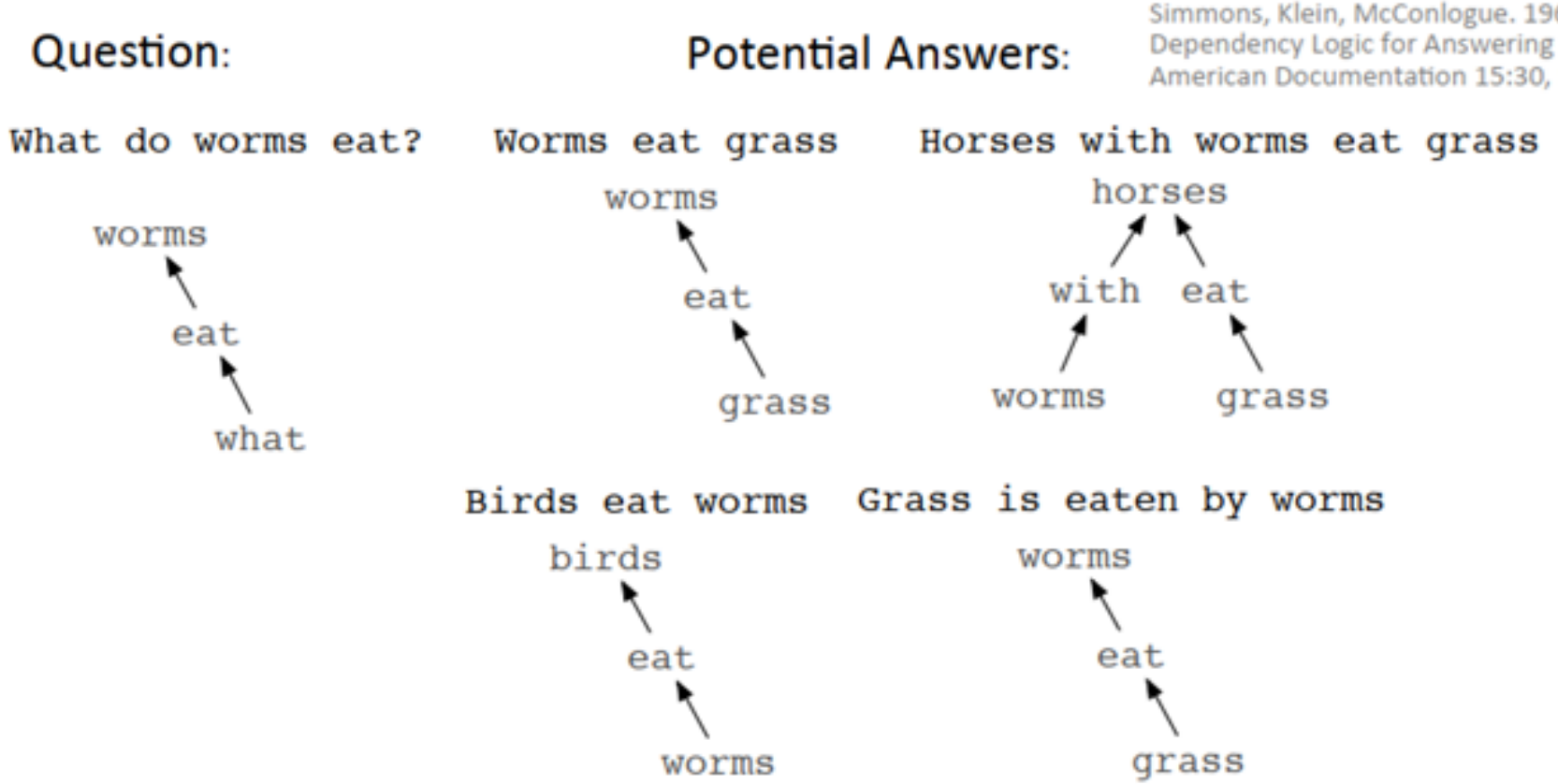
Topic Shifting – Examples from Wikipedia



Question Answering

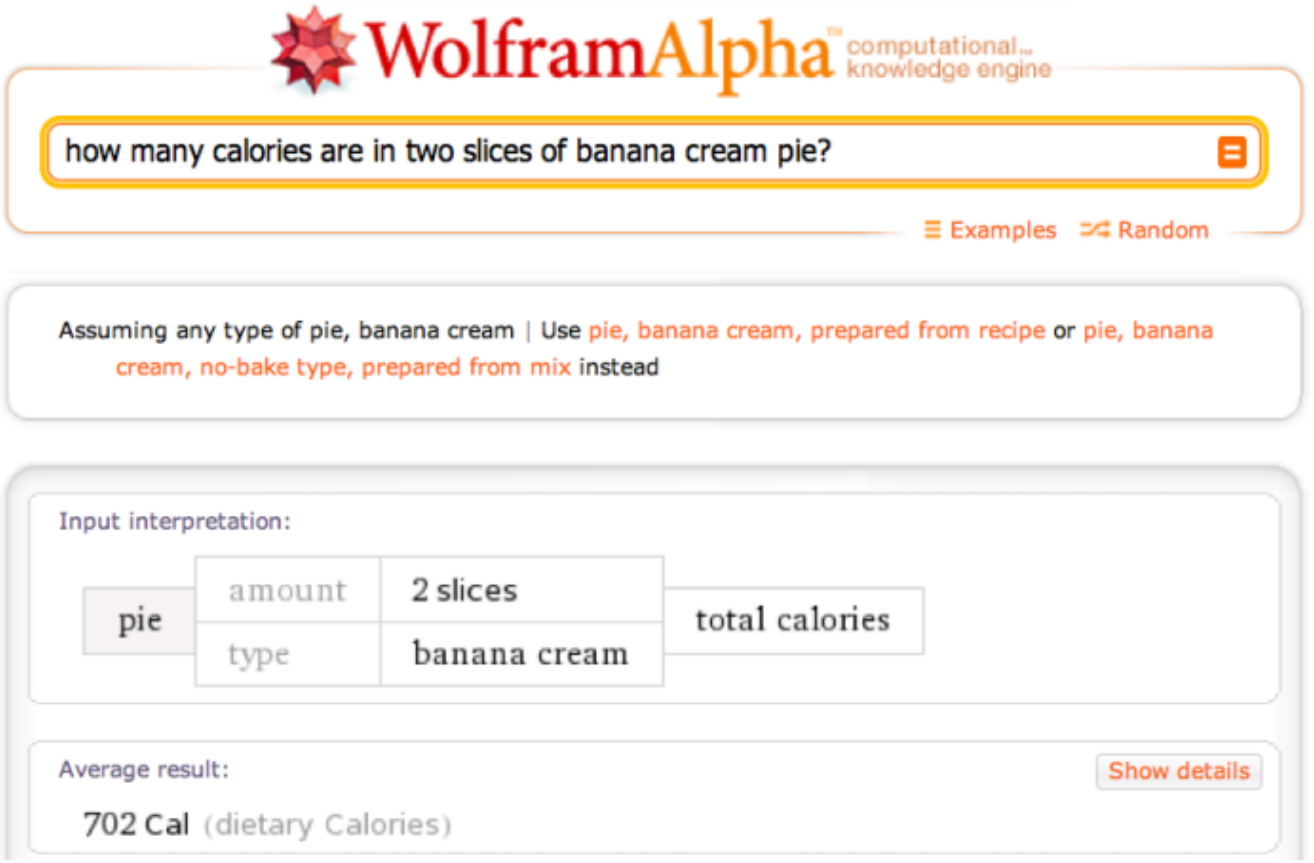
- One of the oldest NLP tasks (punched card systems in 1961)

Simmons, Klein, McConlogue. 1964. Indexing and Dependency Logic for Answering English Questions. American Documentation 15:30, 196-204



Question Answering - applications

Apple's Siri



Types of Questions in Modern Systems

- Factoid questions
  - *Who wrote "The Universal Declara4on of Human Rights"?*
  - *How many calories are there in two slices of apple pie?*
  - *What is the average age of the onset of au4sm?*
  - *Where is Apple Computer based?*
- Complex (narrative) questions:
  - *In children with an acute febrile illness, what is the efficacy of acetaminophen in reducing fever?*
  - *What do scholars think about Jefferson's position on dealing with pirates?*

Paradigms for QA

- IR-based approaches
  - TREC; IBM Watson; Google

- Knowledge--based and Hybrid approaches
  - TBD

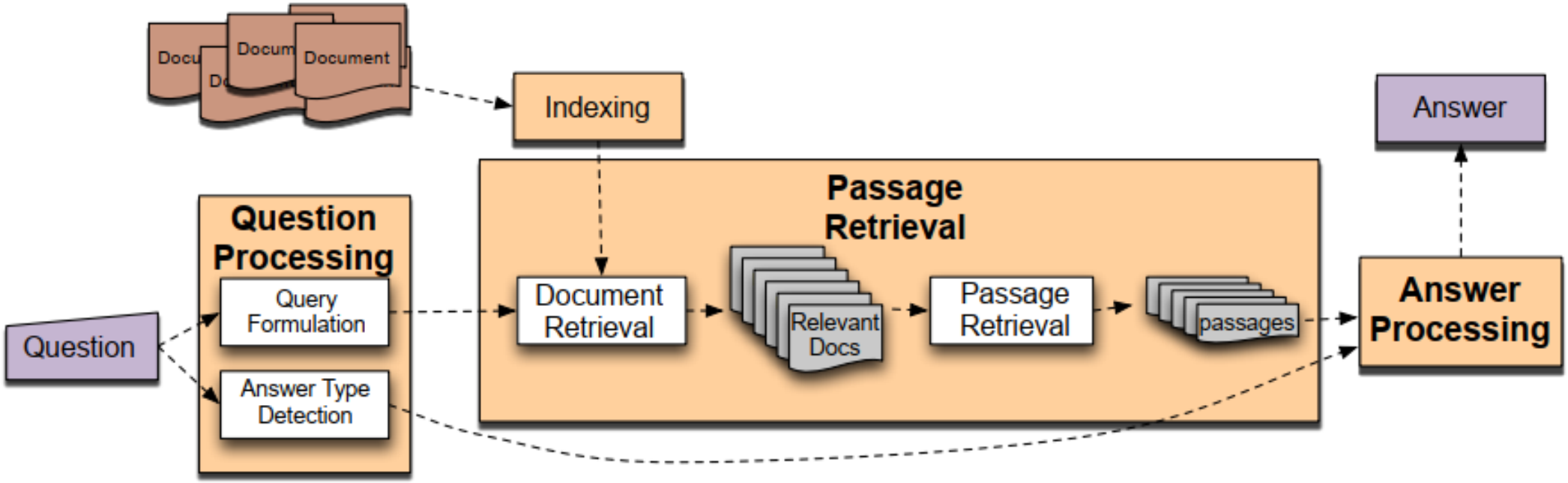
IR-based Question Answering

IR-based Factoid QA

- QUESTION PROCESSING
  - Detect question type, answer type, focus, relations
  - Formulate queries to send to a search engine
- PASSAGE RETRIEVAL
  - Retrieve ranked documents
  - Break into suitable passages and rerank
- ANSWER PROCESSING
  - Extract candidate answers
  - Rank candidates
  - using evidence from the text and external sources

Dan Jurafsky

IR-based Factoid QA



Question Processing
Things to extract from the question

- Answer Type Detection
  - Decide the **named entity type** (person, place) of the answer
- Query Formulation
  - Choose **query keywords** for the IR system
- Question Type classification
  - Is this a definition question, a math question, a list questions
- Focus Detection
  - Find the question words that are replaced by the answer
- Relation Extraction
  - Find relations between entities in the question

Answer Type Detection: Named Entities

- *Who founded Virgin Airlines?*
  - PERSON
- *What Canadian city has the largest population?*
  - *CITY.*

Answer Type Taxonomy

Xin Li, Dan Roth. 2002. Learning Question Classifiers. COLING'02

- 6 coarse classes
  - ABBEVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, NUMERIC
- 50 finer classes
  - LOCATION: city, country, mountain…
  - HUMAN: group, individual, title, description
  - ENTITY: animal, body, color, currency…

Answer Types

ENTITY	
animal	What are the names of Odin's ravens?
body	What part of your body contains the corpus callosum?
color	What colors make up a rainbow ?
creative	In what book can I find the story of Aladdin?
currency	What currency is used in China?
disease/medicine	What does Salk vaccine prevent?
event	What war involved the battle of Chapultepec?
food	What kind of nuts are used in marzipan?
instrument	What instrument does Max Roach play?
lang	What's the official language of Algeria?
letter	What letter appears on the cold-water tap in Spain?
other	What is the name of King Arthur's sword?
plant	What are some fragrant white climbing roses?
product	What is the fastest computer?
religion	What religion has the most members?
sport	What was the name of the ball game played by the Mayans?
substance	What fuel do airplanes use?
symbol	What is the chemical symbol for nitrogen?
technique	What is the best way to remove wallpaper?
term	How do you say " Grandma " in Irish?
vehicle	What was the name of Captain Bligh's ship?
word	What's the singular of dice?

Answer Type Detection

- Regular expression-based rules can get some cases:
  - Who {is|was|are|were} PERSON
  - PERSON (YEAR – YEAR)
- Other rules use the **question headword**:

  (the headword of the first noun phrase after the wh-word)

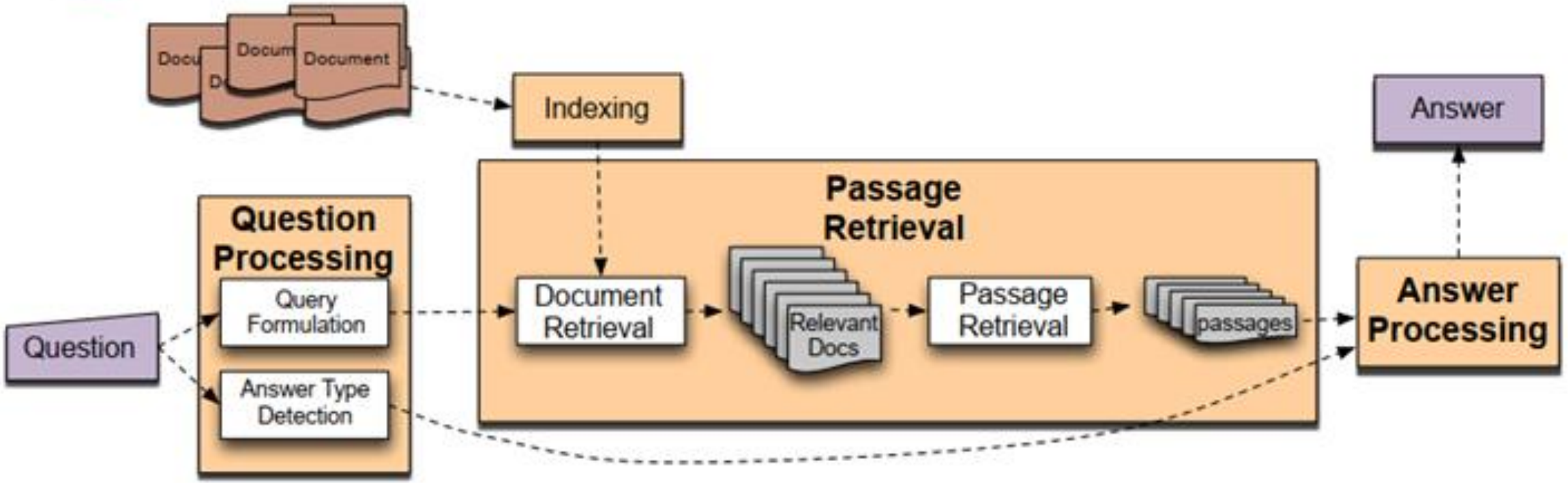
  - Which **city** in China has the largest number of foreign financial companies?
  - What is the state **flower** of California?

Answer Type Detection

- Most often, we treat the problem as machine learning classification
  - **Define** a taxonomy of question types
  - **Annotate** training data for each question type
  - **Train** classifiers for each question class using a rich set of features.
    - features include those hand-written rules!

Dan Jurafsky

IR-based Factoid QA



Keyword Selection Algorithm

Dan Moldovan, Sanda Harabagiu, Marius Paca, Rada Mihalcea, Richard Goodrum, Roxana Girju and Vasile Rus. 1999. Proceedings of TREC-8.

