Vector Space Modeling

Transforming text into measurable data

Quick recap of previous lecture
(Sequence labeling problem)

POS Tagging
(Part-of-speech)

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:
Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

KEY:

N	=	Noun
V	=	Verb
P	=	Preposition
Adv	=	Adverb
Adj	=	Adjective

. . .

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

KEY:

NA	= No entity
SC	= Start Company
CC	= Continue Company
SL	= Start Location
CL	= Continue Location

Supervised learning

Sequence labelling problems is a supervised learning problem.

Training examples:

- $(x^{(i)}, y^{(i)})$, $i = 1 \dots m$
- $x_1^{(i)} \dots x_n^{(i)}$ represents the word sequence of i^{th} sentence having word length n
- $y_1^{(i)} \dots y_n^{(i)}$ represents the tag sequence of i^{th} sentence having word length n
- $x_j^{(i)}$ represents the j^{th} word in the i^{th} training example
- Our task is to learn a function $f : X \rightarrow Y$ from these training examples.
  - X refer to all the $x_1 \dots x_n$
  - Y refer to all the $y_1 \dots y_n$

Part-Of-Speech (POS) Tagging

- Individual words have **statistical preferences** for their part of speech.
  - E.g. ***quarter*** can be a noun or a verb, but is more likely to be a noun.
    - “A **quarter** of the cake is left.”
    - “Please **quarter** the apples before adding them to the pie.”

- The **context** has an important effect on the part of speech for a word.
  - The sequence **D N V** will be frequent in English,
    whereas the sequence **D V N** is much less likely.
    - The/D dog/N ran/V …

Conditional model

- $(x^{(i)}, y^{(i)})$, i = 1 … m

$$p(y|x)$$

$$f(x) = \arg\max_{y \in \mathcal{Y}} p(y|x)$$

Generative Models

- Our task is to learn a function from inputs x to labels $y = f(x)$. We assume training examples $(x^{(i)}, y^{(i)})$ for $i = 1 \dots n$.

- In the noisy channel approach, we use the training examples to estimate models $p(y)$ and $p(x|y)$. These models define a joint (generative) model

$$\boxed{p(x, y) = p(y)p(x|y)}$$

- Given a new test example x, we predict the label

$$\boxed{f(x) = \arg\max_{y \in \mathcal{Y}} p(y)p(x|y)}$$

Generative tagging models

- A finite set of vocabulary V
- A finite set of tags K
- S is a set of all tag-sequence pairs $\langle x_1^{(i)},\ldots, x_n^{(i)}, y_1^{(i)},\ldots,y_n^{(i)}\rangle$
  - $n >= 0$
  - $x_j \in V$ for $j = 1, \ldots, n$
  - $y_j \in K$ for $j = 1, \ldots, n$

- A generative tagging model is then a function *p* such that:

1. *For any* $\langle x_1 \ldots x_n, y_1 \ldots y_n \rangle \in \mathcal{S}$,

$$p(x_1 \ldots x_n, y_1 \ldots y_n) \geq 0$$

2. *In addition,*

$$\sum_{\langle x_1 \ldots x_n, y_1 \ldots y_n \rangle \in \mathcal{S}} p(x_1 \ldots x_n, y_1 \ldots y_n) = 1$$

$$\boxed{f(x_1 \ldots x_n) = \arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)}$$

Trigram Hidden Markov Models (Trigram HMMs)

- *A finite set of vocabulary V*
- *A finite set of tags K*
- *A parameter*

$$q(s|u,v)$$

Context

Transition probability

for any trigram (u, v, s) *such that* $s \in \mathcal{K} \cup \{STOP\}$, *and* $u, v \in \mathcal{K} \cup \{*\}$. *The value for* $q(s|u,v)$ *can be interpreted as the probability of seeing the tag* s *immediately after the bigram of tags* (u, v).

- *A parameter*

$$e(x|s)$$

Statistical preferences

Emission probability

for any $x \in \mathcal{V}$, $s \in \mathcal{K}$. *The value for* $e(x|s)$ *can be interpreted as the probability of seeing observation* x *paired with state* s.

Trigram Hidden Markov Models (Trigram HMMs)

- S is a set of all tag-sequence pairs $<x_1^{(i)},\ldots, x_n^{(i)}, y_1^{(i)},\ldots,y_n^{(i)}>$
    - $y_0 = y_{-1}$ = <s>/*/STRT
    - y_{n+1} = </s>/STOP/END
- Probability for any tag-sequence pairs $<x_1^{(i)},\ldots, x_n^{(i)}, y_1^{(i)},\ldots,y_n^{(i)}> \in S$:

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

POS tagging Example

Probability for a sentence "the dog laughs" with tag sequence "D N V STOP" with Trigram HMMs as:

$$\begin{aligned} p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) &= q(\mathtt{D}|*, *) \times q(\mathtt{N}|*, \mathtt{D}) \times q(\mathtt{V}|\mathtt{D}, \mathtt{N}) \times q(\mathtt{STOP}|\mathtt{N}, \mathtt{V}) \\ &\quad \times e(\textit{the}|\mathtt{D}) \times e(\textit{dog}|\mathtt{N}) \times e(\textit{laughs}|\mathtt{V}) \end{aligned}$$

Prior probability: $\boxed{q(\mathtt{D}|*, *) \times q(\mathtt{N}|*, \mathtt{D}) \times q(\mathtt{V}|\mathtt{D}, \mathtt{N}) \times q(\mathtt{STOP}|\mathtt{N}, \mathtt{V})}$

Transition probability

Conditional probability: $\boxed{e(\textit{the}|\mathtt{D}) \times e(\textit{dog}|\mathtt{N}) \times e(\textit{laughs}|\mathtt{V})}$

$\boxed{p(\textit{the dog laughs}|\mathtt{D}\ \ \mathtt{N}\ \ \mathtt{V}\ \ \mathtt{STOP})}$

Emission probability

Maximum-likelihood estimates

$$q(s|u,v) = \frac{c(u,v,s)}{c(u,v)}$$

Transition probability

$$e(x|s) = \frac{c(s \rightsquigarrow x)}{c(s)}$$

Emission probability

$c(u,v,s)$ - number of times the tag sequence (u,v,s) are seen in training data.
$c(s \rightsquigarrow x)$ - number of times the word x is seen paired with the tag s

Maximum-likelihood estimates

$$q(s|u,v) = \lambda_1 \times q_{ML}(s|u,v) + \lambda_2 \times q_{ML}(s|v) + \lambda_3 \times q_{ML}(s)$$

$$e(x|s) = \frac{c(s \leadsto x)}{c(s)}$$

$c(u,v,s)$ - number of times the tag sequence (u,v,s) are seen in training data.
$c(s \leadsto x)$ - number of times the word x is seen paired with the tag s

Decoding problem

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

$$\arg \max_{y_1 \cdots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

Decoding problem (Naive approach)

For a sentence: “*the dog barks*” to decode from a set of possible tags K = {D, N, V}

- The possible tag sequences are:

D	D	D	STOP
D	D	N	STOP
D	D	V	STOP
D	N	D	STOP
D	N	N	STOP
D	N	V	STOP

...

