Introduction to Semantic Processing

Definition

Semantic Processing in Natural Language Processing (NLP) is the stage where a system moves **beyond structure and syntax** to understand the **meaning** of words, phrases, and sentences.

It involves interpreting text by analysing relationships between words, their context, and the overall message being conveyed.

The goal of semantic processing is to enable machines to interpret language as humans do — not just how words are written, but what they *mean*.

Motivation

While **syntactic parsing** ensures grammatical correctness, it doesn't guarantee meaning comprehension.

For example:

"The dog chased the ball." vs. "The ball chased the dog."

Both are syntactically correct, but semantically, the roles of *dog* and *ball* make only the first sentence meaningful.

Semantic processing provides this layer of **interpretive understanding**, allowing systems to:

- Extract factual information
- Resolve ambiguities
- Support reasoning and question answering
- Enable natural conversational systems

Levels of Meaning Representation

Level	Focus	Example
Word Level	Lexical semantics – meaning of individual words	"Bank" (river bank vs. financial bank)

Sentence Level	Compositional semantics – how meanings combine	“Riya bought a car.”
Discourse Level	Meaning across multiple sentences	“She loves it.” → refers to “car”

Semantic analysis progresses through these levels, integrating them for contextual understanding.

Core Sub-tasks in Semantic Processing

Subtask	Purpose	Example
Word Sense Disambiguation (WSD)	Identify the correct meaning of ambiguous words.	“He sat by the *bank*.” → river bank
Semantic Role Labelling (SRL)	Determine *who did what to whom*.	“Riya gave a book to Meera.” → (Agent=Riya, Theme=book, Recipient=Meera)
Named Entity Recognition (NER)	Identify proper nouns and classify them.	“Microsoft”, “India”, “July 2024”
Semantic Similarity	Measure how similar meanings are.	“doctor” ≈ “physician”
Coreference Resolution	Identify when two expressions refer to the same entity.	“Priya said she is tired.” → “she” = “Priya”

Approaches to Semantic Processing

Approach	Description	Examples / Tools
Rule-Based / Ontology-Based	Uses hand-crafted semantic rules or knowledge bases.	WordNet, ConceptNet, FrameNet

Statistical / Distributional	Learns meaning from word co-occurrences in corpora.	TF-IDF, PMI, Word2Vec
Neural / Contextual	Learns meaning dynamically from large text data.	BERT, GPT, RoBERTa

Modern NLP systems often use **hybrid methods**, combining lexical resources with deep learning.

Example: Meaning through Context

Word	Context Sentence	Meaning
“light”	“The room was filled with light.”	Noun → illumination
“light”	“He carried a light bag.”	Adjective → not heavy
“light”	“Please light the lamp.”	Verb → to ignite

Semantic models must use surrounding context to infer correct interpretation.

Applications of Semantic Processing

- **Search Engines:** Understand intent behind queries.
- **Chatbots / Virtual Assistants:** Interpret natural user commands.
- **Machine Translation:** Preserve meaning across languages.
- **Information Retrieval:** Match queries with conceptually similar documents.
- **Sentiment Analysis:** Understand underlying emotions and attitudes.
- **Question Answering Systems:** Extract correct factual answers from text.

Challenges

Aspect	Challenge
Ambiguity	Words often have multiple meanings depending on context.
Idiomatic Expressions	Literal parsing fails for idioms (“kick the bucket”).
Context Dependency	Meaning depends on discourse and world knowledge.
Polysemy & Synonymy	Words can share or overlap in meaning.
Pragmatic Interpretation	Real understanding needs common sense and intent recognition.

Lexical Semantics and Word Relations

Definition

Lexical Semantics is the study of the meaning of words and the relationships between them.

It focuses on how individual words convey meaning and how those meanings combine to represent more complex ideas in sentences.

In NLP, lexical semantics helps computers interpret words correctly, recognise semantic similarity, and disambiguate meanings based on context.

Core Idea

Words are **not isolated symbols** — their meanings are shaped by relationships to other words.

For example:

- “car”, “vehicle”, and “automobile” are semantically related.
- “run” can mean different things in different contexts: *run a race*, *run a company*, *run a program*.

Lexical semantics thus provides the **foundation for understanding vocabulary, synonymy, polysemy, and sense relations** in natural language.

Goals of Lexical Semantics

1. Define **word meaning** systematically.
2. Identify **semantic relationships** among words.
3. Build computational representations like **lexicons**, **semantic networks**, and **ontologies** (e.g., WordNet).
4. Support downstream NLP tasks like **query expansion**, **text similarity**, and **semantic search**.

Types of Word Meaning

Aspect	Description	Example
Denotative Meaning	The literal, dictionary meaning.	"Rose" → a flower.
Connotative Meaning	Emotional or cultural associations.	"Rose" → love, beauty.
Compositional Meaning	Meaning derived from combining words.	"Red car" → car with the property of being red.