1. Select all non-stop words in quotations
2. Select all NNP words in recognized named entities
3. Select all complex nominals with their adjectival modifiers
4. Select all other complex nominals
5. Select all nouns with their adjectival modifiers
6. Select all other nouns
7. Select all verbs
8. Select all adverbs
9. Select the QFW word (skipped in all previous steps)
10. Select all other words

Dan Jurafsky

Choosing keywords from the query

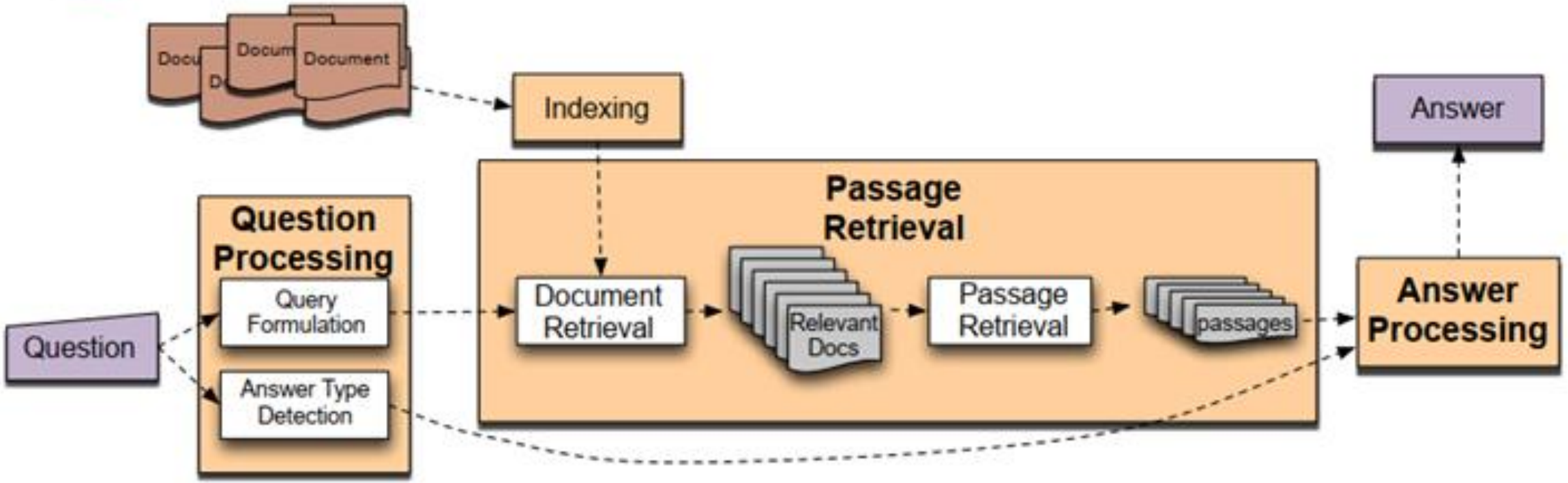
Slide from Mihai Surdeanu

Who coined the term "cyberspace" in his novel "Neuromancer"?

7 4 1 4 1

cyberspace/1 Neuromancer/1 term/4 novel/4 coined/7

IR-based Factoid QA



Passage Retrieval

- Step 1: IR engine retrieves documents using query terms
- Step 2: Segment the documents into shorter units
  - something like paragraphs
- Step 3: Passage ranking
  - Use answer type to help rerank passages

Dan Jurafsky

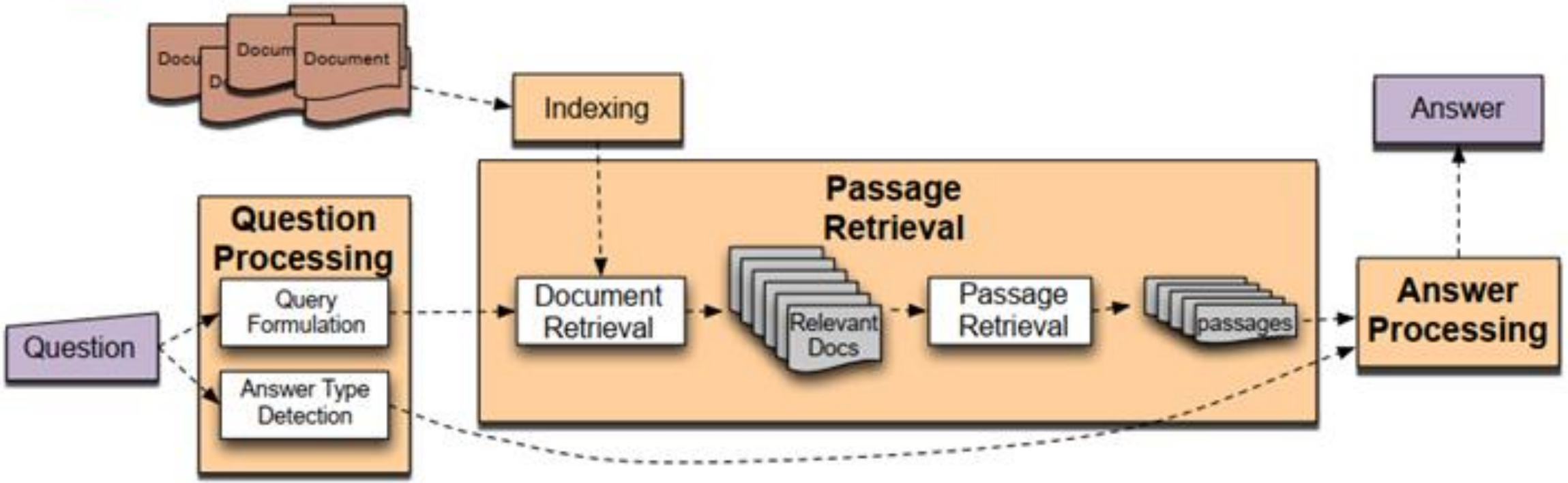
Features for Passage Ranking

Either in rule-based classifiers or with supervised machine learning

- Number of Named Entities of the right type in passage
- Number of query words in passage
- Number of question N-grams also in passage
- Proximity of query keywords to each other in passage
- Longest sequence of question words
- Rank of the document containing passage

Dan Jurafsky

IR-based Factoid QA



Answer Extraction

- Run an answer-type named-entity tagger on the passages
  - Each answer type requires a named-entity tagger that detects it
  - If answer type is CITY, tagger has to tag CITY
    - Can be full NER, simple regular expressions, or hybrid
- Return the string with the right type:
  - Who is the prime minister of India (PERSON)

Manmohan Singh, Prime Minister of India, had told left leaders that the deal would not be renegotiated.

  - How tall is Mt. Everest? (LENGTH)

The official height of Mount Everest is **29035 feet**

Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: **Person**

- Passage:

The Marie biscuit is named after Marie Alexandrovna, the daughter of Czar Alexander II of Russia and wife of Alfred, the second son of Queen Victoria and Prince Albert

Ranking Candidate Answers

- But what if there are multiple candidate answers!

Q: Who was Queen Victoria's second son?

- Answer Type: **Person**

- Passage:

The Marie biscuit is named after **Marie Alexandrovna**, the daughter of **Czar Alexander II of Russia** and wife of **Alfred**, the second son of **Queen Victoria** and **Prince Albert**

Dan Jurafsky

Use machine learning: Features for ranking candidate answers

Answer type match: Candidate contains a phrase with the correct answer type.

Pattern match: Regular expression pattern matches the candidate.

Question keywords: # of question keywords in the candidate.

Keyword distance: Distance in words between the candidate and query keywords

Novelty factor: A word in the candidate is not in the query.

Apposition features: The candidate is an appositive to question terms

Punctuation location: The candidate is immediately followed by a comma, period, quotation marks, semicolon, or exclamation mark.

Sequences of question terms: The length of the longest sequence of question terms that occurs in the candidate answer.

Dan Jurafsky

Candidate Answer scoring in IBM Watson

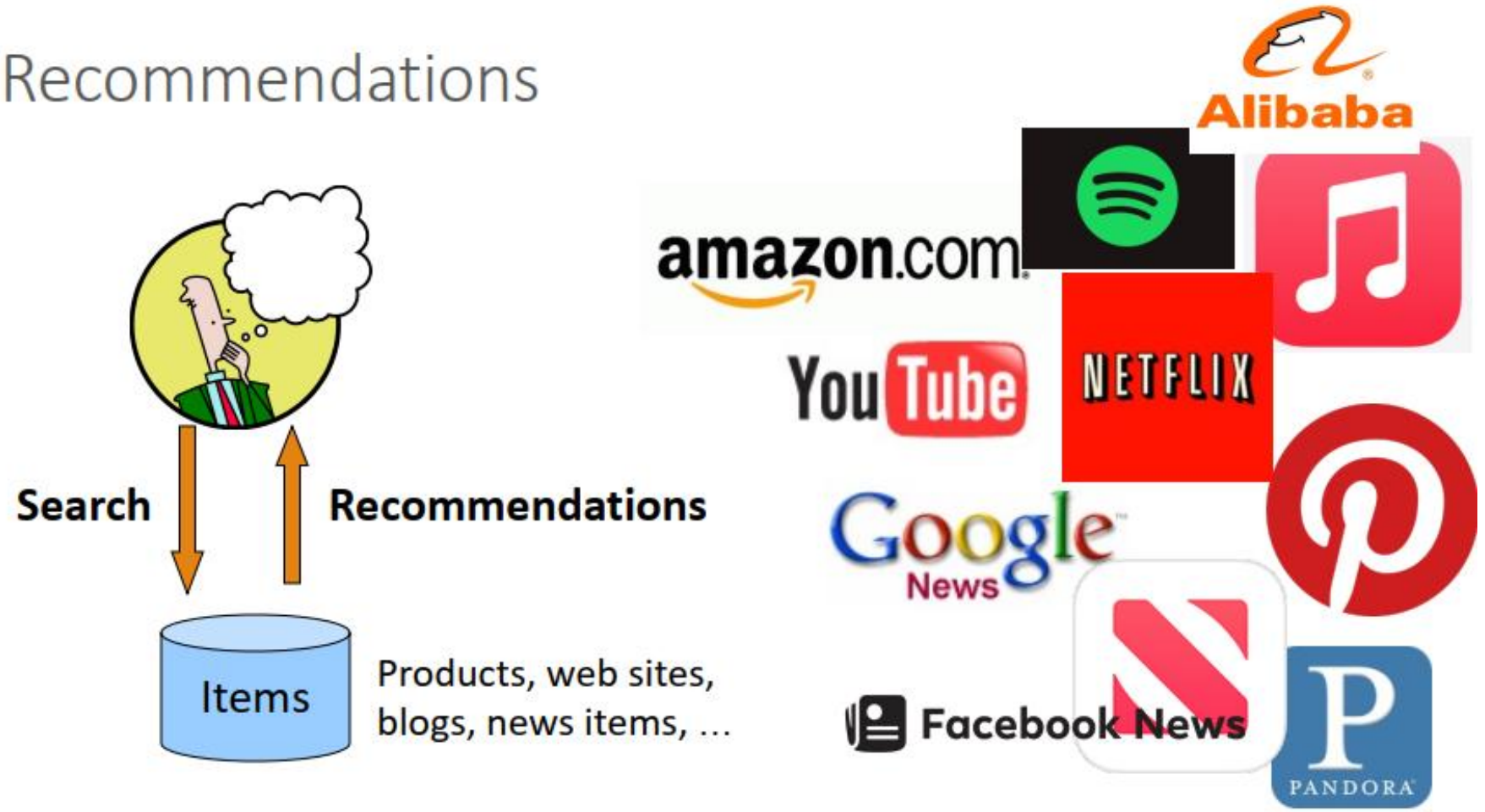
- Each candidate answer gets scores from >50 components
  - (from unstructured text, semi-structured text, triple stores)
  - logical form (parse) match between question and candidate
  - passage source reliability
  - geospatial location
    - California is "southwest of Montana"
  - temporal relationships
  - taxonomic classification

Common Evaluation Metrics

1. *Accuracy* (does answer match gold-labeled answer?)
2. *Mean Reciprocal Rank*
   - For each query return a ranked list of M candidate answers.
   - Query score is 1/Rank of the first correct answer
     - *If first answer is correct: 1*
     - *else if second answer is correct: ½*
     - *else if third answer is correct: ⅓, etc.*
     - *Score is 0 if none of the M answers are correct*
   - Take the mean over all N queries

$$MRR = \frac{\sum_{i=1}^{N} \frac{1}{rank_i}}{N}$$

Recommender systems: The task



Types of Recommendations

Editorial and hand curated

- List of favorites
- Lists of “essential” items

Simple aggregates

- Top 10, Most Popular, Recent Uploads

Tailored to individual users

- Amazon, Netflix, Apple Music…

Knowing how personalized recommendations work

Relevant for building practical news or product recommenders

Formal Model

X = set of **Users**

S = set of **Items**

Utility function *u*: *X* × *S* → *R*

- ***R*** = set of ratings
- ***R*** is a totally ordered set
- e.g., **1-5** stars, real number in **[0,1]**

Utility Matrix

		Harry Potter			Twilight	Star Wars		
		HP1	HP2	HP3	TW	SW1	SW2	SW3
Anita	A	4			5	1		
Beyonce	B	5	5	4				
Calvin	C				2	4	5	
David	D		3					3

Key Problems

1. **Gathering "known" ratings for matrix**
   - How to collect the data in the utility matrix

2. **Extrapolate unknown ratings from known ones**
   - Mainly interested in high unknown ratings
   - We are not interested in knowing what you don't like but what you like

3. **Evaluating extrapolation methods**
   - How to measure performance of recommendation methods

(1) Gathering Ratings

Explicit

- Ask people to rate items
- Doesn’t work well in practice – people can't be bothered
- Crowdsourcing: Pay people to label items

Implicit

- Learn ratings from user actions
  - E.g., purchase (or watch video, or read article) implies high rating

(2) Extrapolating Utilities

Key problem: Utility matrix ***U*** is **sparse**

- Most people have not rated most items

- **The "Cold Start" Problem:**
  - New items have no ratings
  - New users have no history

(2) Extrapolating Utilities

Three approaches to recommender systems:

1. Content-based
2. Collaborative Filtering

} **This lecture!**

3. Latent factor (Neural embedding) based

Content-based Recommendations

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Movie recommendations

- Recommend movies with same actor(s), director, genre, …

Websites, blogs, news

- Recommend other sites with similar types or words

Item Profiles

For each item, create an **item profile**

Profile is a set (vector) of features

- **Movies:** genre, director, actors, year…
- **Text:** Set of "important" words in document

How to pick important features?

- **TF-IDF** (Term frequency * Inverse Doc Frequency)
- For example use all words whose tf-idf > threshold, normalized for document length

Content-based Item Profiles

	Melissa McCarthy	Actor A	Actor B	…	Johnny Depp	Comic Genre	Spy Genre	Pirate Genre
Movie X	0	1	1	0	1	1	0	1
Movie Y	1	1	0	1	0	1	1	0

But what if we want to have real or ordinal features too?

Content-based Item Profiles

	Melissa McCarthy	Actor A	Actor B	…	Johnny Depp	Comic Genre	Spy Genre	Pirate Genre	Avg Rating
Movie X	0	1	1	0	1	1	0	1	3
Movie Y	1	1	0	1	0	1	1	0	4

For example "average rating"

Maybe we want a scaling factor α between binary and numeric features

Content-based Item Profiles

	Melissa McCarthy	Actor A	Actor B	…	Johnny Depp	Comic Genre	Spy Genre	Pirate Genre	Avg Rating
Movie X	0	1	1	0	1	1	0	1	3α
Movie Y	1	1	0	1	0	1	1	0	4α

Scaling factor α between binary and numeric features

$$\text{Cosine(Movie X, Movie Y)} = \frac{2+12\alpha^2}{\sqrt{5+9\alpha^2}\sqrt{5+16\alpha^2}}$$

$\alpha = 1$: 0.82 $\alpha = 2$: 0.94 $\alpha = 0.5$: 0.69

User Profiles

Want a vector with the same components/dimensions as items

- Could be 1s representing user purchases
- Or arbitrary numbers from a rating

User profile is aggregate of items:

- Weighted average of rated item profiles

Sample user profile

- Items are movies
- Utility matrix has 1 if user has seen movie
- 20% of the movies user U has seen have Melissa McCarthy
- U[“Melissa McCarthy”] = 0.2

	Melissa McCarthy	Actor A	Actor B	…	
User U	0.2	.005	0	0	…

Prediction

- Users and items have the same dimensions!

	Melissa McCarthy	Actor A	Actor B	…	
Movie i	0	1	1	0	…
User x	0.2	.005	0	0	0

- So just recommend the items whose vectors are most similar to the user vector!

- Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$,
- estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

Pros: Content-based Approach

+: No need for data on other users

- No user sparsity problems

+: Able to recommend to users with unique tastes

+: Able to recommend new & unpopular items

- No first-rater problem

+: Able to provide explanations

- Just list the content-features that caused an item to be recommended

Cons: Content-based Approach

- **Finding the appropriate features is hard**
  - E.g., images, movies, music
- **Recommendations for new users**
  - **How to build a user profile?**
- **Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to exploit quality judgments of other users**

Collaborative filtering

Instead of using content features of items to determine what to recommend

Find similar users and recommend items that they like!

Collaborative Filtering
Version 1: "User-User" Collaborative Filtering

Consider user ***x***

and unrated item ***i***

Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings

Estimate ***x***'s ratings for ***i*** based on ratings for ***i*** of users in ***N***

Collaborative filtering

Find similar users and recommend items that they like:

- Represent users by their rows in the **utility matrix**
- Two users are similar if their vectors are similar!