3^3 = 27 possible sequences

K^n possible sequences for sentence with n words and K tags.

Viterbi Algorithm

Backpointer: which previous tag (w) led to this score.

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u, v)$ and $e(x|s)$.
Definitions: Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \dots n$.
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

- For $k = 1 \dots n$,
  - For $u \in \mathcal{K}_{k-1}$, $v \in \mathcal{K}_k$,

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

$$bp(k, u, v) = \arg\max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

- Set $(y_{n-1}, y_n) = \arg\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} (\pi(n, u, v) \times q(\text{STOP}|u, v))$
- For $k = (n-2) \dots 1$,

$$y_k = bp(k+2, y_{k+1}, y_{k+2})$$

- **Return** the tag sequence $y_1 \dots y_n$

Log-linear tagging models (MEMMs)

- Referred to as Maximum Entropy Markov Models (MEMMs)

$$
\begin{aligned}
& P(Y_1 = y_1 \ldots Y_n = y_n | X_1 = x_1 \ldots X_n = x_n) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \ldots X_n = x_n, Y_1 = y_1 \ldots Y_{i-1} = y_{i-1}) \\
= & \prod_{i=1}^{n} P(Y_i = y_i | X_1 = x_1 \ldots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})
\end{aligned}
$$

$$
\boxed{\arg \max_{y_1 \ldots y_n \in \mathcal{Y}(n)} p(y_1 \ldots y_n | x_1 \ldots x_n)}
$$

Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \dots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Word/tag features:

$$f_{100}(h, y) = \begin{cases} 1 & \text{if } x_i \text{ is } \texttt{base} \text{ and } y = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \dots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Prefix and Suffix features:

$$f_{101}(h, y) = \begin{cases} 1 & \text{if } x_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

Features in Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

Trigram, Bigram and Unigram Tag features:

$$f_{103}(h, y) = \begin{cases} 1 & \text{if } \langle y_{-2}, y_{-1}, y \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

Trigram MEMMs

$$h_i = \langle y_{i-2}, y_{i-1}, x_1 \ldots x_n, i \rangle$$

$$f(h_i, y) \in \mathbb{R}^d \qquad y \in \mathcal{K}$$

$$P(Y_i = y_i | X_1 = x_1 \ldots X_n = x_n, Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1})$$

$$= \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

$$\boxed{p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}}$$

Trigram MEMMs

$$p(y_1 \dots y_n | x_1 \dots x_n) = \prod_{i=1}^{n} p(y_i | h_i; \theta)$$

$$p(y_i | h_i; \theta) = \frac{\exp\left(\theta \cdot f(h_i, y_i)\right)}{\sum_{y \in \mathcal{K}} \exp\left(\theta \cdot f(h_i, y)\right)}$$

$$\boxed{\arg \max_{y_1 \dots y_n \in \mathcal{Y}(n)} p(y_1 \dots y_n | x_1 \dots x_n)}$$

Parameter Estimation in Trigram MEMMs

Log-likelihood function

$$L(\theta) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \log p(y_i^{(k)} | h_i^{(k)}; \theta)$$

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} L(\theta)$$

Parameter Estimation in Trigram MEMMs

Regularized log-likelihood function

$$L(\theta) = \sum_{k=1}^{m} \sum_{i=1}^{n_k} \log p(y_i^{(k)} | h_i^{(k)}; \theta) - \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

A regularization term, which penalizes large parameter values

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} L(\theta)$$

$$p(y|x;\underline{w}) = \frac{\exp\left(\underline{w}\cdot\underline{\phi}(x,y)\right)}{\sum_{y'\in\mathcal{Y}}\exp\left(\underline{w}\cdot\underline{\phi}(x,y')\right)}$$

$$\boxed{p(y_i|h_i;\theta) = \frac{\exp\left(\theta\cdot f(h_i,y_i)\right)}{\sum_{y\in\mathcal{K}}\exp\left(\theta\cdot f(h_i,y)\right)}}$$

$$\boxed{L(\theta) = \sum_{k=1}^{m}\sum_{i=1}^{n_k}\log p(y_i^{(k)}|h_i^{(k)};\theta) - \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2}$$

$$\frac{\partial}{\partial w_j}L(\underline{w}) = \sum_i \phi_j(x_i,y_i) - \sum_i\sum_y p(y|x_i;\underline{w})\phi_j(x_i,y) \qquad \boxed{\theta \leftarrow \theta + \eta\cdot\nabla_\theta L(\theta)}$$

Observed feature count − Expected feature count under current model

Decoding (Viterbi Algorithm)

Input: A sentence $x_1 \dots x_n$. A set of possible tags $\mathcal{K}$. A model (for example a log-linear model) that defines a probability

$$p(y|h;\theta)$$

for any h, y pair where h is a history of the form $\langle y_{-2}, y_{-1}, x_1 \dots x_n, i \rangle$, and $y \in \mathcal{K}$.
Definitions: Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \dots n$.
Initialization: Set $\pi(0, *, *) = 1$.
Algorithm:

- For $k = 1 \dots n$,
  - For $u \in \mathcal{K}_{k-1}, v \in \mathcal{K}_k$,

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} \left(\pi(k-1, w, u) \times p(v|h;\theta)\right)$$

$$bp(k, u, v) = \arg\max_{w \in \mathcal{K}_{k-2}} \left(\pi(k-1, w, u) \times p(v|h;\theta)\right)$$

where $h = \langle w, u, x_1 \dots x_n, k \rangle$.

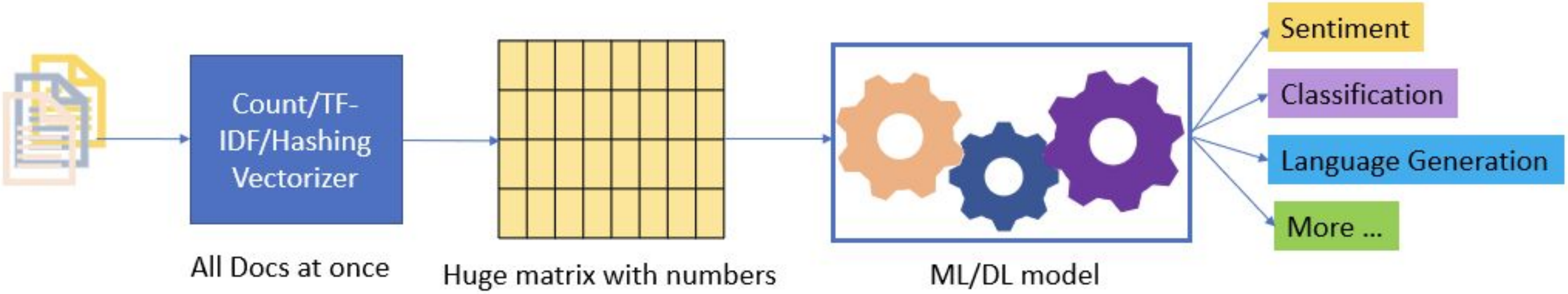
- Set $(y_{n-1}, y_n) = \arg\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} \pi(n, u, v)$
- For $k = (n-2) \dots 1$,