Lexical Relations

Words are semantically related through several **lexical relations**.

Understanding these helps in mapping word meanings and building semantic networks.

Relation Type	Definition	Example
Synonymy	Words with similar meaning.	happy ↔ joyful

Antonymy	Words with opposite meanings.	hot ↔ cold
Hyponymy	Specific concept within a general category.	mango ⊂ fruit
Hypernymy	General category containing specific instances.	fruit ⊃ mango
Meronymy	Part–whole relationship.	wheel ↦ car
Holonymy	Whole–part relationship.	car ↦ wheel
Polysemy	One word with multiple related meanings.	"run" → race / operate / function
Homonymy	Same form, unrelated meanings.	"bat" → animal / cricket bat
Troponymy	A specific manner of an action.	stroll ↦ walk

Examples

Word	Relation	Connected Words
"car"	Hyponym of	vehicle
"vehicle"	Hypernym of	car, bus, truck
"wheel"	Meronym of	car
"drive"	Related Verb	travel, move
"fast"	Antonym of	slow

These relationships form the **semantic backbone** of WordNet and other lexical databases.

WordNet: A Lexical Database

WordNet is a large lexical network of English words organised by semantic relationships.

- **Synsets** (Synonym Sets): Groups of synonymous words expressing a single concept.

Example: {car, automobile, machine, motorcar}

- **Relations:** Synsets connected via relations like *is-a* (hypernymy) or *part-of* (meronymy).
- **Hierarchy:** Organised from general → specific concepts.

Example: *entity* → *object* → *vehicle* → *car* → *sports car*

Applications:

- Word sense disambiguation
- Query expansion in search
- Semantic similarity measures
- Ontology construction

Homonymy vs. Polysemy

Aspect	Homonymy	Polysemy
Meaning Relation	Unrelated meanings	Related meanings
Example	“bat” → animal / sport equipment	“head” → body part / leader
Detection	Requires contextual understanding	Requires sense clustering
Challenge in NLP	Causes ambiguity in retrieval and translation	Affects semantic models and embeddings

Role in NLP Applications

Task	How Lexical Semantics Helps
Information Retrieval	Matches queries with semantically similar terms.
Text Classification	Groups documents by topic using related words.
Chatbots & QA Systems	Resolves word ambiguity to answer correctly.
Machine Translation	Ensures correct sense mapping across languages.
Word Embeddings	Encodes meaning and relations into numeric space.

Challenges in Lexical Semantics

Aspect	Challenge
Ambiguity	Words have multiple possible senses.
Granularity	Some distinctions are too subtle to model.
Cultural Variation	Meaning shifts across domains and languages.
Dynamic Language	Word meanings evolve over time (e.g., “cloud”).

Semantic Ambiguity, Polysemy, and Homonymy

Definition

Semantic Ambiguity occurs when a word, phrase, or sentence has **multiple possible meanings**.

In human communication, we use context to resolve ambiguity easily — but for machines, this is one of the most challenging aspects of natural language understanding.

Ambiguity can occur at the level of individual words (*lexical ambiguity*) or at the level of full sentences (*structural ambiguity*).

Types of Semantic Ambiguity

Type	Description	Example
Lexical Ambiguity	A single word has multiple meanings.	“bank” → river bank / financial bank
Syntactic Ambiguity	Sentence structure allows multiple interpretations.	“I saw the man with the telescope.”
Semantic Role Ambiguity	Unclear who performs or receives an action.	“John promised Peter to go.”
Pragmatic Ambiguity	Depends on speaker intent or situation.	“Can you open the door?” (request, not ability check)

Lexical Ambiguity: Polysemy vs. Homonymy

Lexical ambiguity arises primarily because of **polysemy** and **homonymy**, two key phenomena in lexical semantics.

1. Polysemy

A word has **multiple related meanings**, derived from a shared origin or concept.

Example Word	Different but Related Meanings
“head”	body part / leader / top of something

“paper”	material / newspaper / research article
“run”	to move fast / to operate / to manage

Polysemy often reflects *metaphorical extensions* or *functional shifts* of meaning.

2. Homonymy

A word (or word form) has **two or more entirely unrelated meanings**, though they share the same spelling or pronunciation.

Example Word	Unrelated Meanings
“bat”	flying mammal / cricket bat
“bear”	animal / to carry or tolerate
“left”	opposite of right / departed

Homonyms are distinct words that happen to share the same form, creating ambiguity in interpretation.