	Harry Potter			Twilight	Star Wars		
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Let r_x be the vector of user x's ratings

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

Cosine similarity measure

- $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \, ||r_y||}$

Problem: This representation leads to unintuitive results

Problems with raw utility matrix cosine

	Harry Potter			Twilight	Star Wars		
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)

$$\text{sim(A,B)} = \frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{5^2 + 5^2 + 4^2}} = 0.380$$

$$\text{sim(A,C)} = \frac{5 \times 2 + 1 \times 4}{\sqrt{4^2 + 5^2 + 1^2}\sqrt{2^2 + 4^2 + 5^2}} = 0.322$$

Yes, 0.380 > 0.322

But only barely works…

Problem with raw cosine

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Problem with cosine:
  - C really loves SW
  - A hates SW
  - B just hasn't seen it
- Another problem: we'd like to normalize the raters
  - D rated everything the same; not very useful

Mean-Centered Utility Matrix: subtract the means of each row

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

- Now a 0 means no information
- And negative ratings means viewers with opposite ratings will have vectors in opposite directions!

Modified Utility Matrix: subtract the means of each row

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

$$\text{Cos(A,B)} = \frac{(2/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(1/3)^2 + (1/3)^2 + (-2/3)^2}} = 0.092$$

$$\text{Cos(A,C)} = \frac{(5/3) \times (-5/3) + (-7/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2}\sqrt{(-5/3)^2 + (1/3)^2 + (4/3)^2}} = -0.559$$

Now A and C are (correctly) way further apart than A,B

Terminological Note: subtracting the mean is **mean-centering**

Finding similar users with overlapping-item mean-centering

Let r_x be the vector of user x's ratings

r_x = {1, 0, 0, 1, 3}

r_y = {1, 0, 2, 2, 0}

r_x = [*, _, _, *, ***]

r_y = [*, _, **, **, _]

Mean-centering:

- For each user x, let $\bar{r_x}$ be mean of r_x (ignoring missing values)
- $\bar{r_x}$ = (1 + 1 + 3)/3 = 5/3 $\bar{r_y}$ = (1 + 2 + 2)/3 = 5/3
- Subtract this average from each of their ratings
  - (but do nothing to the "missing values"; they stay "null").
  - mean centered r_x = {-2/3, 0, 0, -2/3, 4/3}

One new idea: Keep only items they both rate (unlike 2 slides ago)

r_x = {-2/3, [redacted] -2/3, [redacted]} r_y = {-2/3, [redacted], 1/3, [redacted]}

r_x = {-2/3, -2/3} r_y = {-2/3, 1/3}

Now take cosine:

- Now compute cosine between user vectors
  - cos([-2/3, -2/3], [-2/3, 1/3])

Mean-centered overlapping-item cosine similarity

Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings,and $\overline{r_x}$ be its mean (ignoring missing values)

Instead of basic cosine similarity measure

- $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r_x}, \boldsymbol{r_y}) = \frac{r_x \cdot r_y}{||r_x||\ ||r_y||}$

Mean-centered overlapping-item cosine similarity

(Variant of Pearson correlation)

- $\boldsymbol{S_{xy}}$ = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

Rating Predictions

From similarity metric to recommendations for an unrated item *i*:

Let r_x be the vector of user ***x***'s ratings

Let ***N*** be the set of ***k*** users most similar to ***x*** who have rated item ***i***

Prediction for item *i* of user *x*:

- Rate i as the mean of what k-people-like-me rated i

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

- Even better: Rate i as the mean weighted by their similarity to me …

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \, r_{yi}}{\sum_{y \in N} s_{xy}}$$

- **Many other tricks possible…**

Shorthand:

$$s_{xy} = sim(x, y)$$

Collaborative Filtering Version 2: Item-Item Collaborative Filtering

So far: User-user collaborative filtering

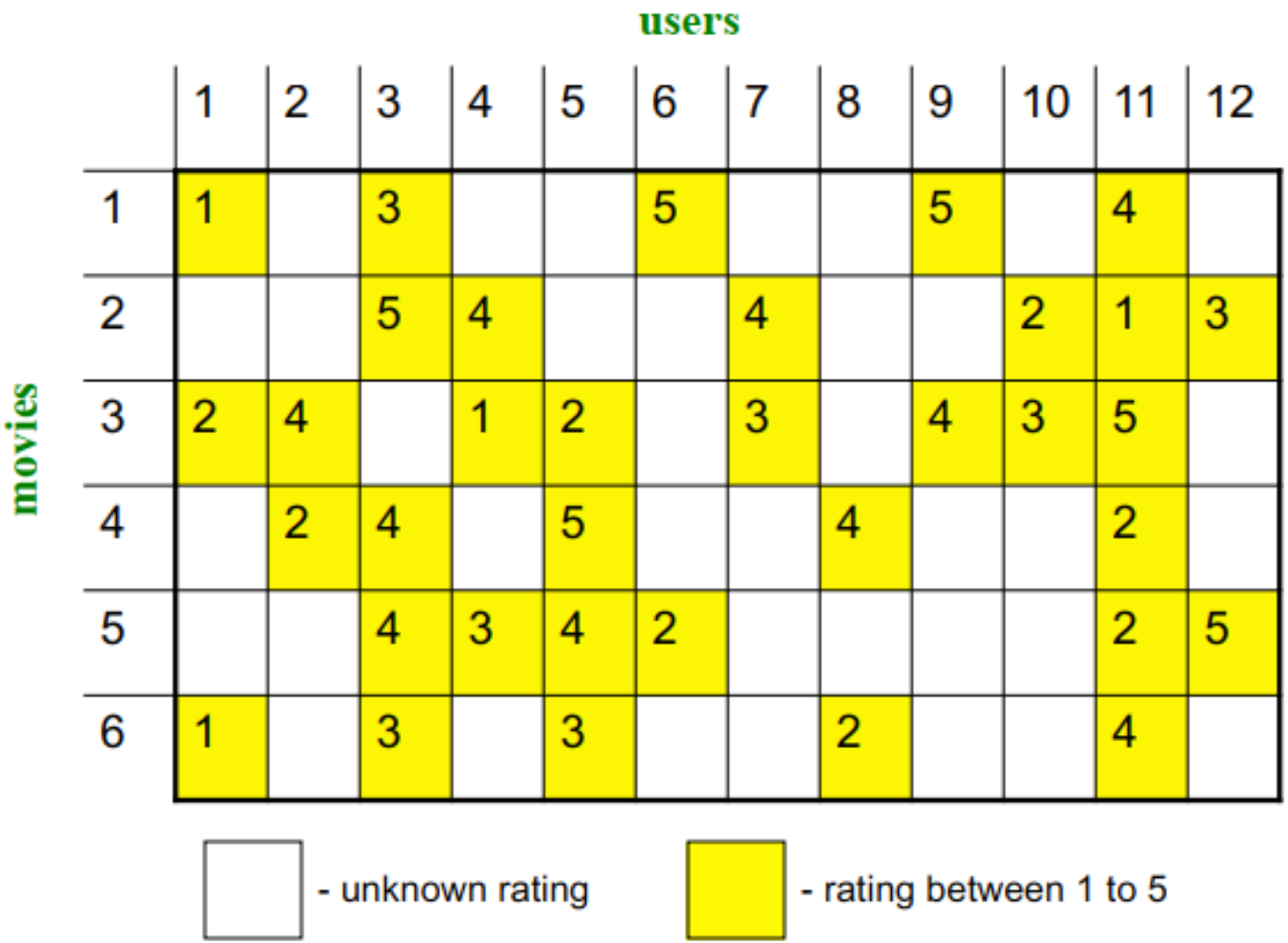
Alternate view that often works better: Item-item

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for those similar items
- Can use same similarity metrics and prediction functions as in user-user model
- "Rate i as the mean of my ratings for other items, weighted by their similarity to i"

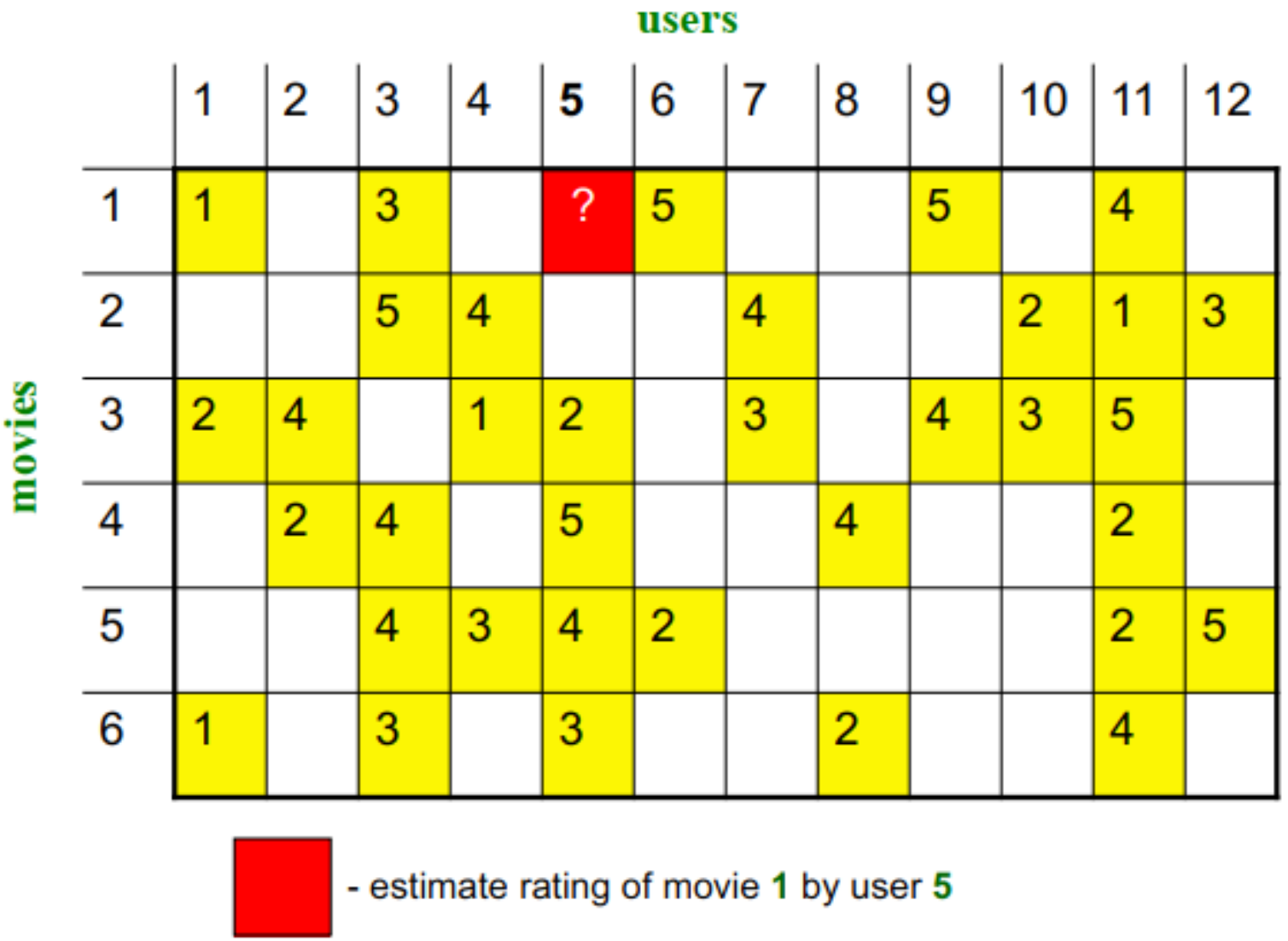
$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \; r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

N(i;x)…set of items rated by ***x*** *and* similar to ***i***
s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***x*** on item ***j***

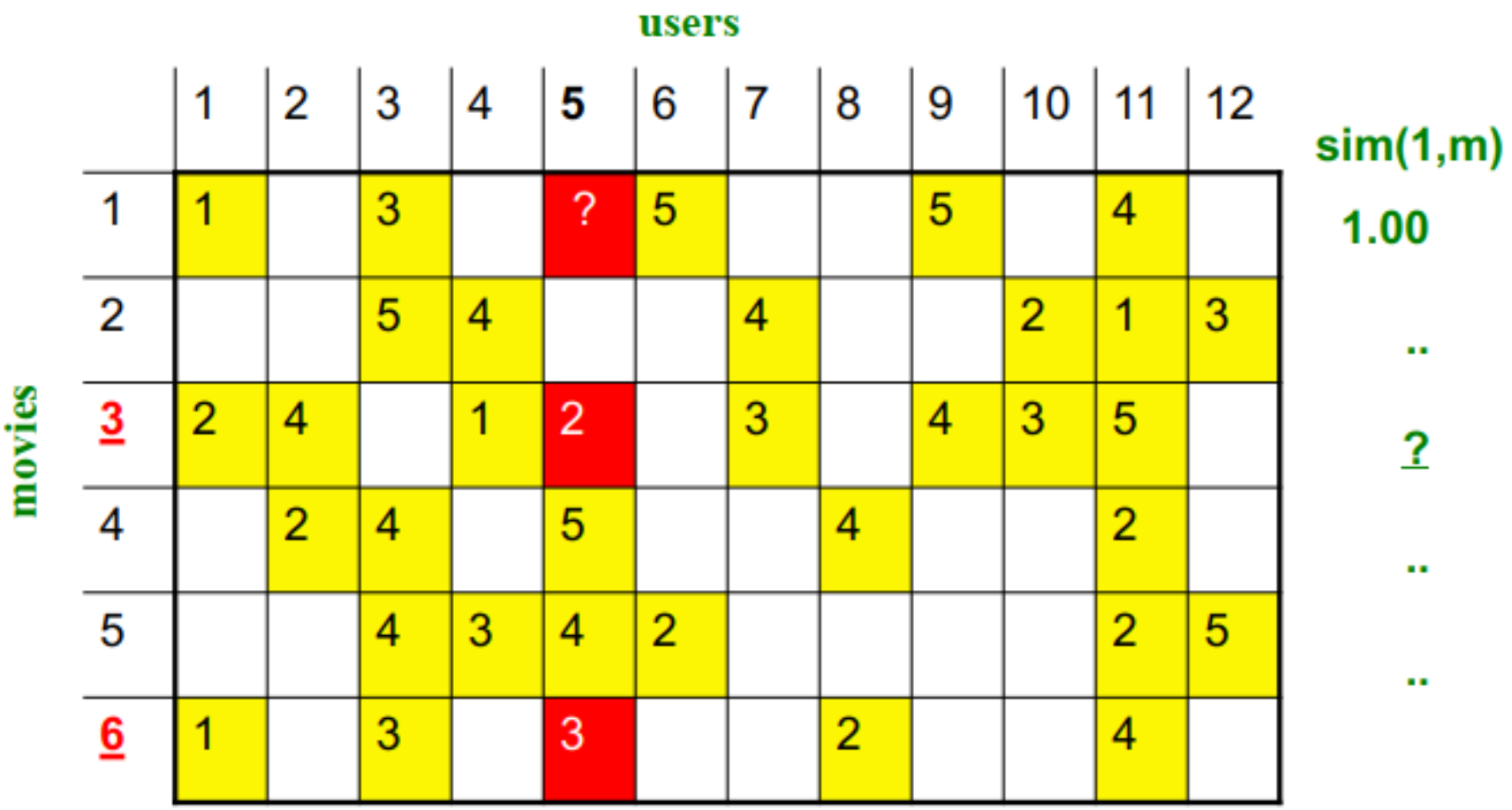
Item-Item CF (|N|=2)



Item-Item CF (|N|=2)



Item-Item CF (|N|=2)



Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use mean centered item-overlap cosine as similarity:

1) Subtract mean rating m_i from each movie ***i*** between rows

2) Compute (item-overlapping) cosine similarities

Item-Item CF (|N|=2)

users

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	

movies

Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = 18/5$

	1	2	3	4	5	6	7	8	9	10	11	12
1	-13/5		-3/5		?	7/5			7/5		2/5	
3	-1	1		-2	-1		0		1	0	2	
6	-8/5		2/5		2/5			-3/5			7/5	

Showing computation only for #3 and #6

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use mean centered item-overlap cosine as similarity:

1) Subtract mean rating m_i from each movie i

2) Compute (item-overlapping) cosine similarities between rows

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	-13/5		-3/5		?	7/5			7/5		2/5		1.00
2			5	4			4			2	1	3	..
3	-1	1		-2	-1		0		1	0	2		?
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	-8/5		2/5		2/5			-3/5			7/5		?