$$y_k = bp(k+2, y_{k+1}, y_{k+2})$$

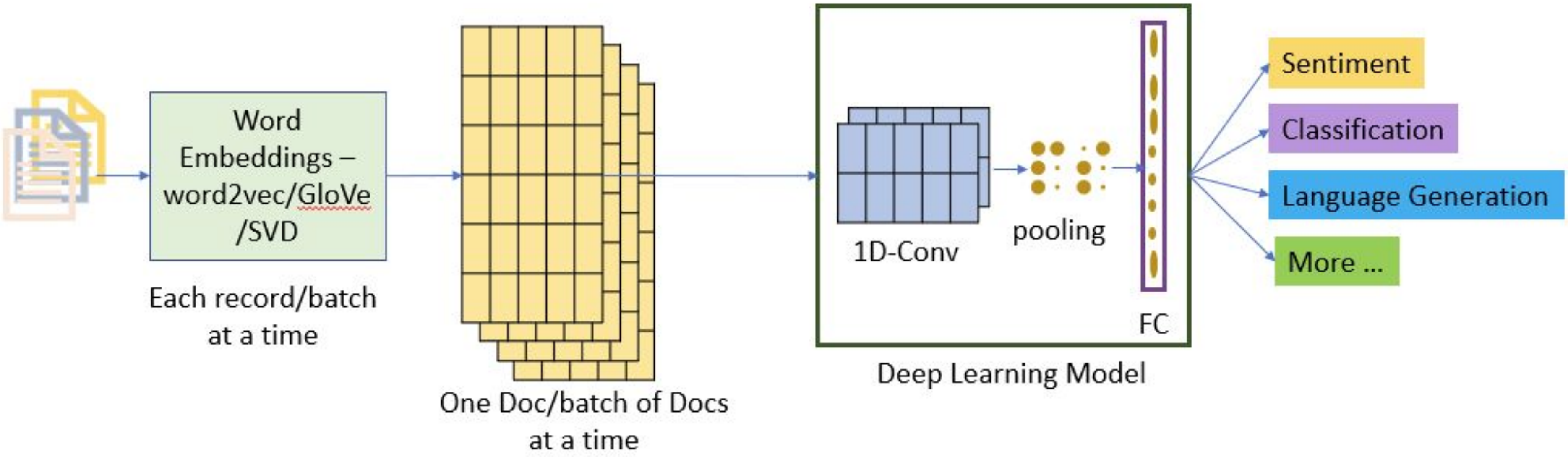
- **Return** the tag sequence $y_1 \dots y_n$

Vector Space Modeling

Natural Language Processing Traditional Modules



ML model for TEXT – With Deep Learning



Semantics

- Study of meaning in language such as how words, phrases, and sentences convey ideas.
- Concerned with:
  - **Word meaning** (*lexical semantics*)
  - **Sentence meaning** (*compositional semantics*)
  - **Contextual meaning** (discourse, pragmatics)
- Answers questions like:
  - How do we know *bank* means “river bank” or “financial bank” in a sentence?
  - Why do *dog* and *puppy* feel semantically closer than *dog* and *car*?
- Understanding *meaning relationships* in language is essential for communication.
- Acts as the **linguistic foundation** for many NLP tasks.

Word → Phrase → Sentence → Context → Meaning

Why Semantics Matters in NLP?

- **Semantics enables computers to capture the meaning behind language, not just process text as symbols.**
- Accurate meaning representation improves **contextual understanding** and reduces misunderstanding and ambiguity.
- Most NLP tasks (**search engine**, **machine translation**, **chatbots**) require interpreting what users *intend* to convey, not just what they say literally.
- Semantic modeling powers applications like **sentiment analysis**, **information extraction**, **summarization**, and many more NLP applications.
- Without semantics, NLP systems fail to bridge the gap between string processing and language understanding.

Lexical Semantics

- **Semantics**: Linguistic or logical study of meaning
- **Lexical semantics**: Linguistic study of word meaning
- **Lemma**: 'Dictionary form' of a word
  - A lemma can have many **word senses**, each representing a different meaning or concept
  - **Word sense disambiguation**: Understanding the correct meaning of a word in context.
- **Wordform**: A specific inflected or derived form of a lemma
  - E.g., `<sing>` is a lemma, `<sings>`, `<singing>` are wordforms

Word Relationships

- **Synonym**: Words with identical or nearly identical senses
  - `<cat>` and `<kitten>`: are synonyms
- **Word Similarity**: Words with similar relationships, not always synonyms
  - `<king>` and `<queen>`: related to royal leaders
- **Word Relatedness / Association**: Words that share a contextual link, but are not necessarily similar
  - `<tea>` and `<cup>`: related by context (you need cup to drink the tea)
- **Semantic field**: A group of related words from a particular domain
  - `<Weather>` and `<Climate>`
- **Topic models**: Tools that can learn associations between words automatically
  - `<player>, <game>,` and `<score>`

From Lexical Semantics to Vector Semantics

- Lexical semantics (qualitative meaning analysis)
  - Focuses on word meaning
  - Deals with senses, synonyms, associations
  - Provides a deep understanding of individual word meanings and their relationships within a language's structure.
  - **Limitations:** Can be time-consuming and labor-intensive to create and maintain comprehensive lexical resources.

- Vector semantics (quantitative numerical encoding of meaning)
  - Represents word meaning into numerical vectors
  - Words with similar meanings are positioned closer to each other in a multi-dimensional vector space.
  - Vector semantics, particularly through techniques like word embeddings (e.g., Word2Vec, GloVe)
  - **Advantages:** Captures Contextual Similarity, Scalability, Generalization

Why Vector Semantics?

- Traditional lexical semantics is qualitative and human-driven
- **Vector semantics** captures word meaning as points in a high-dimensional space
- Enables:
  - Quantitative comparison of word meanings
  - Discovery of unforeseen word relationships
  - Use in various **NLP** tasks such as search engine, machine translation, sentiment analysis, and so on.