Comparison: Polysemy vs. Homonymy

Aspect	Polysemy	Homonymy
Meaning Relation	Related meanings	Unrelated meanings
Etymology	Same origin	Different origins
Example	“mouth” → of river, of person	“bat” → animal, sports equipment
Resolution	Context reveals sense extension	Context identifies correct word altogether

Challenge in NLP	Complex clustering of related meanings	Word-sense identification from form alone

Context as a Disambiguator

Context plays a central role in resolving ambiguity:

Sentence	Word in Question	Correct Meaning
"He deposited money in the bank."	bank	financial institution
"He sat by the bank of the river."	bank	riverside
"The crane flew away."	crane	bird
"The crane lifted the load."	crane	machine

Machines use **statistical context** (surrounding words and co-occurrence patterns) to infer the intended meaning.

Why Ambiguity Matters in NLP

Application	Effect of Ambiguity
Search Engines	Wrong sense may lead to irrelevant results.
Machine Translation	Different meanings translate differently.
Chatbots / QA Systems	May misinterpret user intent.
Information Extraction	Incorrect entity or relation identified.

Hence, **Word Sense Disambiguation (WSD)** is a crucial next step — to assign the correct meaning based on context.

Approaches to Handle Ambiguity

Approach	Method	Example
Knowledge-Based	Use lexical databases like WordNet; compare definitions.	Lesk Algorithm
Statistical	Use co-occurrence frequencies from corpora.	PMI, N-grams
Neural Contextual	Use embeddings to infer meaning dynamically.	BERT, ELMo, GPT

Modern systems often integrate multiple cues — lexical, syntactic, and contextual — to make accurate sense predictions.

Challenges

Aspect	Challenge
Fine-Grained Senses	Dictionaries may list too many subtle distinctions.
Context Dependency	Same word can change meaning within short context spans.
Domain Variance	Words behave differently in technical vs. everyday language.
Idioms & Metaphors	Meanings cannot be derived compositionally ("kick the bucket").

Word Sense Disambiguation (WSD)

Definition

Word Sense Disambiguation (WSD) is the task of determining which sense (meaning) of a word is intended in a given context.

It is a central problem in lexical semantics — because many words are **ambiguous**, and their correct interpretation depends on the context they appear in.

Example: “He sat by the *bank*.”

→ “bank” could mean *river bank* or *financial bank*; WSD selects the correct one based on surrounding words.

Objective

The goal of WSD is to assign the **most appropriate sense** to each ambiguous word in a text, given:

1. The word itself (target word),
2. Its context (neighbouring words), and
3. A predefined **sense inventory** (dictionary or lexical database like WordNet).

Types of WSD Approaches

Approach Type	Description	Examples / Algorithms
Knowledge-Based	Uses lexical databases (WordNet, dictionaries) and semantic similarity.	Lesk Algorithm, Semantic Overlap, Concept Graphs
Supervised Learning	Learns from annotated corpora where word senses are labelled.	Naive Bayes, Decision Trees, Neural Classifiers
Unsupervised Learning	Clusters words by usage patterns without labelled data.	Word Clustering, Context Similarity

Semi-Supervised	Combines a small labelled dataset with large unlabelled corpora.	Bootstrapping, EM Algorithms
Contextual Embeddings	Uses neural models to derive sense dynamically.	BERT, ELMo, GPT-based WSD

Knowledge-Based Methods

One of the earliest and most widely known WSD algorithms.

It relies on **dictionary definitions (glosses)** to find overlap between the definitions of the ambiguous word and its context words.

Algorithm Steps:

1. Retrieve all possible dictionary senses for the target word.
2. Retrieve senses for the surrounding (context) words.
3. Compute overlap between the glosses (definitions).
4. Choose the sense with the **maximum overlap**.

Example:

Sentence: “He sat by the *bank of the river*.”

- “bank_1” → financial institution
- “bank_2” → sloping land beside a river

The gloss for “river” overlaps with “bank_2” → correct sense = river bank.

2. Semantic Similarity Methods

These approaches compute the **semantic distance** between the candidate sense of the target word and the senses of nearby words using resources like WordNet.

Example Measures:

- **Path Similarity:** Based on the shortest path between synsets in WordNet.
- **Wu–Palmer Similarity:** Based on depth of synsets in hierarchy.
- **Resnik / Lin Measures:** Based on shared information content.

Intuition:

The correct sense is the one most semantically similar to the senses of its neighbours.

Supervised Learning Methods

Supervised WSD models treat disambiguation as a **classification problem**.

Example Workflow:

1. **Training Data:** Each occurrence of an ambiguous word is labelled with its correct sense.
2. **Feature Extraction:** Surrounding words, POS tags, collocations, or dependency relations are used as features.
3. **Model Training:** Algorithms like Naive Bayes, SVM, or Neural Networks learn sense patterns.
4. **Prediction:** Model predicts the most likely sense for unseen sentences.

Example:

Sentence: “The *bass* was very loud.” → sound sense

Sentence: “The *bass* swam away.” → fish sense

Model learns contextual cues (“loud”, “swam”) that signal specific senses.

Unsupervised and Neural Approaches

1. Clustering-Based WSD

Groups word usages into clusters based on **contextual similarity**.