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use mean centered item-overlap cosine as similarity:

1) Subtract mean rating m_i from each movie ***i***

2) Compute (item-overlapping) cosine similarities between rows

Compute Cosine Similarity

For rows 1 and 3, they both have values for users 1, 9 and 11.

$$\text{sim}(1, 3) = \frac{(-13/5)(-1)+(7/5)(1)+(2/5)(2)}{\sqrt{(-13/5)^2+(7/5)^2+(2/5)^2}\cdot\sqrt{(-1)^2+(1)^2+(2)^2}} \approx 0.658$$

For rows 1 and 6, they both have values for users 1, 3 and 11.

$$\text{sim}(1, 6) = \frac{(-13/5)(-8/5)+(-3/5)(2/5)+(2/5)(7/5)}{\sqrt{(-13/5)^2+(-3/5)^2+(2/5)^2}\cdot\sqrt{(-8/5)^2+(2/5)^2+(7/5)^2}} \approx 0.768$$

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.000
2			5	4			4			2	1	3	..
3	2	4		1	2		3		4	3	5		.658
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	1		3		3			2			4		.768

Compute similarity weights:

$s_{1,3}$=.658, $s_{1,6}$=.768 (we compute $s_{1,2}$, $s_{1,4}$, $s_{1,5}$ too; let's assume those are smaller)

Item-Item CF (|N|=2)
Approximate rating with weighted mean

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		2.54	5			5		4		1.000
2			5	4			4			2	1	3	..
3	2	4		1	2		3		4	3	5		.658
4		2	4		5			4			2		..
5			4	3	4	2					2	5	..
6	1		3		3			2			4		.768

Predict by taking weighted average:

$r_{1,5}$ = (0.658*2 + 0.768*3) / (0.658+0.768) = 2.54

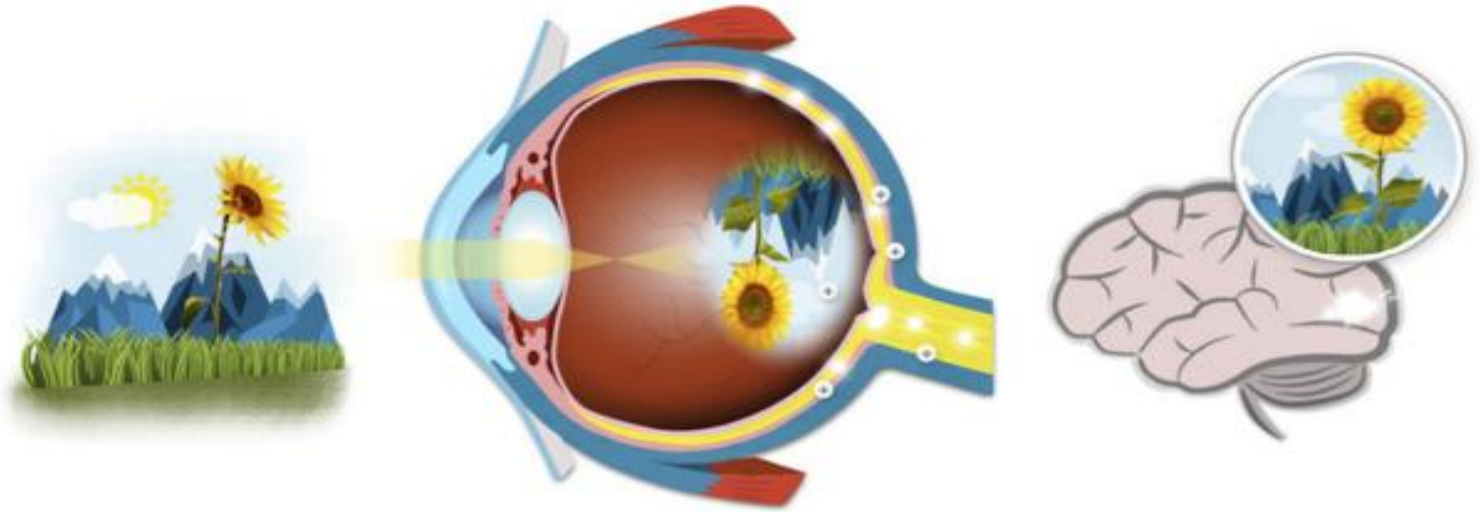
$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij}\, r_{jx}}{\sum s_{ij}}$$

Item-Item vs. User-User

- **In practice, item-item often works better than user-user**

- **Why?** Items are simpler, users have multiple tastes
  - (People are more complex than objects)

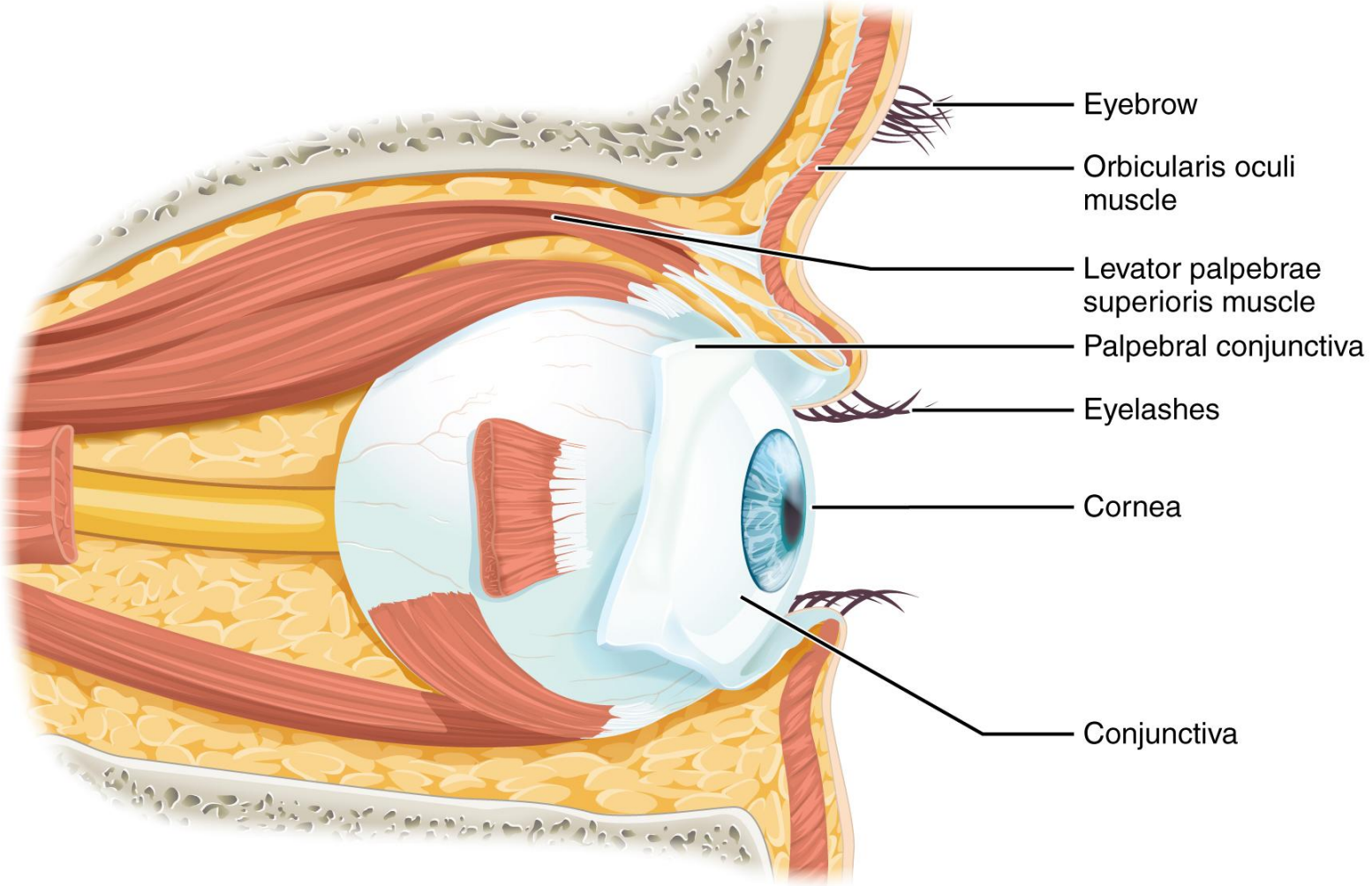
Human Vision



The eye Translates visual information into electric information

The brain's visual cortex interprets the electrical form of the image

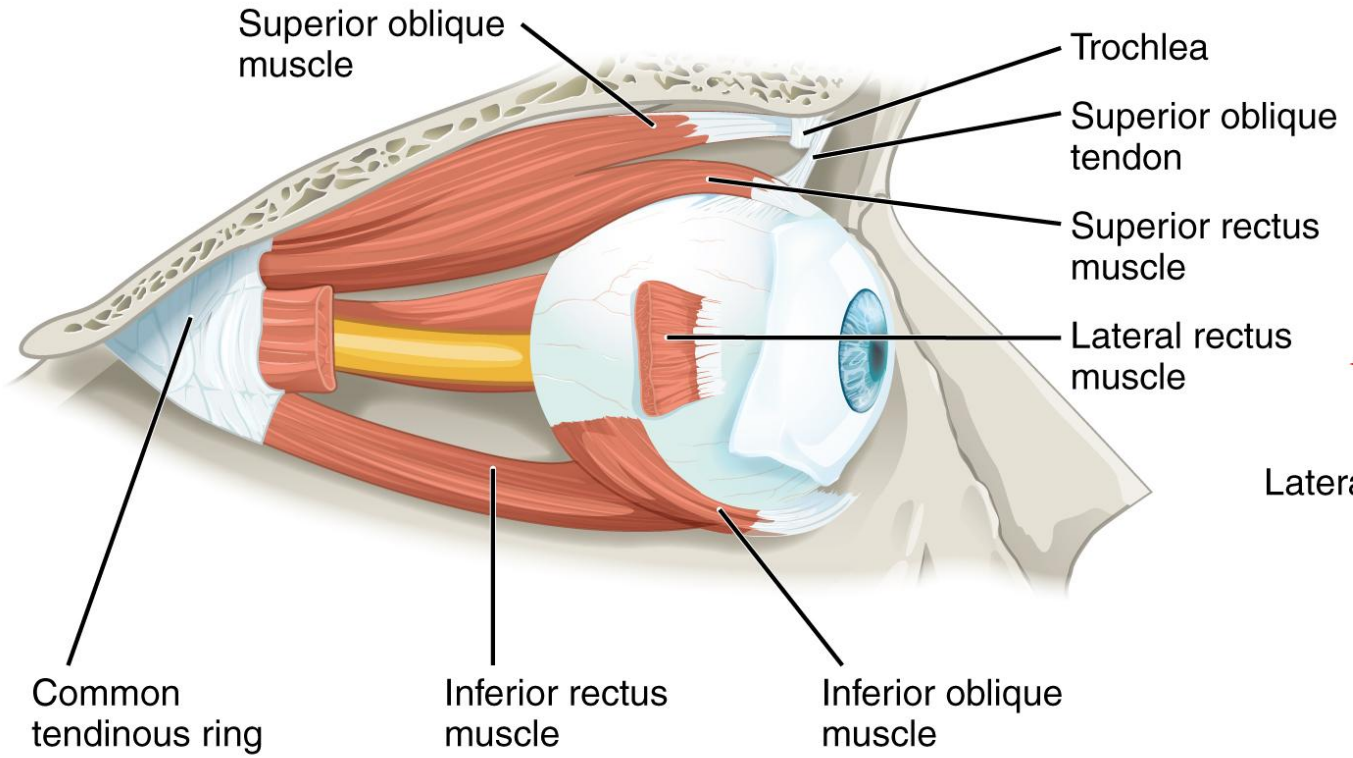
The Eye in the Orbit



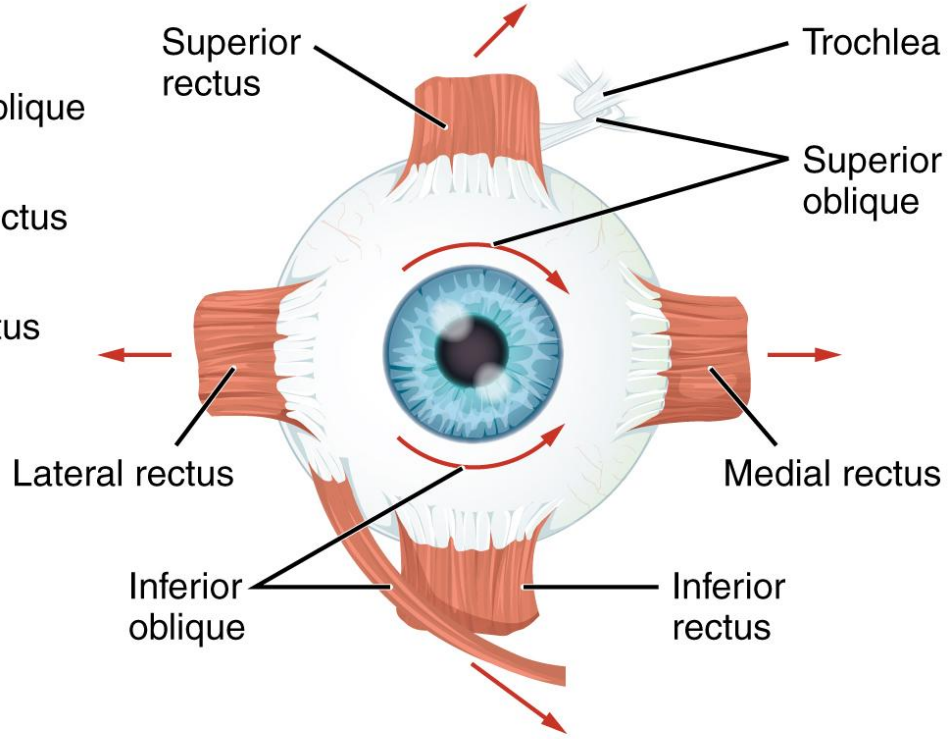
The eye is located within the orbit and surrounded by soft tissues that protect and support its function.

The orbit is surrounded by cranial bones of the skull.

Extraocular Muscles



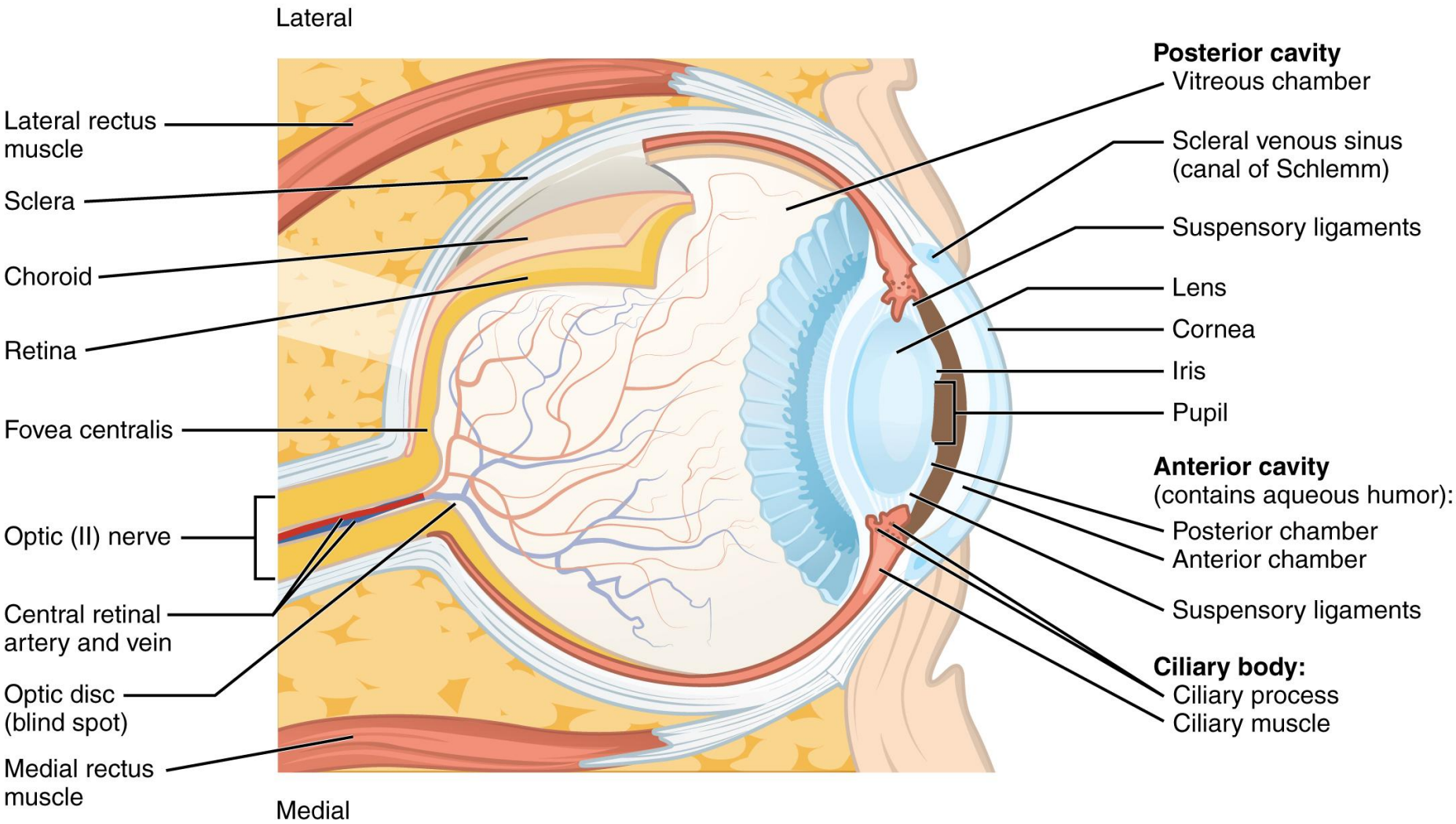
Lateral view of the right eye



Anterior view of the right eye

The extraocular muscles move the eye within the orbit.