Vector Semantics

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

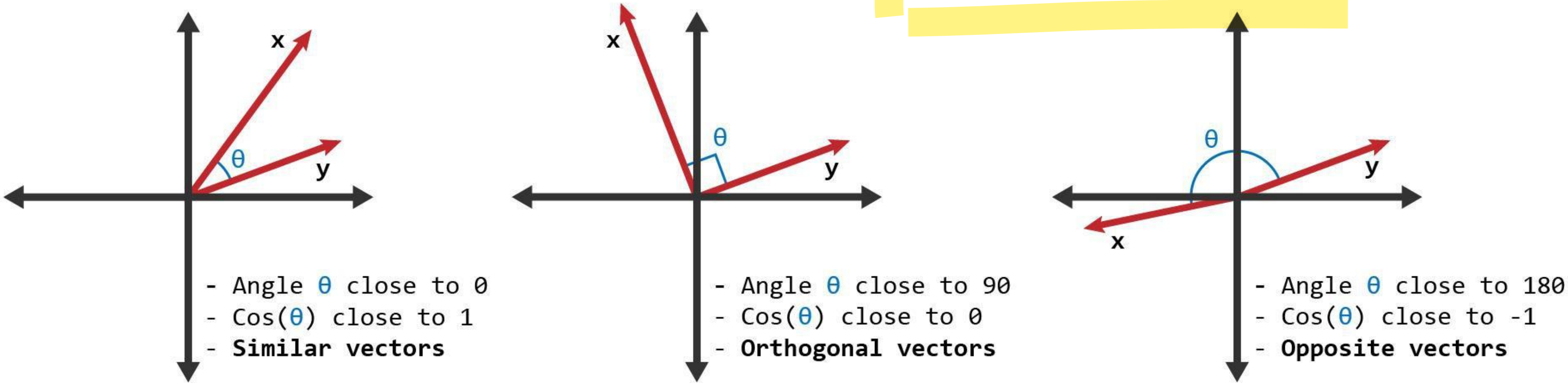


Image source: https://www.learndatasci.com/glossary/cosine-similarity/

How Vector Semantics Works

- Each word = a vector (list of numbers)
- Vectors computed from real data (corpus statistics)
- Words used in similar contexts have similar meanings [Distributional Hypothesis]
- Techniques:
  - **Co-occurrence matrices** (e.g., Term-Context, Term-document matrices)
  - **Word embeddings** (e.g., Word2Vec, GloVe, BERT)

Types of Vector Space Models (Co-occurrence)

- **Matrix Types in VSMs**
  - Term–Document, Word–Context, Pair–Pattern
  - Generating a matrix (VSM) requires
    1. Linguistic processing
    2. Mathematical processing
  - Choice of matrix is important for linguistics and mathematical processing.
  - **Not All Matrices Are VSMs**
    1. In a **VSM**, the values in the matrix (the cell entries) are typically **derived from statistical information derived from actual language use.**
    2. The vector does not attempt to capture the structure in the phrases, sentences, paragraphs, and chapters of the document.

Useful terminology

Sentence	Unit of written language
Utterance	Unit of spoken language
Word Form	The inflected form as it actually appears in the corpus *e.g.* “said”
Lemma	An abstract form, shared by word forms having the same stem, part of speech, word sense *e.g.* “say” Stands for the class of words with same stem
Function words	Indicate the grammatical relationship between terms but have little topical information *e.g.* “by”
Types	Number of distinct words in a corpus *i.e.* vocabulary size
Tokens	Total collection of all words

Types and Tokens

- A token is a single instance of a symbol, whereas
- A type is a general class of tokens

Ever tried. Ever failed.
No matter. Try again.
Fail again. Fail better.

In the above example, there are two tokens of the type *Ever*, two tokens of the type *again*, and two tokens of the type *Fail*.

Two possible types of term-document matrix representation:

- Token–document matrix (Boolean value)
- Type–document matrix (Frequency value)

Term-Document matrix

	Ever tried. Ever failed.	No matter. Try again.	Fail again. Fail better.
Ever	1	0	0
tried	1	0	0
Ever	1	0	0
failed	1	0	0
No	0	1	0
matter	0	1	0
Try	0	1	0
again	0	1	0
Fail	0	0	1
again	0	0	1
Fail	0	0	1
better	0	0	1

Token–document matrix

	Ever tried. Ever failed.	No matter. Try again.	Fail again. Fail better.
Ever	2	0	0
tried	1	0	0
failed	1	0	0
No	0	1	0
matter	0	1	0
Try	0	1	0
again	0	1	1
Fail	0	0	2
better	0	0	1

Type–document matrix

Term-Document matrix

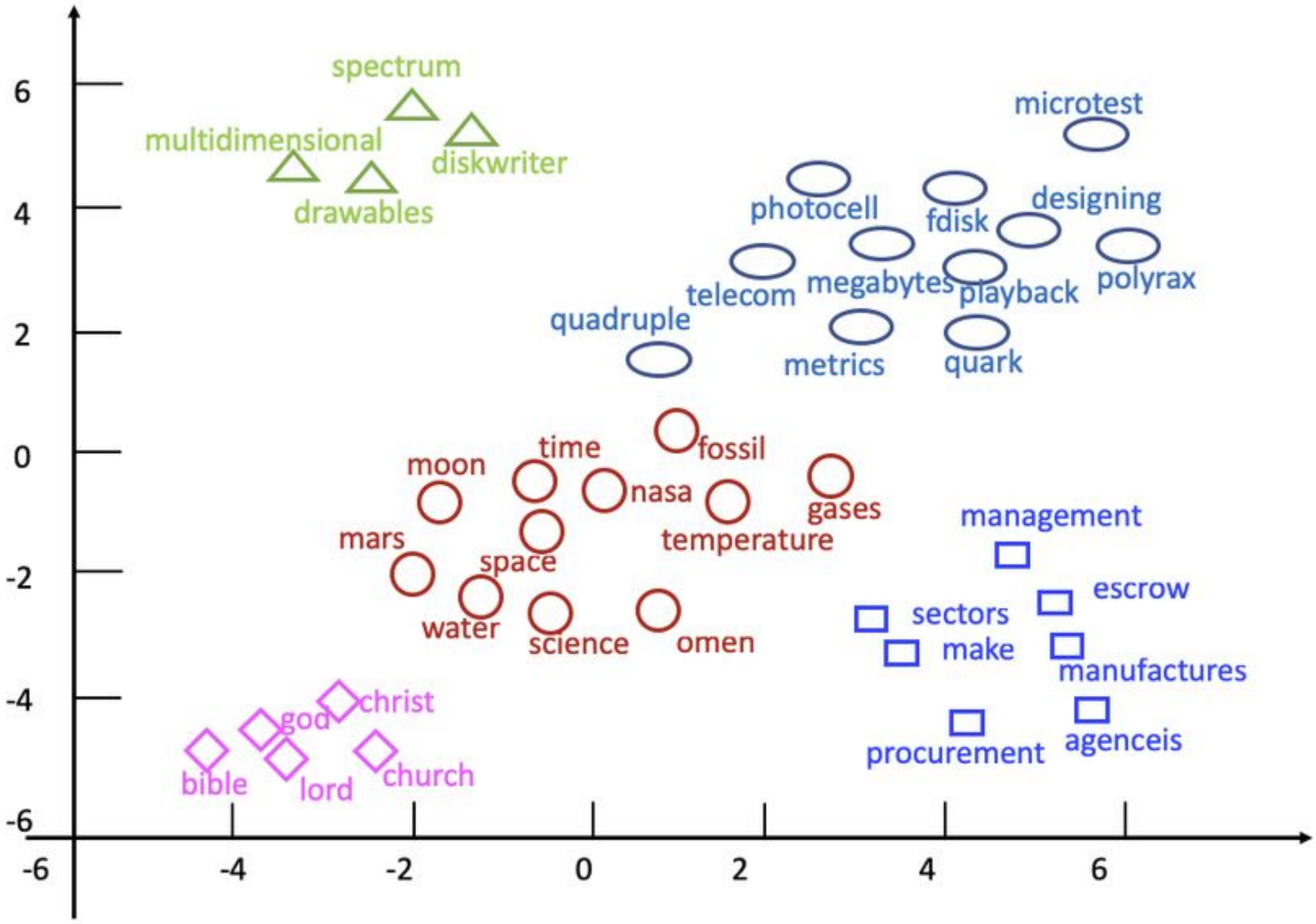
- **Sparse representation:** Most entries are zero because each document contains only a small fraction of the vocabulary.
- **Bag-of-Words Assumption:**
  - Ignores grammar and word order; **only word occurrence counts** matter.
  - The structure within phrases, sentences, paragraphs, or chapters is **not captured**.
- *Bag-of-Words hypothesis*: estimate a document's relevance to a query by comparing their word-based representations (vectors).
- **Topic Similarity:**
  - If two documents discuss **similar topics**, their column vectors in the matrix will show **similar patterns of term frequencies** (raw or weighted, e.g., TF-IDF).
  - This allows measuring document similarity using cosine similarity or other distance metrics.