- Each cluster represents a potential sense.
- Works without labelled data.
- Often uses vector representations of context (e.g., TF-IDF, embeddings).

2. Contextual Embeddings

Modern models like **BERT** or **GPT** implicitly perform WSD.

They generate **different embeddings** for the same word depending on context.

Example:

- "He sat by the *bank of the river*." → vector$_1$
- "He deposited money in the *bank*." → vector$_2$

→ cosine similarity(vector$_1$, vector$_2$) is low → different meanings.

Thus, contextual models dynamically encode sense information without explicit dictionaries.

Applications of WSD

Application	Role of WSD
Machine Translation	Ensures correct word sense is translated.
Information Retrieval	Matches query terms with correct semantic meaning.
Text Summarisation	Avoids ambiguous condensation of key ideas.
Question Answering	Interprets question and source meaning accurately.
Chatbots / Dialogue Systems	Prevents misinterpretation of user input.

Challenges in WSD

Aspect	Challenge
Sense Inventory Dependence	WordNet sense boundaries may not match human intuition.
Domain Sensitivity	Word meanings differ across contexts (finance, medicine, etc.).
Lack of Labeled Data	Supervised WSD needs sense-annotated corpora.

Dynamic Language Use	New senses emerge (e.g., “cloud”, “stream”).
Fine-Grained Senses	Overlapping or nuanced distinctions are hard to learn.

Word Sense Disambiguation (WSD)

Definition

Semantic Similarity measures how close in meaning two linguistic units are — such as words, phrases, or sentences — based on their semantic content.

Semantic Relatedness is a broader concept that measures any kind of semantic association, whether or not the meanings are similar.

Example:

- *Similarity*: “car” and “automobile” → nearly identical meaning.
- *Relatedness:* “car” and “driver” → conceptually related, but not similar.

Core Idea

Humans intuitively recognise semantic relationships (e.g., “doctor” and “nurse” are related).

In NLP, we quantify this intuition using **computational models** that estimate similarity or relatedness between word meanings.

These models help machines perform tasks such as:

- Grouping similar terms
- Expanding search queries
- Detecting paraphrases
- Evaluating text coherence

Semantic Similarity vs. Semantic Relatedness

Aspect	Semantic Similarity	Semantic Relatedness

Focus	Measures degree of shared meaning	Measures any semantic connection
Relation Type	Synonymy, Hyponymy	Association, Functional Relation
Example	"car" ↔ "automobile"	"car" ↔ "road", "driver"
Use Case	Clustering, Paraphrasing	Knowledge graphs, Topic detection
Model Basis	Ontological distance or embeddings	Co-occurrence or association data

Approaches to Measuring Semantic Similarity

Semantic similarity can be computed through **two main approaches**:

1. **Knowledge-Based (Lexical Resource–Driven)**
2. **Corpus-Based (Statistical / Distributional)**

1. Knowledge-Based Similarity

Uses structured lexical resources like **WordNet**, **ConceptNet**, or **Ontologies**.

a. Path-Based Measures

Measure the **shortest path** between two synsets (word senses) in a lexical hierarchy.

$$Similarity(w_1, w_2) = \frac{1}{path\ length(w_1, w_2) + 1}$$

Example:

- Path(car, vehicle) = 1 → High similarity
- Path(car, building) = 6 → Low similarity

2. Corpus-Based (Distributional) Similarity

Based on the **Distributional Hypothesis**:

Words that occur in similar contexts tend to have similar meanings.

a. Co-occurrence Models

Represent words as **context vectors** based on neighbouring words in large corpora.

$$Sim(w_1, w_2) = \cos i\, ne(\overrightarrow{w_1}, \overrightarrow{w_2})$$

Example:

If “car” and “bus” frequently occur with “road”, “driver”, “engine”, they have high cosine similarity.

b. Pointwise Mutual Information (PMI)

Measures the degree of association between two words.

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

High PMI → strong relationship (e.g., “coffee” & “cup”),
Low PMI → weak or random co-occurrence.

c. Word Embeddings

Modern neural methods like **Word2Vec**, **GloVe**, and **BERT** learn dense vector representations of words capturing both similarity and relatedness.

Example:

- cosine(“king”, “queen”) ≈ cosine(“man”, “woman”)

→ relationships captured geometrically in vector space.

Applications

Domain	Use Case

Information Retrieval	Expanding search terms with synonyms or related terms.
Text Clustering	Grouping documents by semantic topic.
Machine Translation	Selecting contextually similar words across languages.
Paraphrase Detection	Identifying sentences with the same meaning.
Question Answering	Matching user query to knowledge base phrases.
Recommendation Systems	Matching semantically similar items.