Structure of the Eye



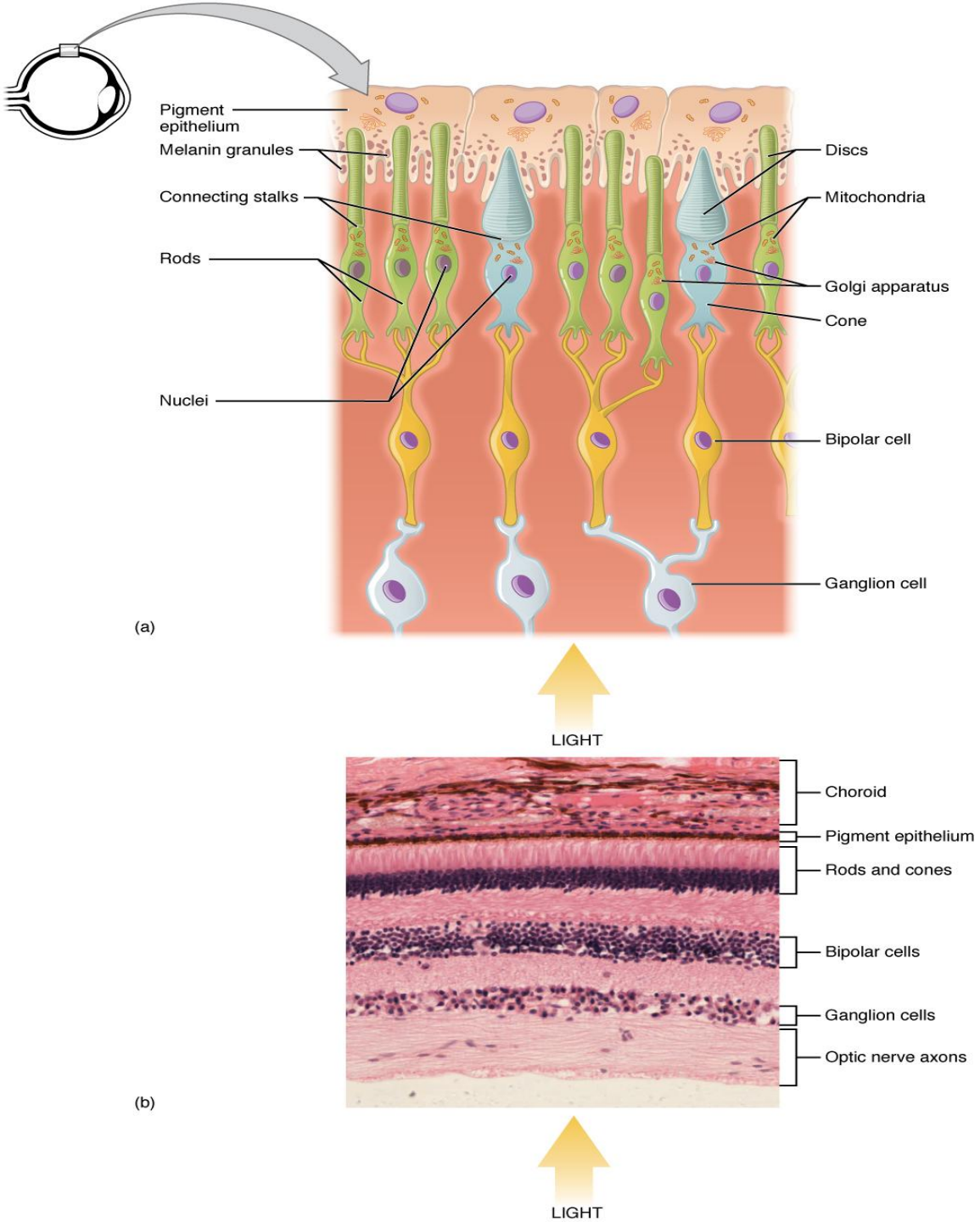
The sphere of the eye can be divided into anterior and posterior chambers.

The wall of the eye is composed of three layers: the fibrous tunic, vascular tunic, and neural tunic.

Within the neural tunic is the retina, with three layers of cells and two synaptic layers in between.

The center of the retina has a small indentation known as the fovea.

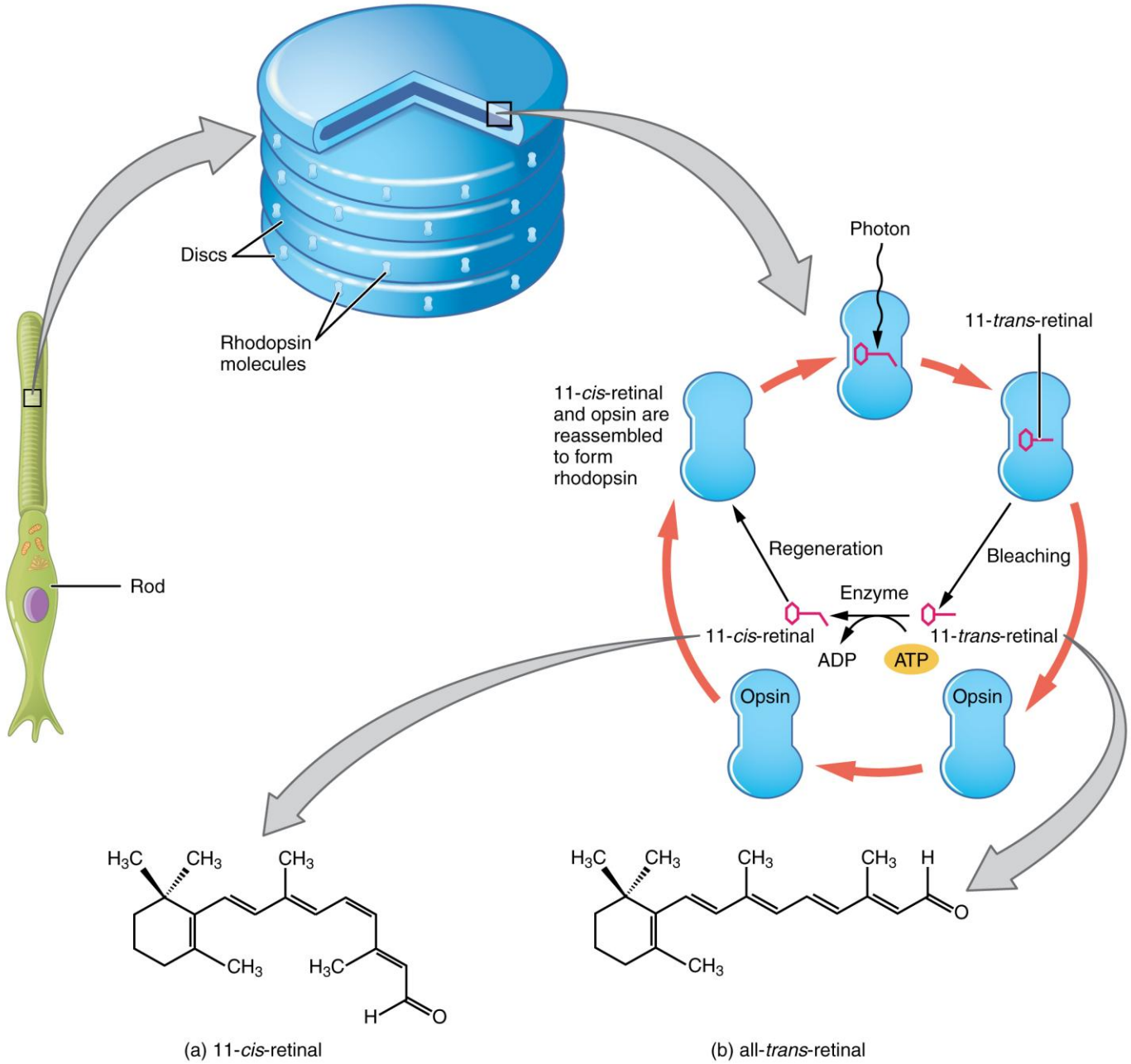
Photoreceptor



(a) All photoreceptors have inner segments containing the nucleus and other important organelles and outer segments with membrane arrays containing the photosensitive opsin molecules. Rod outer segments are long columnar shapes with stacks of membrane-bound discs that contain the rhodopsin pigment. Cone outer segments are short, tapered shapes with folds of membrane in place of the discs in the rods.

(b) Tissue of the retina shows a dense layer of nuclei of the rods and cones. LM × 800

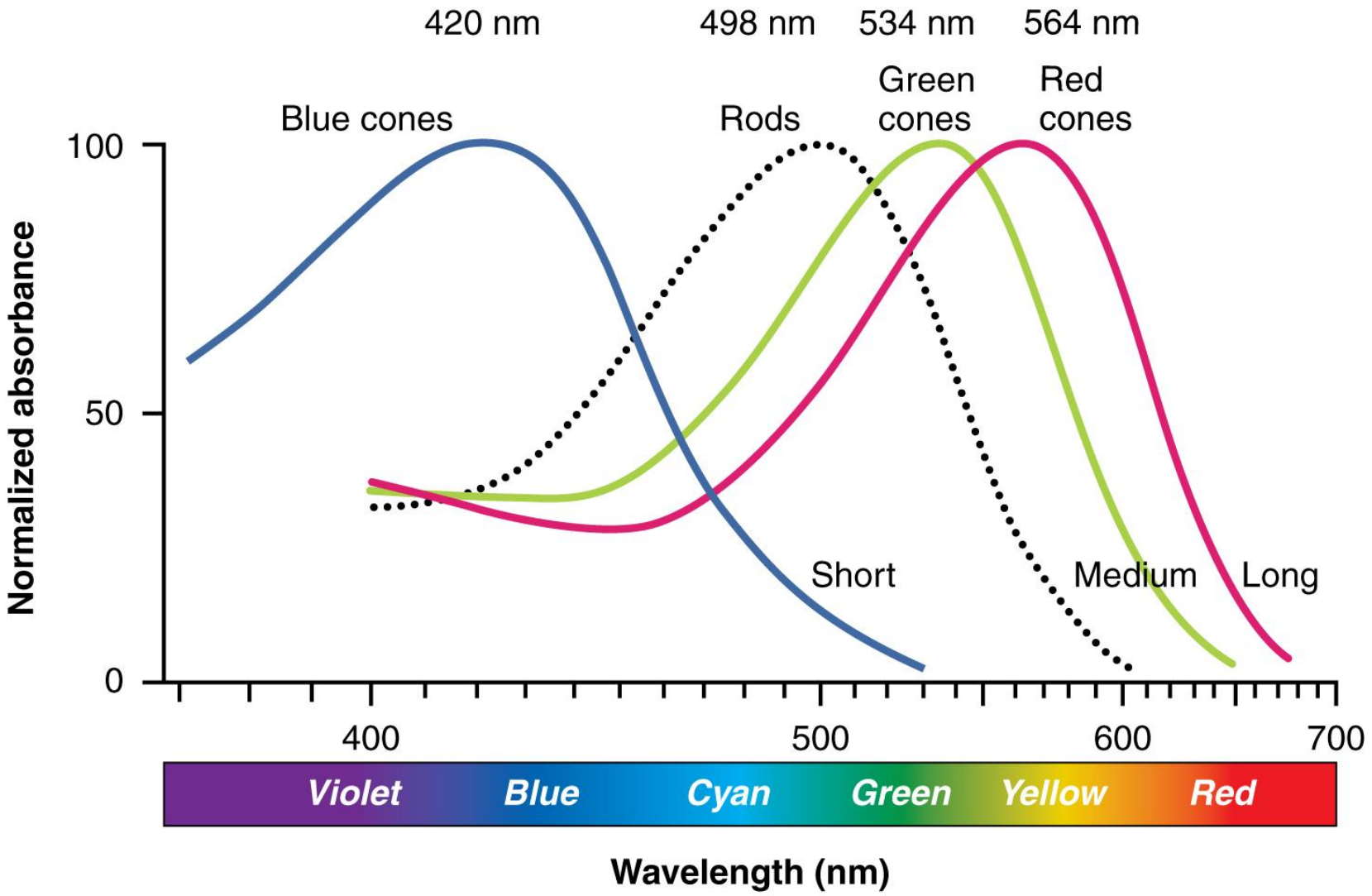
Retinal Isomers



The retinal molecule has two isomers:

(a) one before a photon interacts with it, and

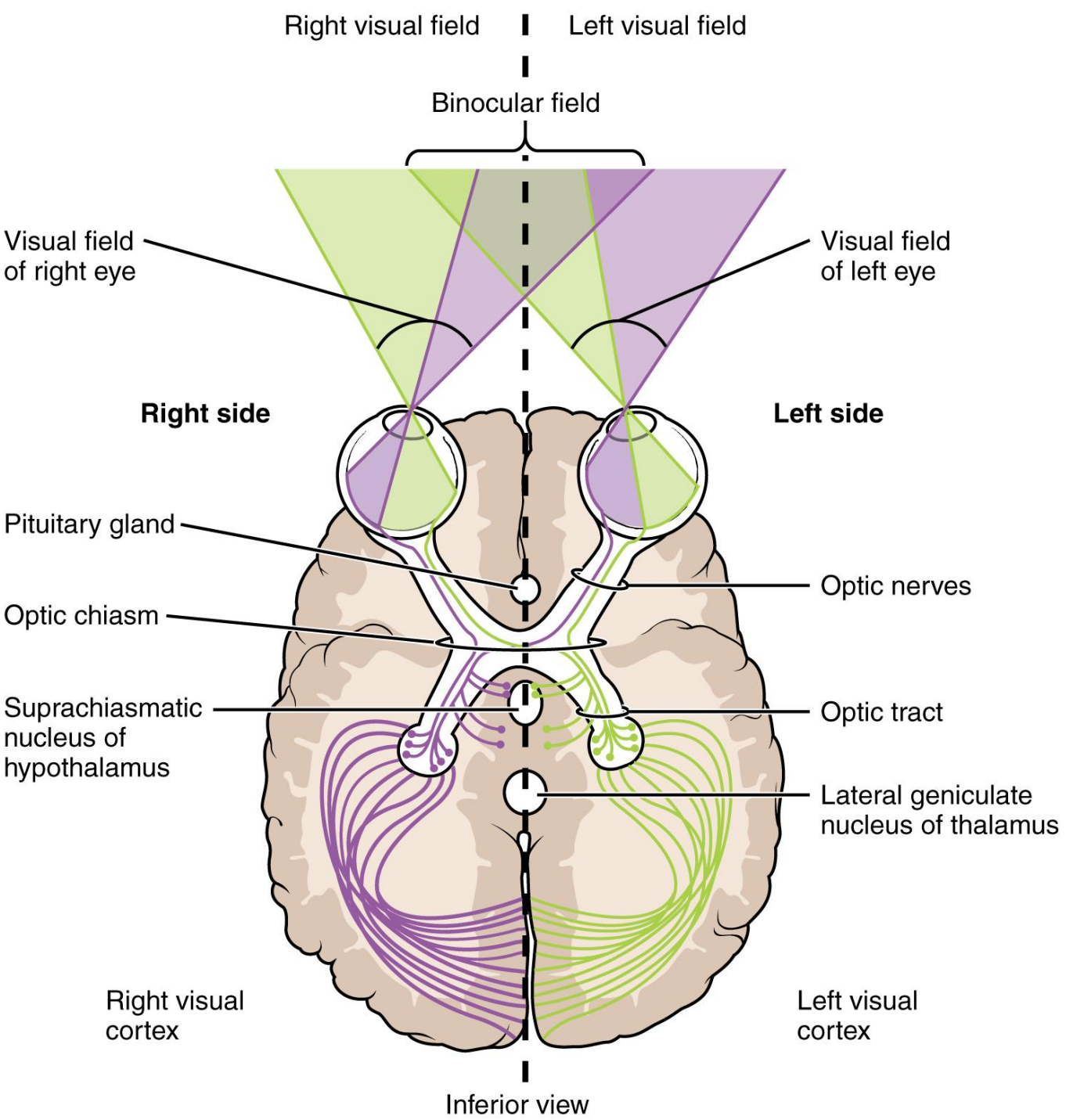
(b) one that is altered through photoisomerization.