Word-Context matrix

	get	see	use	hear	eat	kill
knife	0.027	-0.024	0.206	-0.022	-0.044	-0.042
cat	0.031	0.143	-0.243	-0.015	-0.009	0.131
dog	-0.026	0.021	-0.212	0.064	0.013	0.014
boat	-0.022	0.009	-0.044	-0.040	-0.074	-0.042
cup	-0.014	-0.173	-0.249	-0.099	-0.119	-0.042
pig	-0.069	0.094	-0.158	0.000	0.094	0.265
banana	0.047	-0.139	-0.104	-0.022	0.267	-0.042

Word-Context matrix

- Represent each word by the **contexts** in which it appears.
  a. Each **row** = a word (term).
  b. Each **column** = a context feature (nearby words, syntactic relations, etc.).
  c. Each **cell value** = frequency or weighted score (TF-IDF, or PMI) of the word in that context.
- The similarity of word vectors reflects the **semantic similarity** of the words.
  - **Firth (1957):** *"You shall know a word by the company it keeps."*
  - **Distributional Hypothesis:** Words that occur in similar contexts tend to have similar meanings.
  - Example: "doctor" and "physician" often share contexts like *hospital*, *patient*, *treatment*.
- **Key Application: (Attributional Similarity**)
  - Measures likeness of *properties or meanings* of individual words (e.g., synonyms: "doctor" ≈ "physician").
  - Used in **synonym detection**, **thesaurus building**, and **semantic clustering**.



spectrum
multidimensional
diskwriter
drawables
microtest
photocell
fdisk
designing
telecom
megabytes
playback
polyrax
quadruple
metrics
quark
time
fossil
moon
nasa
gases
mars
temperature
space
water
science
omen
management
escrow
sectors
make
manufactures
procurement
agenceis
christ
god
bible
lord
church
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6

Pair-Pattern matrix

- Capture **relational similarity** (how alike the *relationships* between two pairs of words are).
- Pair–pattern matrix **Structure**:
  - **Rows:** Word pairs (e.g., *mason : stone*, *carpenter : wood*) [*artisan : material*].
  - **Columns:** Lexico-syntactic patterns in which the pair co-occurs (e.g., “X cuts Y”, “X works with Y”).
  - **Cell values:** Frequency or weighted score (e.g., PMI) of the pair occurring in that pattern.
- Given a pattern like “X solves Y”, the model can find semantically similar patterns:
  - “Y is solved by X”
  - “Y is resolved in X”
  - “X resolves Y”
- This demonstrates the system’s ability to match **different surface forms** expressing the same relationship.
- **Latent Relation Hypothesis:** *Word pairs that co-occur in similar patterns tend to share the same semantic relationship.* E.g., *(doctor : patient)* and *(teacher : student)* both fit the relation “X helps Y / tends to Y”.
- **Applications:**
  - Analogy solving (e.g., “Paris : France :: Tokyo : Japan”).
  - Relation extraction from large corpora.
  - Semantic search for relationships, not just single words.

Types of Similarities in Vector Space Models

- **Word–Document (Term–Document) Matrices** → Measure **topical similarity**
  - Compare documents (or queries and documents) based on shared vocabulary and themes.
  - Example: Two articles about climate change will rank as similar even if wording differs.
- **Word–Context Matrices** → Measure **attributional similarity**
  - Compare individual words by the *properties or meanings* they share from similar usage contexts.
  - Example: *doctor* ≈ *physician* (same semantic attributes, different surface forms).
- **Pair–Pattern Matrices** → Measure **relational similarity**
  - Compare *relationships* between pairs of words based on similar pattern usage.
  - Example: (*mason : stone*) ≈ (*carpenter : wood*) → both express “artisan : material” relations.

Core Hypotheses of VSM

- **Statistical Semantics Hypothesis**: Patterns of word usage reflect meaning.
  - In a large corpus, the word *"bark"* co-occurs with *"dog"* and *"tree"* in different contexts. By studying these usage patterns, a model can infer that *"bark"* has multiple meanings (animal sound vs. tree covering).
- **Distributional Hypothesis**: Words in similar contexts have similar meanings.
  - If *"doctor"* and *"physician"* often appear in contexts like *"hospital," "patients,"* and *"treatment,"* the model can infer they have related meanings.
- **Bag-of-Words Hypothesis**: Word frequency indicates relevance to queries.
  - In search engine or document ranking, a document is considered more relevant to "machine learning" if those words appear frequently, regardless of the word order or structured in the text.
- **Latent Relation Hypothesis**: Word pairs in similar patterns have similar relations.
  - If *"Paris"* is to *"France"* as *"Tokyo"* is to *"Japan"* in many sentences, the model learns the *capital–country* relationship and can apply it to unseen pairs like *"Delhi–India"*.

Building the Vector Spaces

- **Data Sources & Preprocessing**
  - Corpora, tokenization, normalization, lemmatization, stopword removal
- **Weighting Schemes** [*give more weight to surprising events and less weight to expected events*]
  - TF-IDF, PMI (Pointwise Mutual Information)
- **Matrix Smoothing & Dimensionality Reduction**
  - Motivation: noise reduction, uncover latent structure
  - Singular Value Decomposition, PCA
- **Similarity Measures**
  - Cosine similarity, Euclidean distance, correlation

TF-IDF

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Pointwise Mutual Information (PMI)

- **PMI** measures the association between two words based on how often they co-occur versus how often they would by chance.

$$\mathrm{PMI}(w_1,w_2) = \log_2 \frac{P(w_1,w_2)}{P(w_1)P(w_2)}$$

  where $P(w_1,w_2)$ [p_{12}] is the probability of both words appearing together, and $P(w_1)$, $P(w_2)$ are their individual probabilities.

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

- If w_1 and w_2 are statistically independent, then $P(w_1,w_2) = P(w_1)*P(w_2)$,
  - Thus $\mathrm{PMI}(w_1,w_2)$ is zero, i.e. $pmi_{12} = 0$
- If the word w_i is unrelated to the context w_j, we may find that pmi_{ij} is negative.