Challenges

Aspect	Challenge
Word Sense Variability	Similarity depends on correct sense interpretation.
Context Sensitivity	Words may have different similarity in different contexts.
Data Sparsity	Co-occurrence models fail for rare terms.
Cross-Lingual Mapping	Measuring similarity across languages is complex.
Interpretability	Neural similarity scores lack human-readable explanation.

upGrad

Concept and Semantic Networks

Definition

A **Semantic Network** (also called a **Concept Network**) is a **graph-based representation of knowledge**, where **nodes** represent *concepts* or *entities*, and **edges** represent *semantic relationships* between them.

In NLP, semantic networks model how words, phrases, and concepts are interconnected in meaning — enabling machines to reason about relationships and context.

Core Idea

Language and thought can be represented as **interconnected concepts**.

Instead of treating words independently, semantic networks connect them based on meaning, allowing systems to infer new relationships and perform reasoning.

Example:

A fragment of a semantic network might look like this:

(car) —— is-a ——► (vehicle)

(car) —— has-part ——► (wheel)

(driver) —— drives ——► (car)

Here, the network encodes facts and relations in a structured, interpretable way.

Structure of a Semantic Network

Element	Description	Example
Node	Represents a concept, word, or entity.	"cat", "mammal", "animal"
Edge / Link	Represents a semantic relationship.	"is-a", "part-of", "has-property"

Label	Defines the relation type.	“is-a” → class-subclass
Weight (optional)	Quantifies the strength of relationship.	0.9 similarity between “car” and “automobile”

Types of Relations in Semantic Networks

Relation Type	Description	Example
Is-a (Hypernymy)	Hierarchical: general to specific.	“Dog” is-a “Animal”
Part-of (Meronymy)	Whole–part relationship.	“Wheel” part-of “Car”
Instance-of	Links a specific entity to a class.	“Fido” instance-of “Dog”
Has-property	Attribute relationship.	“Rose” has-property “Red”
Cause–Effect	Causal linkage.	“Rain” causes “Flood”
Action–Agent	Who performs what.	“Pilot” performs “fly”
Synonym / Antonym	Equivalence or opposition in meaning.	“Happy” ↔ “Joyful”; “Hot” ↔ “Cold”

Example Network Representation

Let’s consider the sentence:

“A dog is an animal that has four legs and a tail.”

This can be represented as:

(dog) —— is-a ——▶ (animal)

(dog) —— has-part ——▶ (leg)

(leg) —— has-property ——▶ (four)

(dog) —— has-part ——▶ (tail)

Each relation captures structured semantic knowledge that can be queried and reasoned about computationally.

1. WordNet: Lexical Semantic Network

WordNet is one of the most widely used **semantic networks** in NLP.

It groups English words into **synsets** (sets of cognitive synonyms) and connects them via **semantic and lexical relations**.

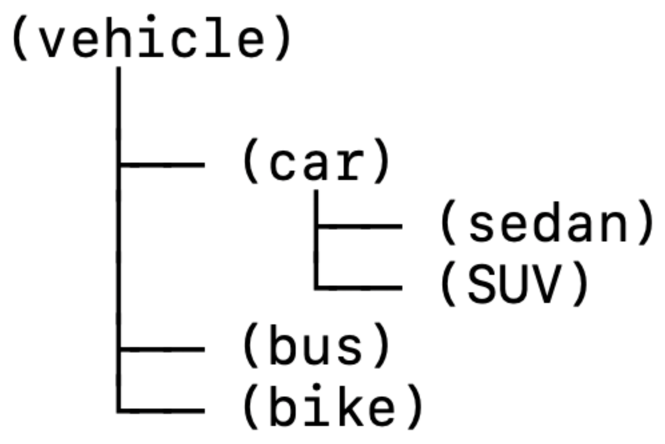
Structure

- **Synset (Concept Node):** Represents a single concept.

Example: {car, automobile, motorcar}

- **Relations:**
    - Hypernymy / Hyponymy ("is-a")
    - Meronymy / Holonymy ("part-of")
    - Troponymy ("manner-of" for verbs)
    - Antonymy (opposite sense)

Example (Simplified View):



WordNet thus forms a **hierarchical semantic graph**, used extensively in word sense disambiguation and similarity computation

2. ConceptNet: Common-Sense Semantic Network

ConceptNet extends semantic networks beyond lexical relations to include **commonsense knowledge** and **everyday reasoning**.

Structure

- Concepts: "car", "road", "drive", "fuel"
- Relations: "UsedFor", "PartOf", "CapableOf", "LocatedAt", etc.
- Edges have **weights**, representing confidence scores learned from large-scale data.

Example (ConceptNet Relations)

Concept 1	Relation	Concept 2	Example Interpretation
"Car"	UsedFor	"Transportation"	Cars are used for transportation
"Driver"	CapableOf	"Drive Car"	Drivers can drive cars
"Car"	HasA	"Engine"	A car has an engine

“Fuel”	UsedFor	“Car”	Fuel is used for a car
“Car”	AtLocation	“Road”	Cars are located on roads

These relations enable **contextual understanding** and **inference** for dialogue systems, reasoning tasks, and semantic search.