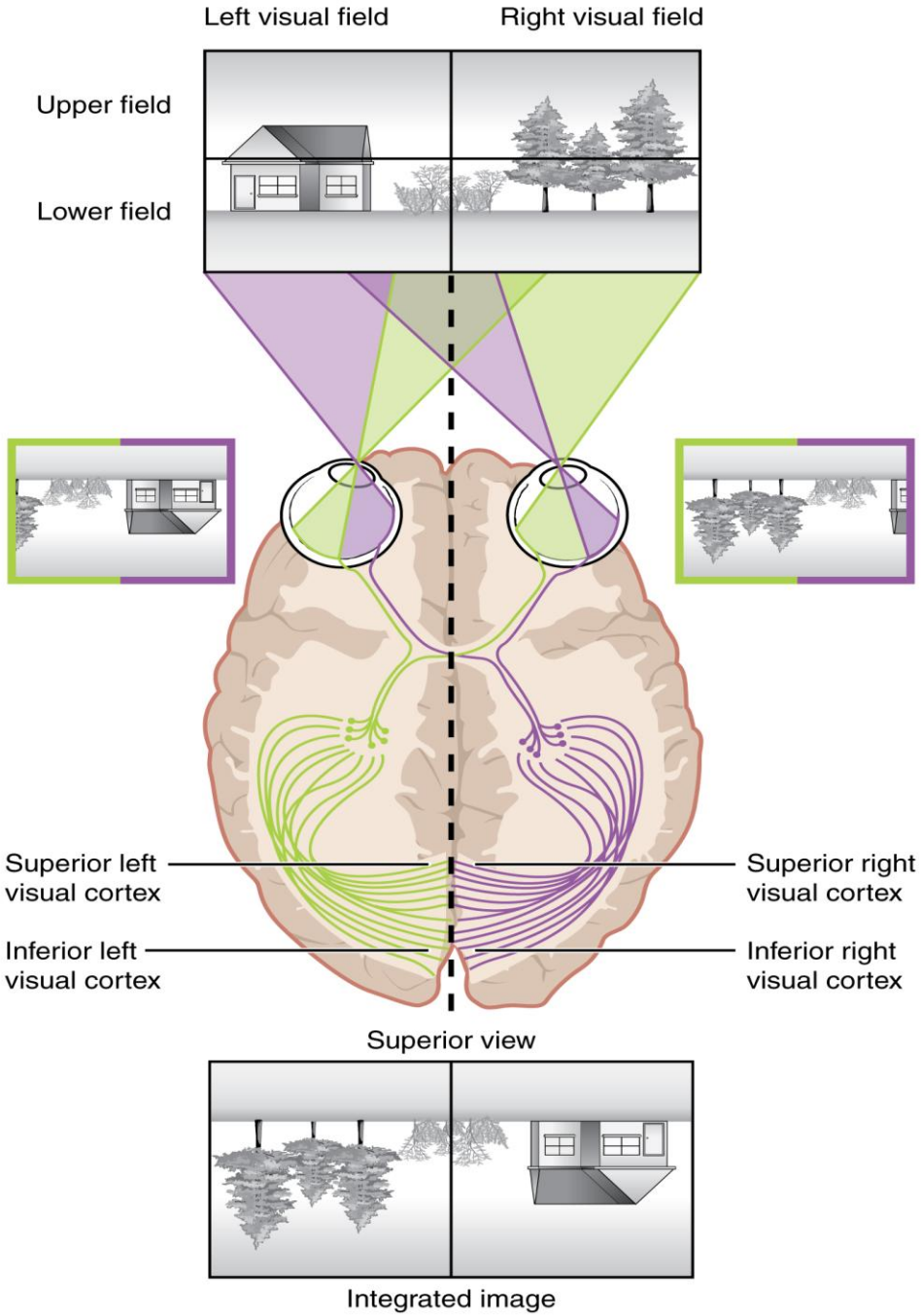
Comparison of Color Sensitivity of Photopigments

Comparing the peak sensitivity and absorbance spectra of the four photopigments suggests that they are most sensitive to particular wavelengths

Segregation of Visual Field Information at the Optic Chiasm



Contralateral visual field information from the lateral retina projects to the ipsilateral brain, whereas ipsilateral visual field information has to decussate at the optic chiasm to reach the opposite side of the brain.



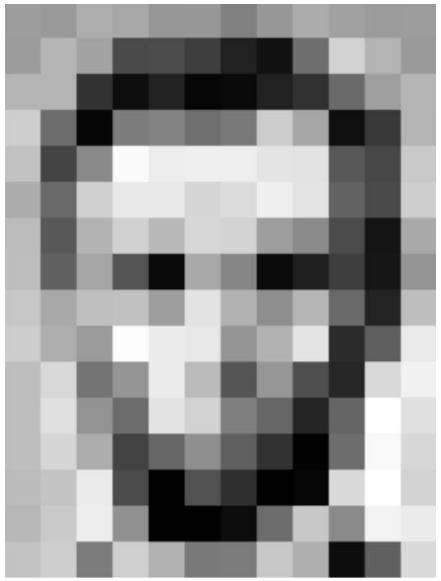
Topographic Mapping of the Retina onto the Visual Cortex

The visual field projects onto the retina through the lenses and falls on the retinae as an inverted, reversed image.

The topography of this image is maintained as the visual information travels through the visual pathway to the cortex.

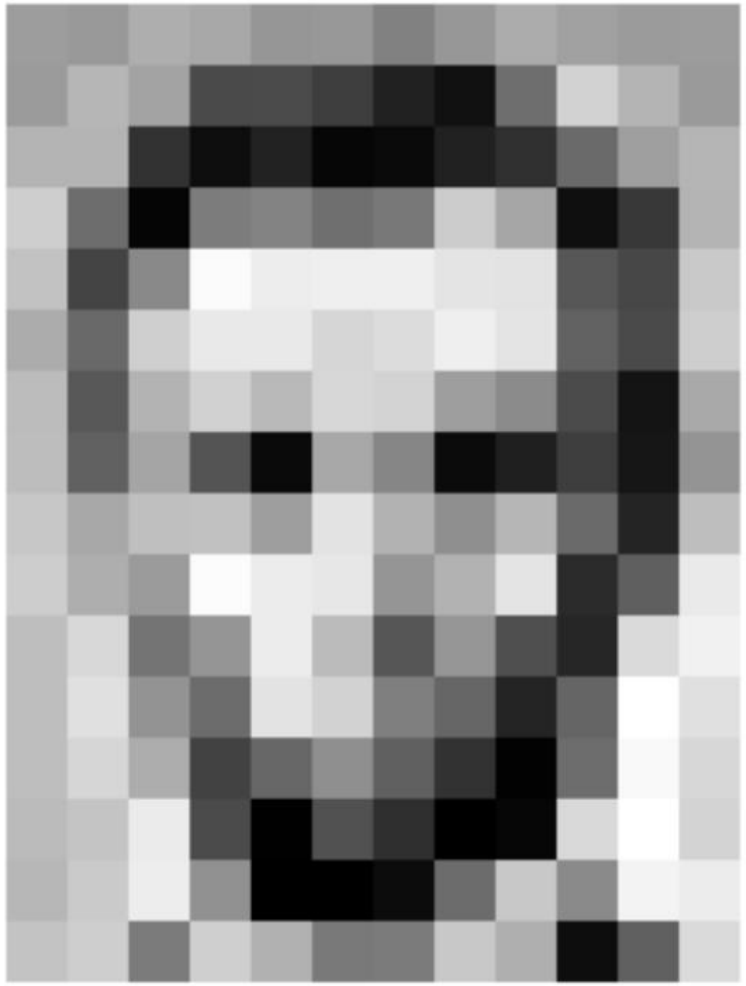
Computer Vision

What computers “see”?



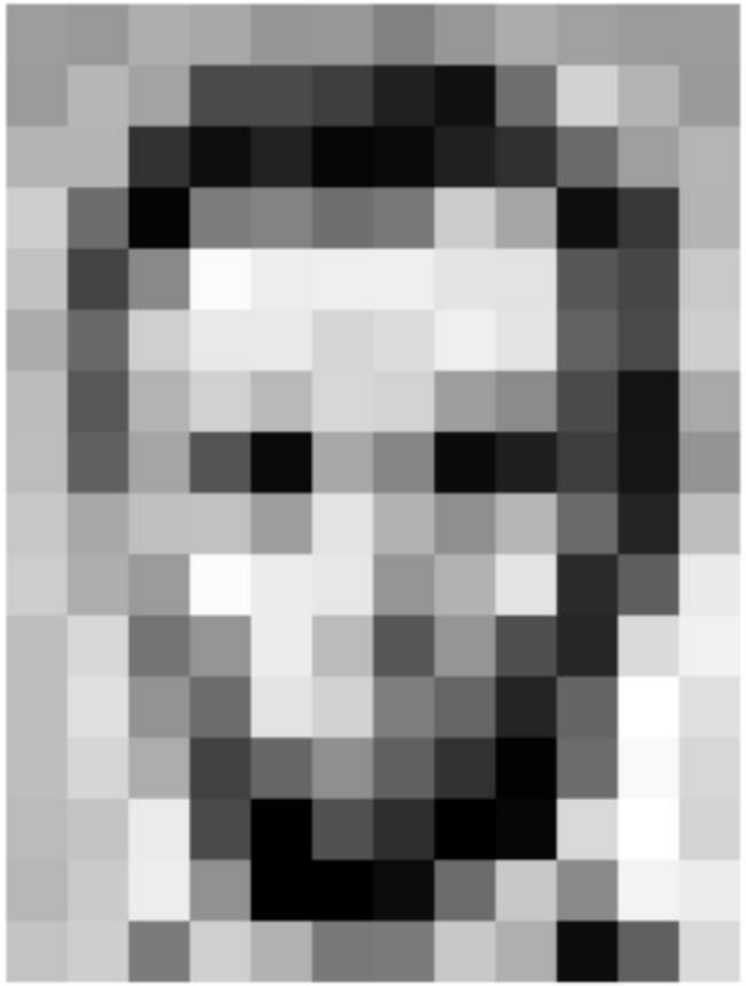
Images are numbers

Images are numbers



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

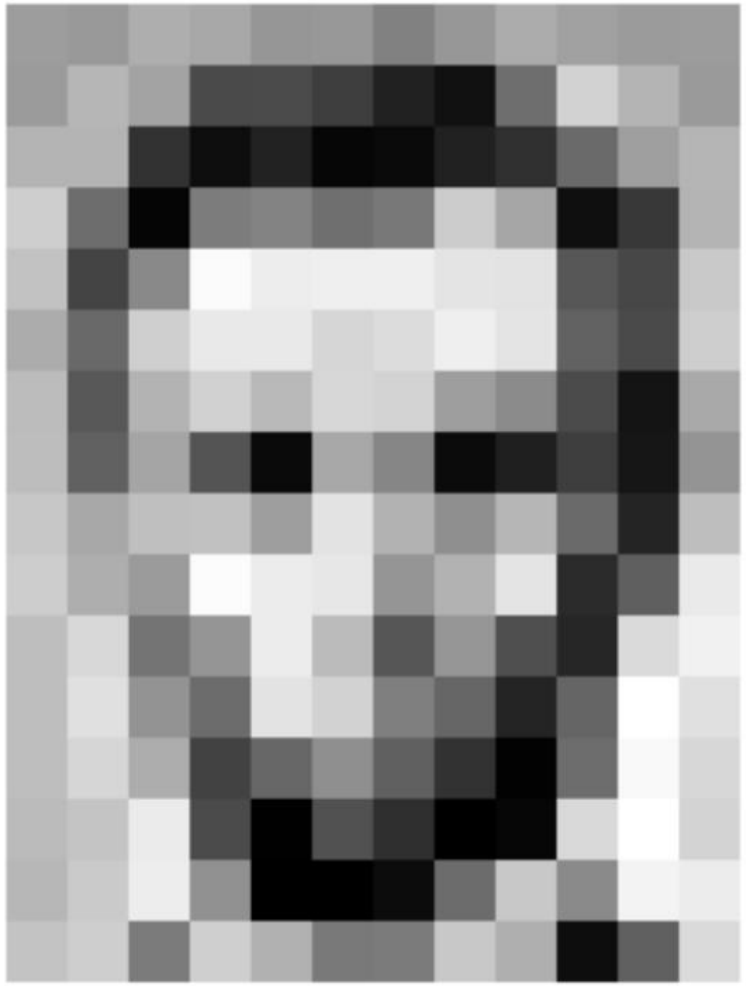
Images are numbers

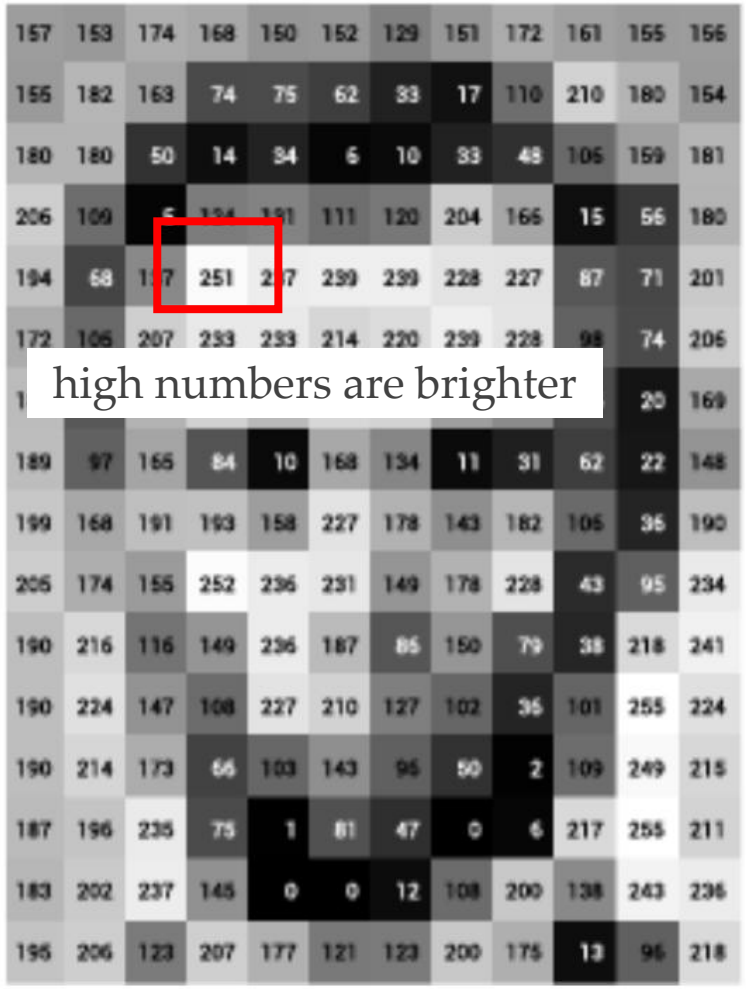


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
										22	148
										36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Low numbers are darker

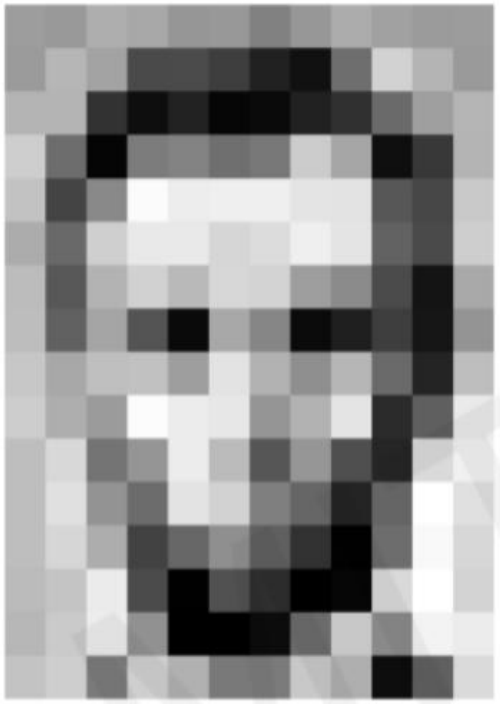
Images are numbers





high numbers are brighter

Images are numbers



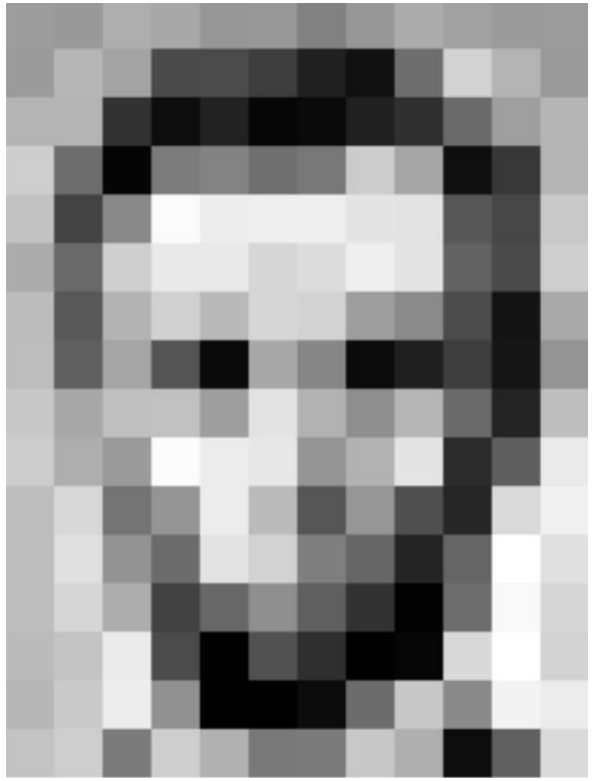
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