Pointwise Mutual Information (PMI)

- **PMI** measures the association between two words based on how often they co-occur versus how often they would by chance.

$$\mathrm{PMI}(w_1,w_2) = \log_2 \frac{P(w_1,w_2)}{P(w_1)P(w_2)}$$

  where $P(w_1,w_2)$ [p_{12}] is the probability of both words appearing together, and $P(w_1)$, $P(w_2)$ are their individual probabilities.
- If w_1 and w_2 are statistically independent, then $P(w_1,w_2) = P(w_1)*P(w_2)$,
  - Thus $\mathrm{PMI}(w_1,w_2)$ is zero, i.e. $pmi_{12} = 0$
- If the word w_i is unrelated to the context w_j, we may find that pmi_{ij} is negative.

PPMI is designed to give a high value to x_{ij} when there is an interesting semantic relation between w_i and w_j, otherwise 0 indicating uninformative.

$$x_{ij} = \begin{cases} \mathrm{pmi}_{ij} & \text{if } \mathrm{pmi}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Feature Selection in Term-Context Matrices

- Term-context matrices can be **huge** — thousands of possible context features lead to **high dimensionality**, **noise**, and **sparsity**.
- **Feature selection** reduces size, improves efficiency, and focuses on the most **semantically informative contexts**.
- **Key Feature Selection Methods**
  a. **Linguistic Filters** → keep informative POS (nouns, verbs), syntactic dependencies; remove stopwords.
  b. **Limit Context Window** → e.g., 2–5 words around target term to capture local associations.
  c. **Frequency Thresholds** → drop rare or overly common contexts to reduce sparsity.
  d. **Statistical Weighting** →
     i. **TF-IDF**: keeps important but not overly frequent contexts.
     ii. **PMI**: captures strong co-occurrence associations.
  e. **Domain Filtering / Contextual Embeddings** → focus on task-specific or usage-driven contexts.

Building the Vector Spaces

- **Data Sources & Preprocessing**
  - Corpora, tokenization, normalization, lemmatization, stopword removal
- **Weighting Schemes** [*give more weight to surprising events and less weight to expected events*]
  - TF-IDF, PMI (Pointwise Mutual Information)
- ➢ **Matrix Smoothing & Dimensionality Reduction**
  - Motivation: noise reduction, uncover latent structure
  - Singular Value Decomposition, PCA
- **Similarity Measures**
  - Cosine similarity, Euclidean distance, correlation

Singular Value Decomposition (SVD)

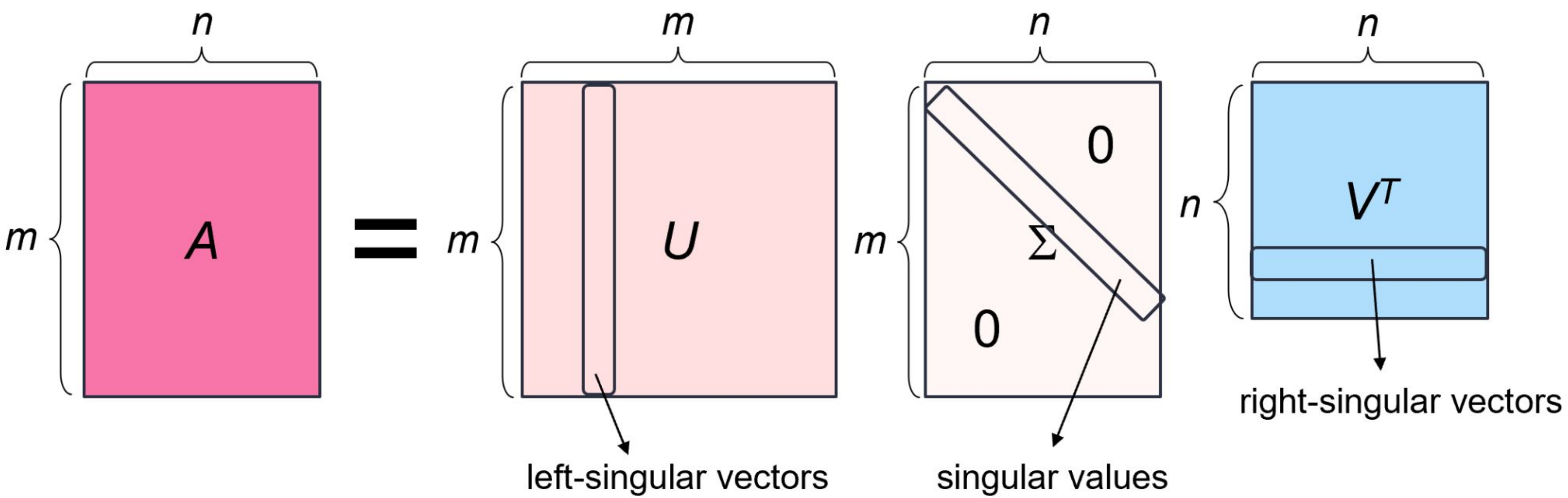


Image source: https://towardsdatascience.com/singular-value-decomposition-svd-demystified-57fc44b802a0/

Truncated Singular Value Decomposition (SVD)

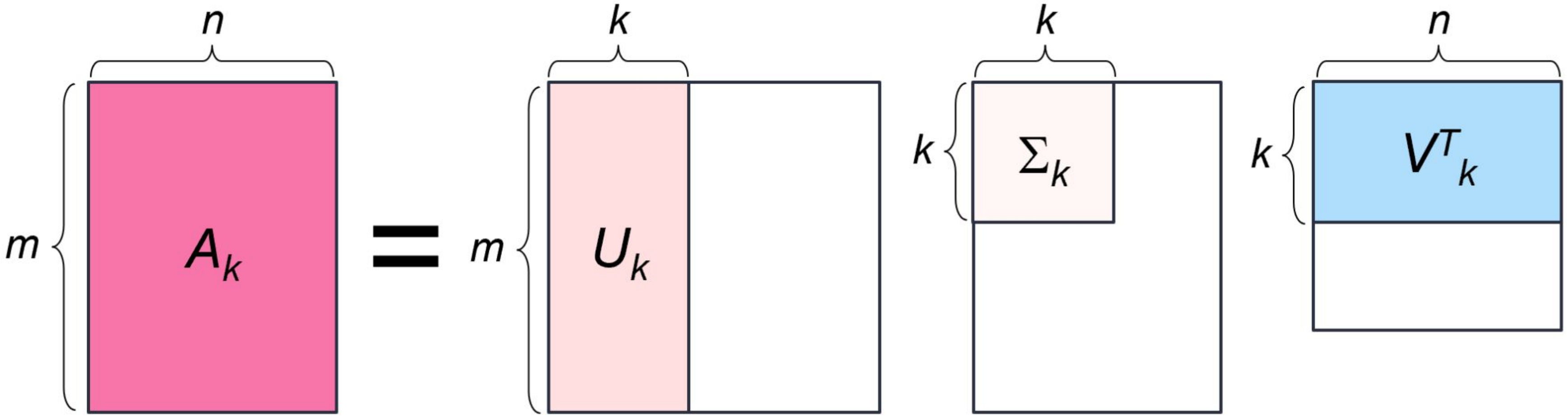


Image source: https://towardsdatascience.com/singular-value-decomposition-svd-demystified-57fc44b802a0/

Singular Value Decomposition (SVD)

- Computing similarity between all pairs of high-dimensional vectors (e.g., documents, words) is computationally expensive and noisy due to sparsity.
- **SVD Overview**: Decomposes a matrix **X** into three matrices: $X=U\Sigma V^T$
  - **U** = orthogonal matrix (left singular vectors, e.g., document vectors)
  - **Σ** = diagonal matrix of singular values (importance/strength of each latent dimension)
  - $\mathbf{V^T}$ = orthogonal matrix (right singular vectors, e.g., term vectors)
  - If **X** is rank **r**, then **Σ** also has rank **r** (r non-zero singular values).
- **Truncated SVD:**
  - Keep only the top **k** singular values and corresponding vectors: $X_k \approx U_k \Sigma_k V_k^T$ where $k<r$
  - Reduces dimensionality while preserving most variance in the data.