Example: Visualising a Small Concept Network

```
  [Driver]
     |
 CapableOf
     |
  [Drive] ⟶ UsedFor ⟶ [Car]
                         |
                       HasA
                         |
                     [Engine]
                         |
                      Uses
                         |
                      [Fuel]
```

This simple network already encodes enough relationships for a system to infer:

- “A driver uses fuel indirectly when driving.”
- “A car has an engine which uses fuel.”

Applications of Semantic Networks

Application	Purpose
Information Retrieval	Retrieve documents based on concept similarity.
Question Answering	Link question entities with relevant knowledge.
Dialogue Systems	Enable contextual and commonsense understanding.

Text Summarisation	Identify central semantic concepts in a document.
Semantic Search	Expand user queries to include related terms.
Recommendation Engines	Suggest conceptually related items.

Advantages

- Enables **explicit reasoning** and **concept-level inference**.
- Represents **structured knowledge** suitable for integration with knowledge graphs.
- Facilitates **semantic similarity**, **disambiguation**, and **concept expansion**.

Limitations

Aspect	Challenge
Coverage	Limited to predefined concepts; new entities may be missing.
Ambiguity	Relations may not always reflect true contextual meaning.
Scalability	Large networks become computationally heavy.
Dynamic Knowledge	Hard to capture evolving language and facts.

Distributional Semantics and Contextual Meaning

Definition

Distributional Semantics is the study of how word meanings can be inferred from their patterns of use in large text corpora.

It is based on the **Distributional Hypothesis**, which states:

“Words that occur in similar contexts tend to have similar meanings.”

— J. R. Firth (1957)

In other words, the meaning of a word can be approximated by the **company it keeps** — i.e., by the words that frequently appear around it.

Core Idea

Rather than relying on dictionaries or human-created hierarchies (like WordNet), distributional semantics **learns meaning from data**.

- If "cat", "dog", and "rabbit" often appear near "pet", "animal", "food", they likely share similar semantics.
- This principle underlies most **modern NLP models**, including word embeddings and contextual representations.

The Distributional Hypothesis

Formally:

If two words w_i and w_j have **similar distributions** across contexts, then they have **similar meanings**.

Example

Word	Common Context Words
"dog"	bark, tail, pet, walk
"cat"	meow, pet, fur, sleep
"car"	drive, road, wheel, fuel

→ "dog" and "cat" have overlapping context sets → semantically similar.
→ "car" shares few contexts → semantically different.

Representing Meaning in Vector Space

Each word can be represented as a **vector** capturing its co-occurrence with other words.

This leads to the concept of a **Vector Space Model (VSM)**, where distance or angle between vectors reflects semantic closeness.

Example (Simplified Co-occurrence Matrix)

	pet	bark	fur	road
dog	5	4	1	0
cat	6	0	3	0
car	0	0	0	7

→ Cosine similarity between “dog” and “cat” is high;
→ between “dog” and “car” is low.

Measuring Semantic Similarity

The **cosine similarity** between two word vectors $\overrightarrow{w_1}$ a nd $\overrightarrow{w_2}$ is defined as:

$$sim(w_1, w_2) = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||\overrightarrow{w_1}|| \times ||\overrightarrow{w_2}||}$$

- Value ranges from **-1 (opposite)** to **1 (identical)**.
- High similarity → similar context and meaning.

Pointwise Mutual Information (PMI)

PMI measures how strongly two words are associated compared to what would be expected if they were independent.

$$PMI(w_i, w_j) = \log_2 \frac{P(w_i, w_j)}{P(w_i) \times P(w_j)}$$

Where:

- $P(w_i, w_j)$ = probability of words co-occurring
  - $P(w_i), P(w_j)$ = individual probabilities

Interpretation

- **High PMI** → words co-occur more often than chance (e.g., *coffee – cup*).
- **Low PMI** → rarely appear together or unrelated (e.g., *coffee – car*).
- Often, **Positive PMI (PPMI)** is used to ignore negative values.

Example Calculation

Suppose in a corpus of 10,000 sentences:

- “coffee” appears in 200
- “cup” appears in 250
- They co-occur in 150 sentences.

$$P(coffee) = \frac{200}{10000}, \quad P(cup) = \frac{250}{10000}, \quad P(coffee, cup) = \frac{150}{10000}$$

$$PMI = \log_2 \frac{0.015}{(0.02 \times 0.025)} = \log_2(30) \approx 4.9$$

→ strong positive association between “coffee” and “cup”.

Context Window and Representation

The **context window** defines how many words around the target word are considered for co-occurrence.

Window Size	Interpretation	Use Case
Small (±1–2)	Captures syntactic relations	POS, local dependencies
Large (±5–10)	Captures semantic topics	Topic modelling, embeddings

Example:

Sentence: “The dog chased the cat.”