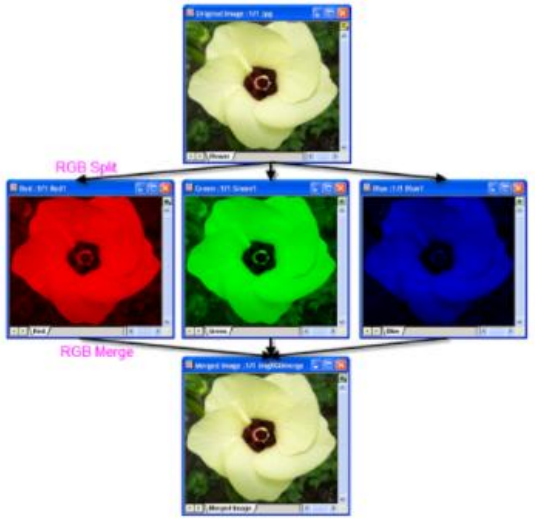
Grey level image



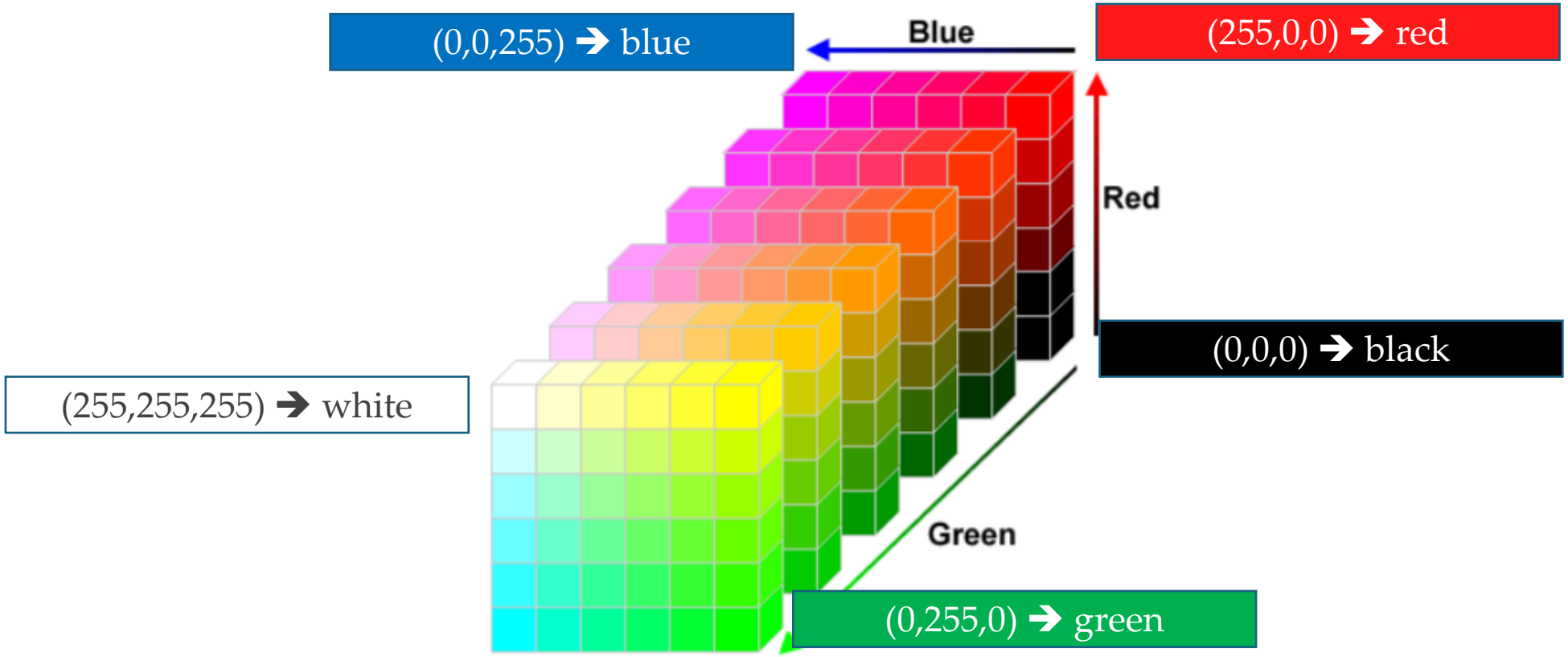
1 b/w band
0-255
0 = black
255 = white

Color image

תמונה צבעונית היא הרכבה של 3 תמונות

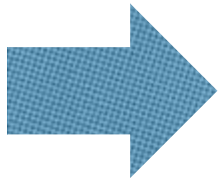


RGB Pallet



Tasks in computer vision – example 1

Classification

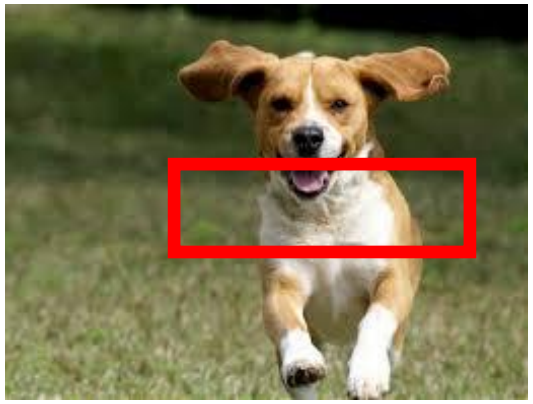


Dog	0.8
Cat	0.1
…	…
Horse	0.01

What is in the picture?

Tasks in computer vision – example 2

Regression #1



x,y,w,h

Where is the object?

Tasks in computer vision – example 3

Regression #2

x

How many cars are parking in the parking lot?

Tasks in computer vision – example 4

Segmentation

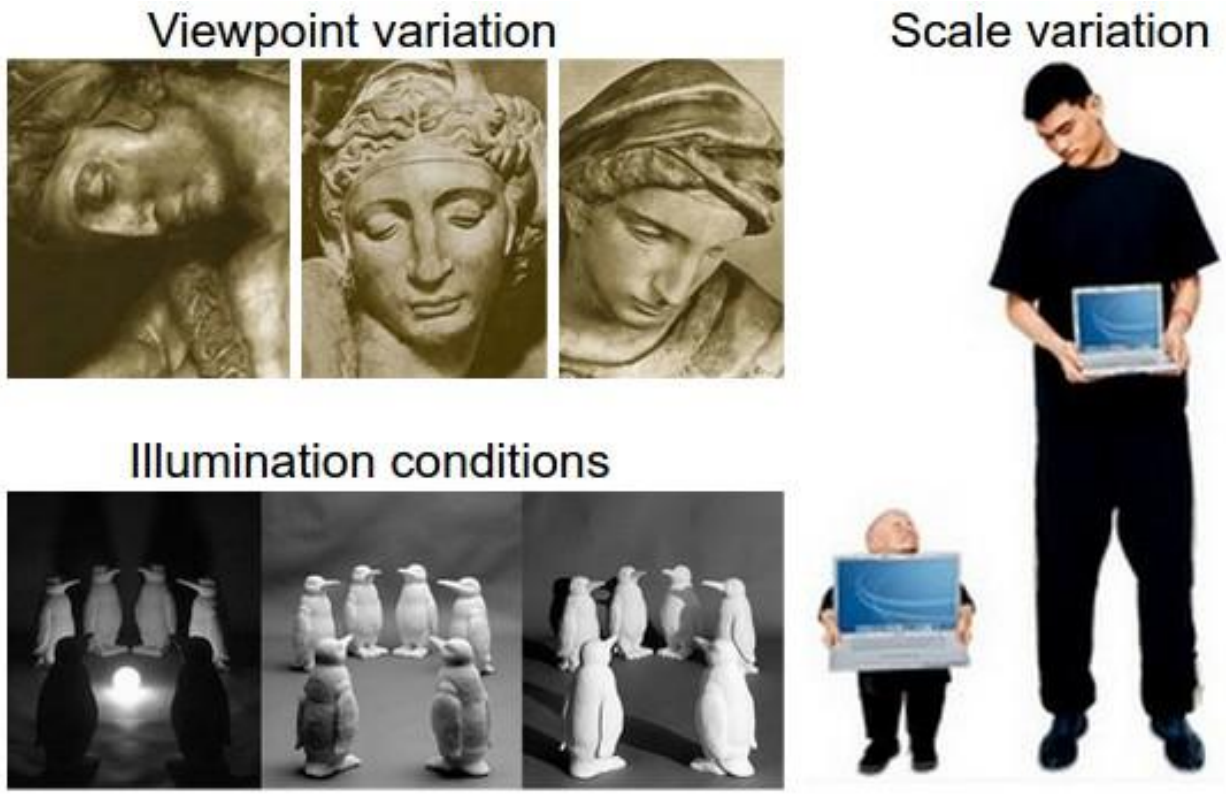
Pixel (i,j) ➔ segment

Tasks in computer vision

Many other tasks:

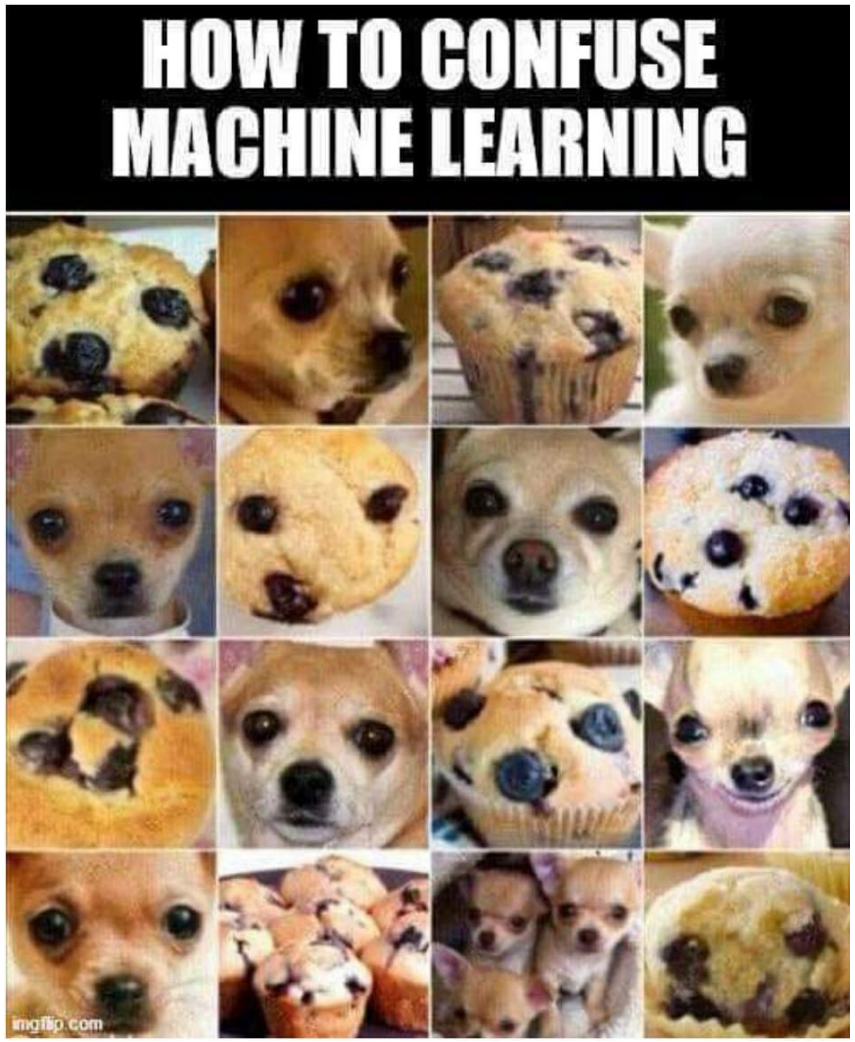
- Action Recognition
- Pose estimation
- Facial recognition
- 3d reconstruction
- …

Challenges in computer vision



Challenges in computer vision

Challenges in computer vision



Why study perception?

- **Perception is messy: Can we avoid it?**

  No!

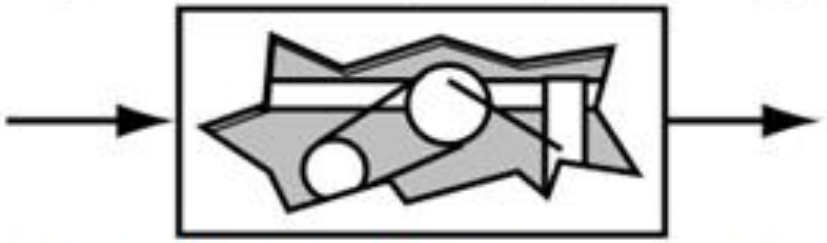
- **Audition provides the 'ground truth' in audio**
  - what is relevant and irrelevant
  - subjective importance of distortion

- **Some sounds are 'designed' for audition**
  - co-evolution of speech and hearing
- **The auditory system is very successful**
  - we would do extremely well to duplicate it

How to study perception?

Three different approaches:

- **Analyze the example: physiology**



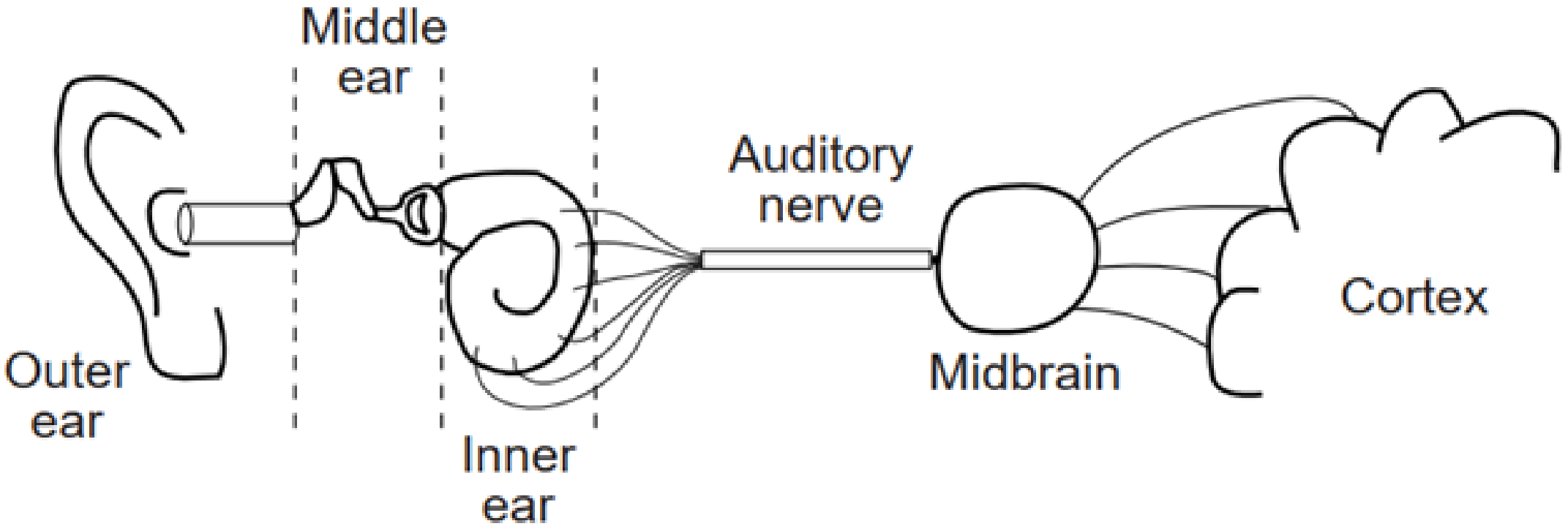
  - dissection & nerve recordings

- **Black box input/output: psychophysics**

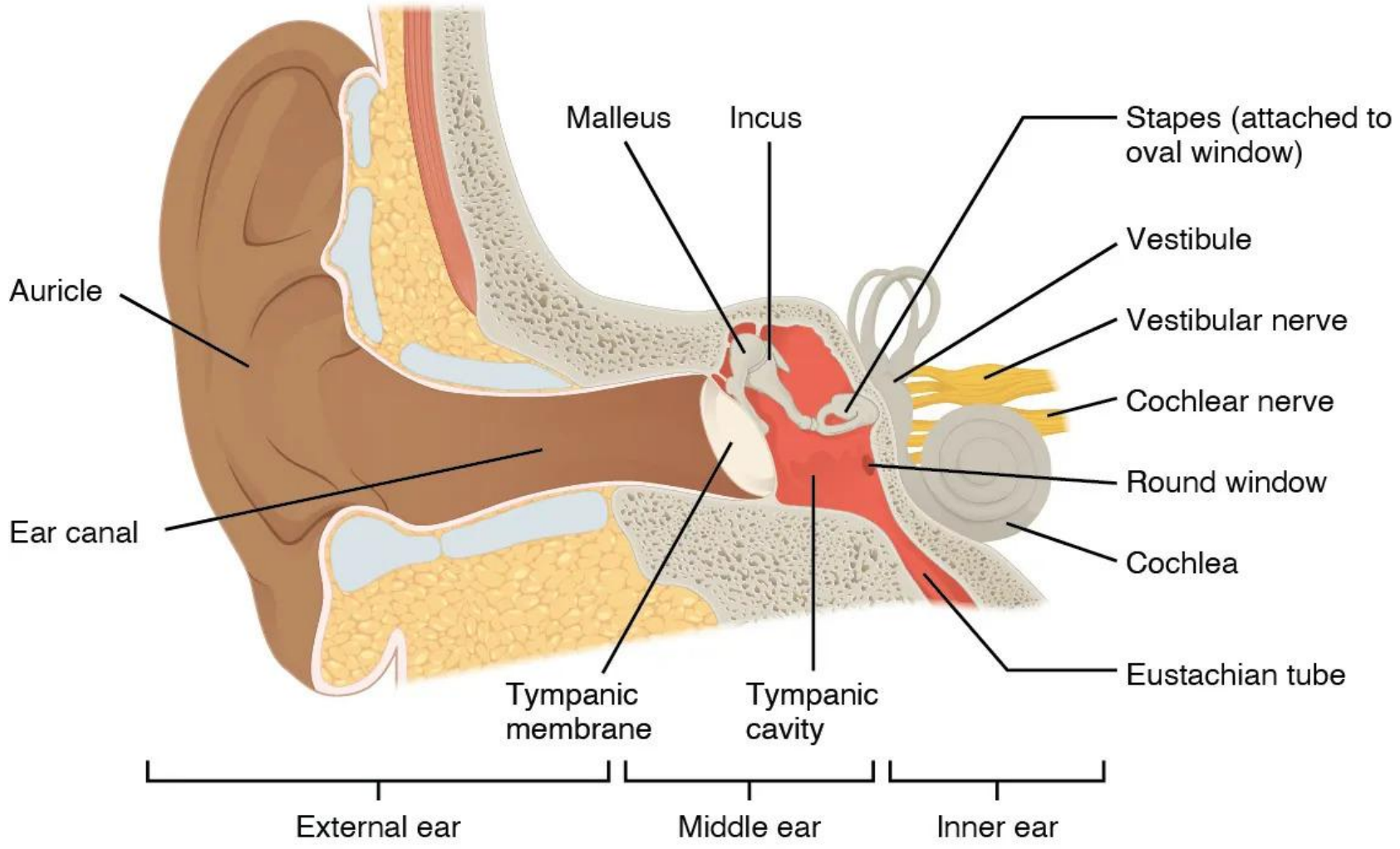
  - fit simple models of simple functions