Singular Value Decomposition (SVD)

Benefits:

- Captures **latent meaning** and hidden semantic structure.
- **Sparsity reduction** (more compact, dense vector representations).
- **Noise reduction** (filters out less significant dimensions).
- Models **higher-order co-occurrence** (indirect word relationships).

Applications:

- **Document similarity:** Truncated SVD is known as **Latent Semantic Indexing (LSI)**.
- **Word similarity:** Truncated SVD is known as **Latent Semantic Analysis (LSA)**.

Wider reading (Applications):

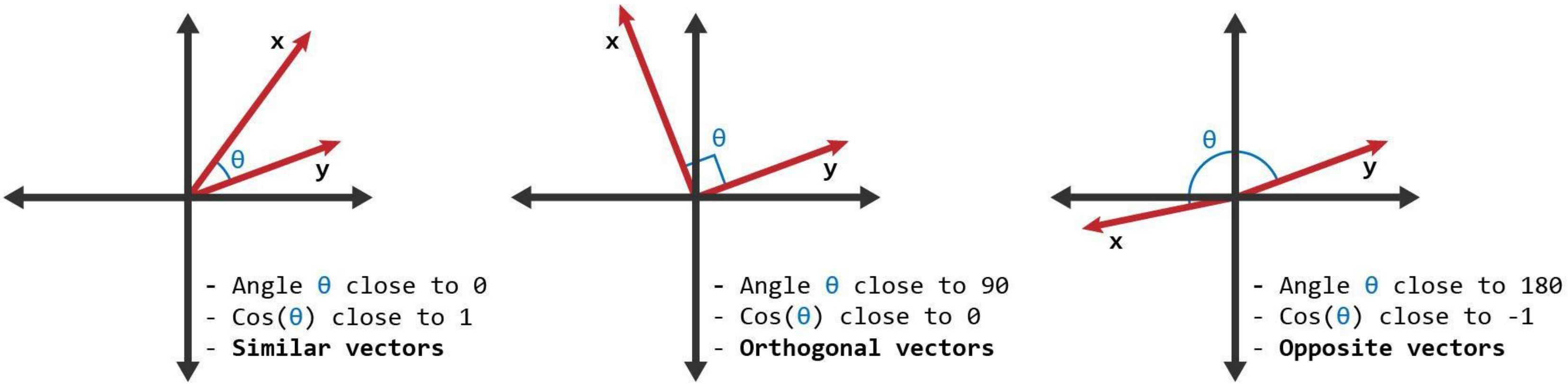
- Latent Semantic Analysis https://zilliz.com/glossary/latent-semantic-analysis-(lsa)
- Latent Semantic Indexing https://www.meilisearch.com/blog/latent-semantic-indexing

Building the Vector Spaces

- **Data Sources & Preprocessing**
  - Corpora, tokenization, normalization, lemmatization, stopword removal
- **Weighting Schemes** [*give more weight to surprising events and less weight to expected events*]
  - TF-IDF, PMI (Pointwise Mutual Information)
- **Matrix Smoothing & Dimensionality Reduction**
  - Motivation: noise reduction, uncover latent structure
  - Singular Value Decomposition, PCA
- ➢ **Similarity Measures**
  - Cosine similarity, Euclidean distance, correlation

Comparing Vectors in Vector Space Models (VSMs)

- **Purpose:** Quantify how similar or different words, documents, or other text units are, based on their vector representations.
- Common comparison measures:
  - **Cosine similarity** → Measures angle between vectors (–1 to +1)
    - Ignores vector length, focuses on direction → ideal for high-dimensional text data.
  - **Euclidean distance** → Measures straight-line distance between points.
    - Sensitive to vector magnitude, works best when data is normalized.

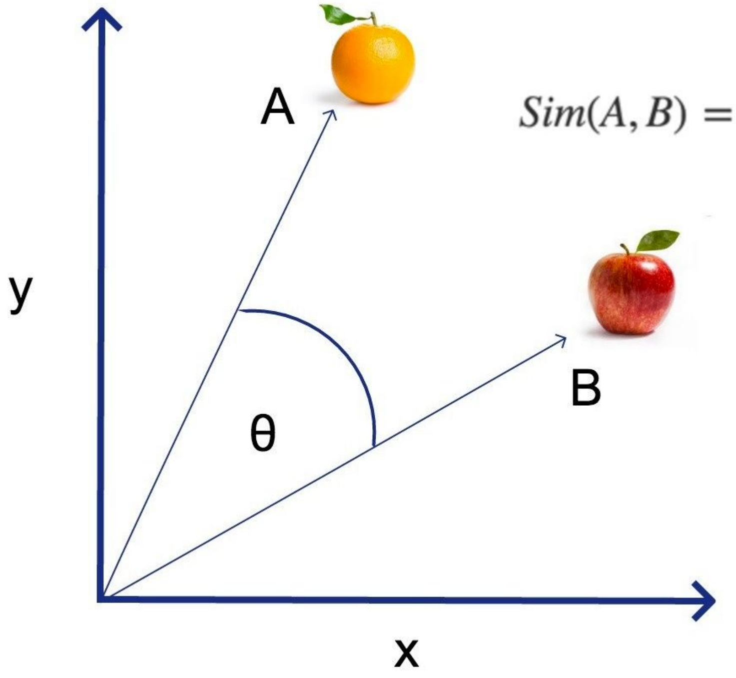


Comparing Vectors in Vector Space Models (VSMs)

- **Purpose:** Quantify how similar or different words, documents, or other text units are, based on their vector representations.
- Common comparison measures:
  - **Cosine similarity** → Measures angle between vectors (–1 to +1)
    - Ignores vector length, focuses on direction → ideal for high-dimensional text data.
  - **Euclidean distance** → Measures straight-line distance between points.
    - Sensitive to vector magnitude, works best when data is normalized.