- For “dog”, context (±2): [The, chased, the, cat]
- For “cat”, context (±2): [chased, the, .]

From Co-occurrence to Meaning

1. Build a **co-occurrence matrix** of words vs. context words.
2. Apply **statistical weighting** (TF-IDF, PMI).
3. Use **dimensionality reduction** (SVD, PCA) to capture latent semantics.
4. Represent each word as a **dense vector** → Semantic space.

This process forms the basis of **Latent Semantic Analysis (LSA)** and later **Word2Vec**-style models.

Applications of Distributional Semantics

Application	Purpose
Word Similarity & Clustering	Group semantically close words.
Information Retrieval	Match conceptually similar terms in queries.
Machine Translation	Align words with similar usage across languages.
Question Answering	Identify semantically similar phrases.
Semantic Search & Recommendation	Retrieve related content using meaning rather than keywords.

Challenges

Aspect	Challenge
Ambiguity	Word meanings depend on context (polysemy).
Sparsity	Rare words have few co-occurrences.

High Dimensionality	Co-occurrence matrices can be extremely large.
Context Independence	Traditional models assign one vector per word (no sense separation).
Interpretability	Hard to explain why two words are similar numerically.

Evolution Towards Contextual Semantics

Traditional distributional models treat meaning as **static**.

Modern neural models like **BERT** and **GPT** extend this by making representations **context-sensitive**, where each word gets a unique vector based on its surrounding context.

Example:

- "He sat by the *bank* of the river." → embedding_1
- "He went to the *bank* to deposit money." → embedding_2

Same word, **different vectors**, different meanings.

Semantic Role Labelling (SRL): Rule-Based Methods

Definition

Semantic Role Labelling (SRL) identifies the *who did what to whom* structure of a sentence.

It determines the main **predicate (verb)** and the **roles** played by each entity involved in the action.

Example:

"Riya gave a book to Meera."

- Predicate: *gave*
- Agent (doer): *Riya*
- Theme (object): *book*
- Recipient (receiver): *Meera*

SRL helps uncover the **meaningful relationships** between verbs and their participants, forming the basis of semantic understanding in NLP.

Core Idea

Rule-based SRL systems depend on **grammatical and syntactic rules** derived from linguistic analysis.

They map **syntactic dependencies** (like subject, object, and prepositional phrases) to **semantic roles** using fixed, hand-crafted mappings.

These systems generally start from a **dependency parse** of the sentence, then assign roles according to defined rules.

Rule Mapping Approach

Dependency Relation	Semantic Role	Example
nsubj(verb, noun)	Agent (doer)	*Riya* → Agent of *gave*
dobj(verb, noun)	Theme / Patient	*book* → Theme of *gave*
nmod:to(verb, noun)	Recipient	*Meera* → Recipient of *gave*
nmod:by(verb, noun)	Agent (in passive voice)	*by Riya* → Agent in *was baked by Riya*

Example sentence:

"Riya gave a book to Meera."

→ Predicate: *gave*

→ Roles: Agent = Riya, Theme = book, Recipient = Meera

Handling Active and Passive Voice

Rule-based SRL systems explicitly handle voice transformations through syntactic cues.

Sentence	Role Assignment

Riya baked a cake.	Agent = Riya, Theme = cake
A cake was baked by Riya.	Theme = cake, Agent = Riya (from *by*-phrase)

By defining both *nsubj* and *nsubjpass* relations, the system correctly maps roles across voices.

Example Output Structure

For the sentence: *"Riya gave a book to Meera."*

Predicate: gave

Agent: Riya

Theme: book

Recipient: Meera

The final structure captures the **predicate–argument frame**, forming a machine-readable representation of meaning.

Advantages

- **High precision** for grammatical constructions that match defined rules.
- **Transparent and interpretable**, as role assignments can be traced directly to rules.
- Works well for **clear, well-formed sentences** in restricted domains.

Limitations

- **Low flexibility:** fails on unseen or syntactically complex sentences.
- **Manual rule creation:** requires linguistic expertise and extensive tuning.
- **Poor domain generalisation:** rules must be adapted for new text types or languages.

Named Entity Recognition (NER)

Definition

Named Entity Recognition (NER) is the process of identifying and classifying **specific names** in text that refer to real-world entities such as *people*, *organizations*, *locations*, *dates*, *quantities*, and *events*.

Example:

“Riya works at Google in Bengaluru.”

- *Riya* → PERSON
- *Google* → ORGANIZATION
- *Bengaluru* → LOCATION

NER helps convert **unstructured text** into **structured information** by tagging key entities that carry factual meaning.

Objective

The goal of NER is twofold:

1. **Entity Detection:** Identify which words or phrases in the text refer to entities.
2. **Entity Classification:** Assign each identified entity to its appropriate category (e.g., PERSON, LOCATION).