- **Information processing models**
  - investigate and model complex functions
  - e.g. scene analysis, speech perception

- **Processing chain from air to brain:**



Structures of the Ear

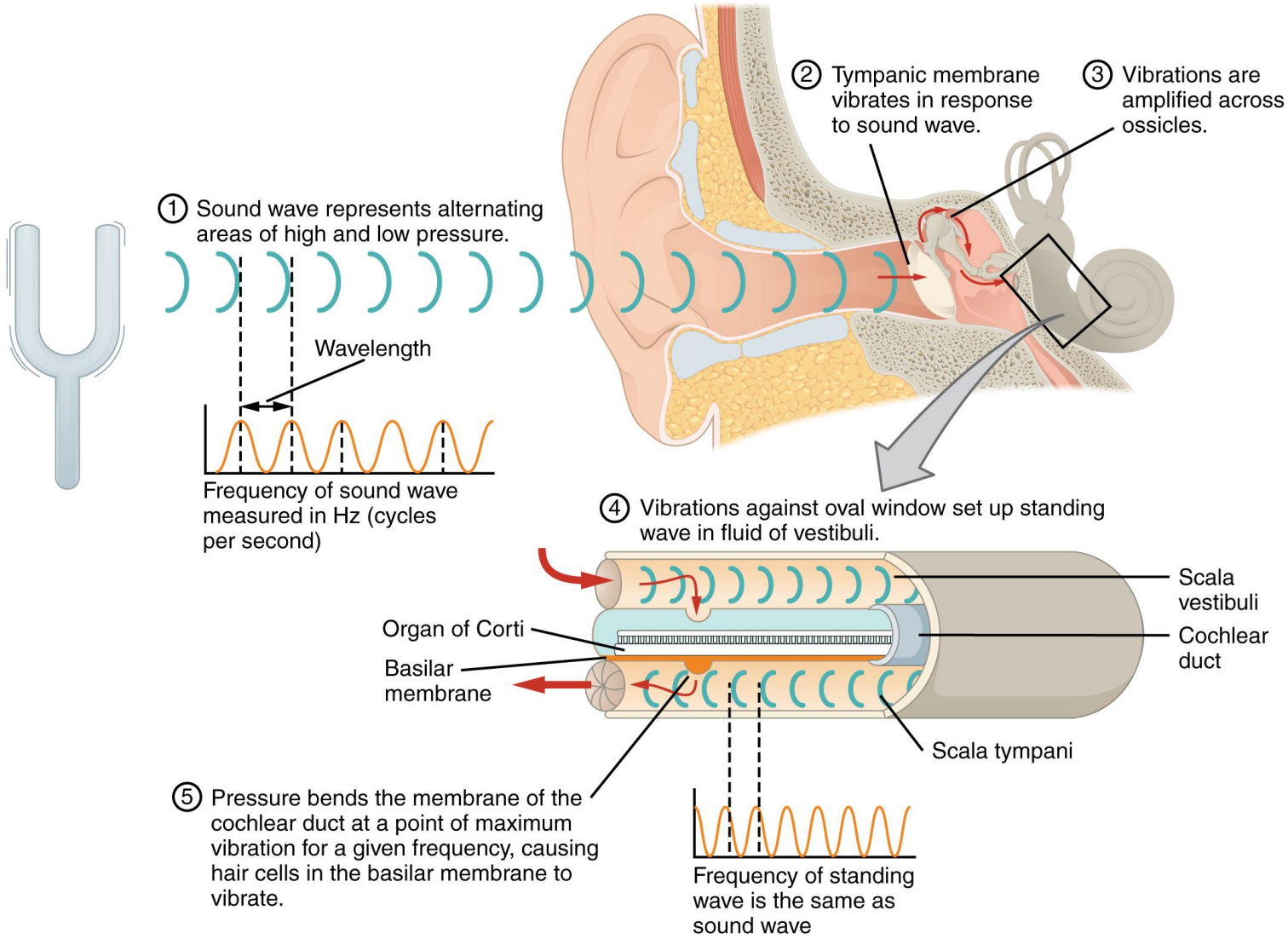


<u>The external ear</u> contains the auricle, ear canal, and tympanic membrane.

<u>The middle ear</u> contains the ossicles and is connected to the pharynx by the Eustachian tube.

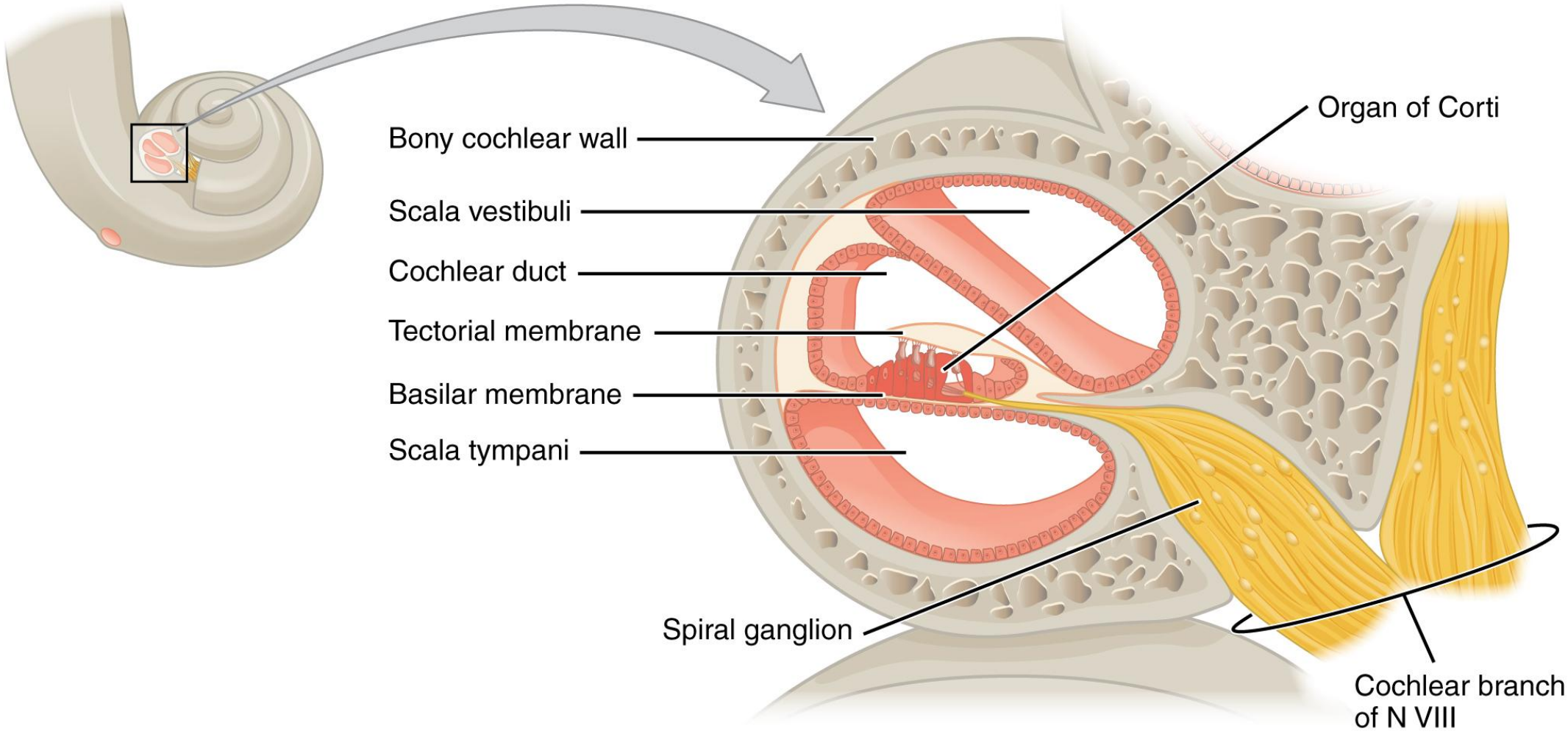
<u>The inner ear</u> contains the cochlea and vestibule, which are responsible for audition and equilibrium, respectively

Transmission of Sound Waves to Cochlea

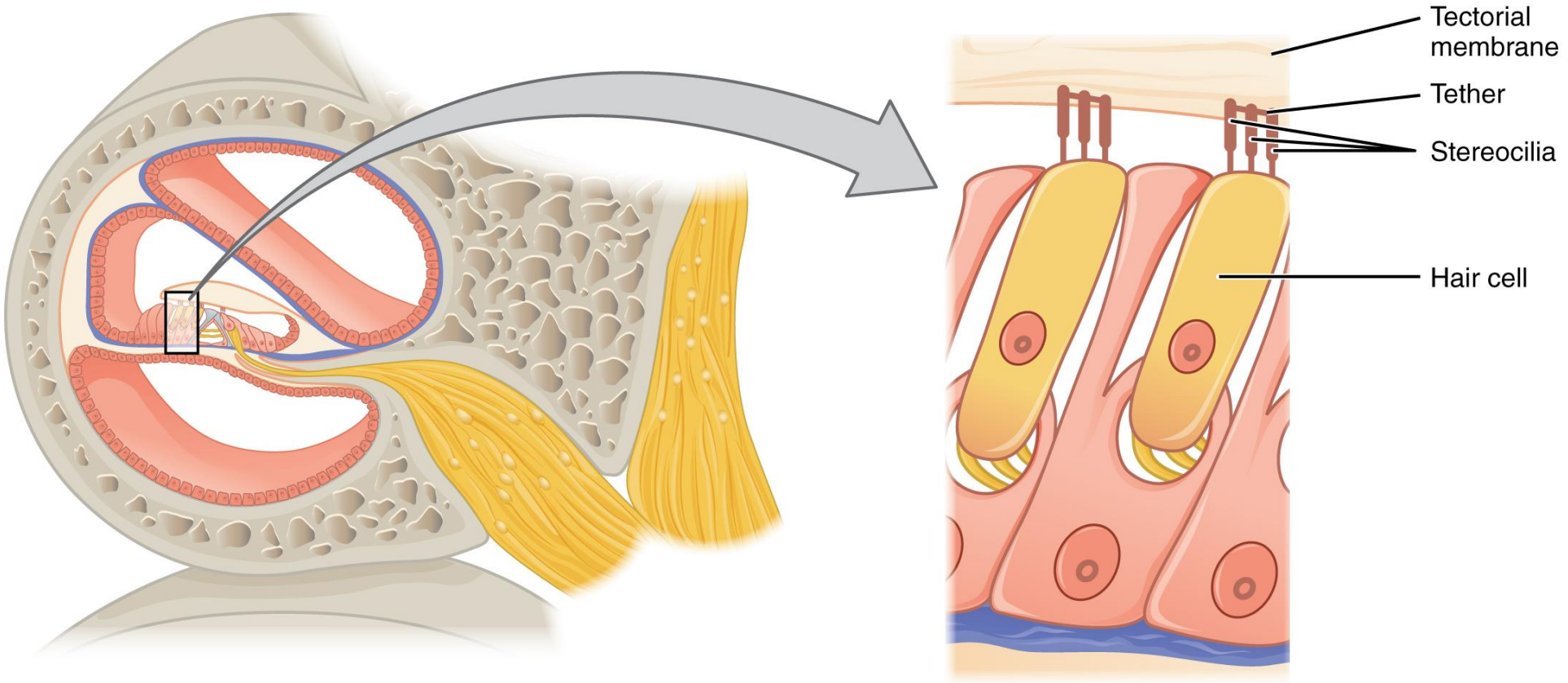


- *A sound wave causes the tympanic membrane to vibrate.*
- *This vibration is amplified as it moves across the malleus, incus, and stapes.*
- *The amplified vibration is picked up by the oval window causing pressure waves in the fluid of the scala vestibuli and scala tympani.*
- *The complexity of the pressure waves is determined by the changes in amplitude and frequency of the sound waves entering the ear.*

Cross Section of the Cochlea



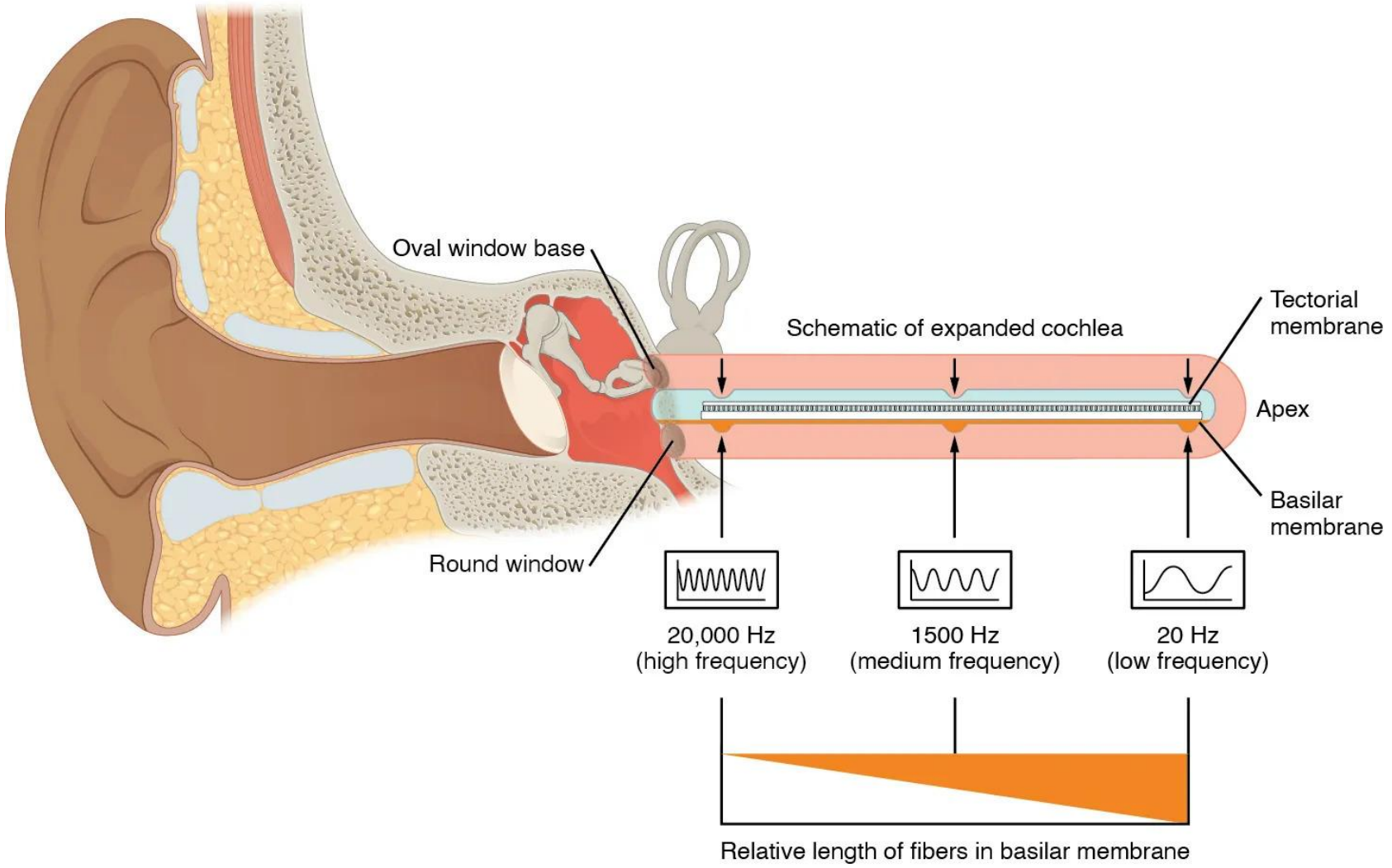
Hair Cell



The hair cell is a mechanoreceptor with an array of stereocilia emerging from its apical surface.

The stereocilia are tethered together by proteins that open ion channels when the array is bent toward the tallest member of their array, and closed when the array is bent toward the shortest member of their array.

Frequency Coding in the Cochlea



- *The standing sound wave generated in the cochlea by the movement of the oval window deflects the basilar membrane on the basis of the frequency of sound.*
- *Therefore, hair cells at the base of the cochlea are activated only by high frequencies, whereas those at the apex of the cochlea are activated only by low frequencies.*

Until the next time ☺