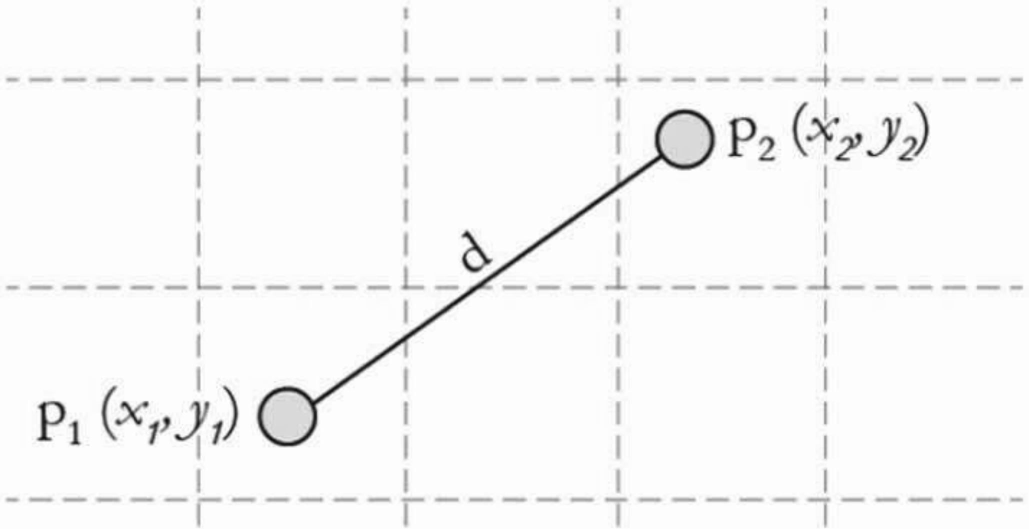
Note: In practice, **cosine similarity** is the most widely used for comparing frequency-based vectors (raw or weighted like TF-IDF) because it's scale-independent and works well for sparse data.

Cosine Similarity

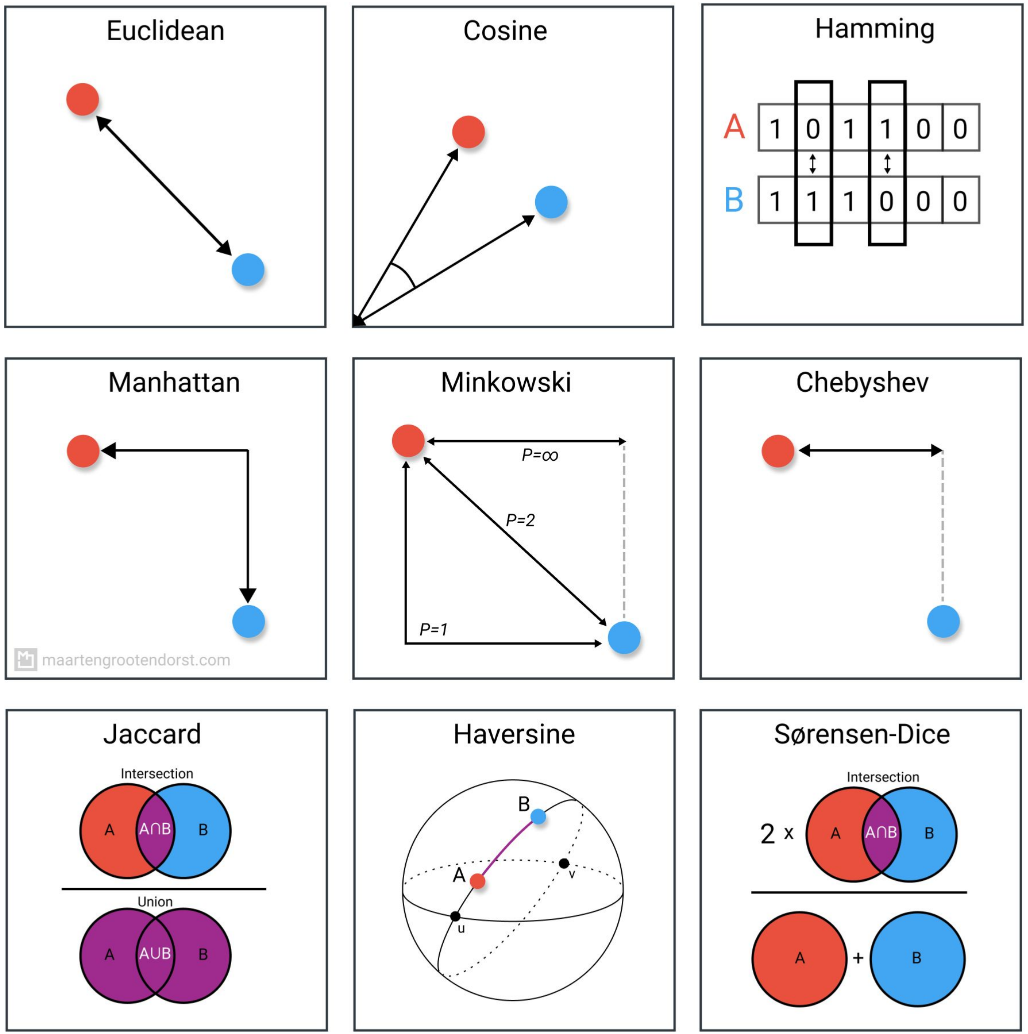


$$Sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Euclidean Distance



$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$


Euclidean
Cosine
Hamming
A 1 0 1 1 0 0
B 1 1 1 0 0 0
Manhattan
maartengrootendorst.com
Minkowski
P=∞
P=2
P=1
Chebyshev
Jaccard
Intersection
A A∩B B
Union
A A∪B B
Haversine
B
A
v
u
Sørensen-Dice
Intersection
2 x A A∩B B
A + B

Applications

- **Information Retrieval**
  - Search engine ranking with vector spaces (Bag-of-Words Hypothesis)
- **Synonym Detection**
  - TOEFL synonym test results (Distributional Hypothesis)
  - Word similarity evaluation benchmarks (Distributional Hypothesis)
- **Analogy Solving**
  - SAT analogy questions (Latent Relation, Distributional Hypothesis)
- **Relation Extraction** (Latent Relation Hypothesis)
  - Finding semantic relations (cause-effect, part-whole, etc.)
- **Topic Modeling & Semantic Clustering**
  - Linking to semantic fields and latent structures (Bag-of-Words, Distributional Hypothesis)

Limitations & Extensions

- **Limits of Classical VSM**
  - Ignores word order, syntax, ambiguity resolution
- **From VSM to Word Embeddings**
  - Connection to Word2Vec, GloVe
  - Neural embeddings learn similar vector spaces
- **Beyond Words: Sentence & Document Embeddings**
  - Averaging, Paragraph Vector, transformer encoders (SBERT)

Wider Reading

- Principal Component Analysis
    - https://builtin.com/data-science/step-step-explanation-principal-component-analysis
    - https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/
    - https://medium.com/@notsokarda/pca-vs-svd-simplified-32c5c753998
- From Frequency to Meaning: Vector Space Models of Semantics (2010)
    - https://www.jair.org/index.php/jair/article/view/10640
- Measuring praise and criticism: Inference of semantic orientation from association (2003)
    - https://dl.acm.org/doi/pdf/10.1145/944012.944013

Thank you for your attention!