By doing so, NER allows systems to recognise meaningful data within natural language input.

Core Idea

NER works by analysing both **surface features** (like capitalization or suffixes) and **contextual patterns** (like words occurring before or after an entity).

Early systems were built using **linguistic rules and dictionaries**, while later ones applied **statistical models** that could automatically learn such patterns.

In both approaches, the core idea remains the same — identifying **named entities** based on patterns and their usage in context.

Common Entity Categories

Entity Type	Description	Example
PERSON	Names of individuals	*Riya*, *Elon Musk*
ORGANIZATION	Companies, institutions, or groups	*Google*, *UNICEF*
LOCATION / GPE	Cities, countries, regions	*Bengaluru*, *India*
DATE / TIME	Dates or time expressions	*July 2025*, *3 PM*
MONEY / QUANTITY	Numeric or currency expressions	*₹500*, *50 kilograms*
EVENT / PRODUCT	Named events or products	*Olympics 2024*, *iPhone 15*

Rule-Based and Pattern-Based NER

Rule-based systems use **lists of known entities**, called **gazetteers**, which contain names of people, cities, companies, etc.

If a word in the text matches an entry in the gazetteer, it is tagged as that entity type.

Example:

- *Paris* → LOCATION
- *Google* → ORGANIZATION

Gazetteers are often customized for specific domains (medical, legal, financial).

Pattern and Heuristic Rules

NER also uses **pattern-based heuristics** based on linguistic structure and surrounding words.

Examples of Rules:

- If a word **starts with a capital letter** and follows “Mr.”, “Ms.” → likely PERSON
- If a word **ends with “Ltd.”, “Inc.”, or “Corp.”** → likely ORGANIZATION
- If a phrase follows **“in” or “at”** → likely LOCATION
- If text matches **date formats** (e.g., 12/10/2025, “March 5”) → DATE
- If word contains **currency symbols (₹, $, €)** → MONEY

These rules capture regularities found in natural text without needing machine learning.

Example

Sentence: “Riya works at Google in Bengaluru.”

Rule Applications:

- Capitalized first word, not at sentence start → *Riya* → PERSON
- Word ending pattern “Google” (in organization gazetteer) → ORGANIZATION
- Word following “in” → *Bengaluru* → LOCATION

NER Output:

Riya → PERSON

Google → ORGANIZATION

Bengaluru → LOCATION

Statistical Overview

Later, statistical approaches automated this process using **feature-based models** such as Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs).

These models learn from annotated examples and use **features** like:

- Capitalization
- Prefixes and suffixes
- Part-of-speech tags
- Surrounding context words (e.g., “in”, “at”, “from”)

However, in concept, they aim to detect the *same kinds of regularities* as rule-based systems — just learned automatically instead of manually coded.

Advantages

Aspect	Benefit
High Precision	Accurate for structured and domain-specific data.
Interpretability	Easy to understand how rules identify entities.
No Training Data Required	Works without labelled datasets.

Limitations

Aspect	Challenge
Low Recall	Misses entities not covered by rules or dictionaries.
Manual Maintenance	Requires frequent updating of gazetteers.
Ambiguity	Same word can belong to different entity types (*Apple* → fruit or company).
Language Variability	Rules must be rewritten for each language or domain.

Applications

Domain	Example Use Case
Information Extraction	Identifying people, places, and organizations from documents.

Search and Retrieval	Improving search accuracy through entity tags.
Question Answering	Detecting the target entity type from user queries.
Knowledge Graph Building	Linking recognized entities to structured knowledge bases.

IOB Labelling

Definition

IOB Labelling (Inside–Outside–Beginning) is a **tagging format** used to represent the boundaries of entities or phrases in a sequence of words.

It is the standard annotation scheme for tasks like **Named Entity Recognition (NER)**, **Chunking**, and **Semantic Role Labelling (SRL)**.

The IOB scheme tells us **where an entity starts**, **continues**, and **ends** within a sentence.

Core Idea

When recognising entities or chunks in text, we need not only the **type of entity**, but also its **extent** — where it begins and ends.

IOB labelling encodes this boundary information token by token.

It uses three primary tags:

- **B** – *Beginning* of an entity
- **I** – *Inside* (continuation) of the same entity
- **O** – *Outside* (not part of any entity)

The IOB Tagging Scheme

Tag	Meaning	Example
B-XXX	Beginning of an entity of type *XXX*	*B-PER* (Beginning of a Person name)
I-XXX	Continuation of the same entity	*I-PER* (Next word in the same Person name)
O	Not part of any entity	–

Example: Applying IOB to NER

Sentence:

“Riya works at Google in Bengaluru.”

Word	IOB Tag
Riya	B-PER
works	O
at	O
Google	B-ORG
in	O
Bengaluru	B-LOC
.	O

Explanation

- *Riya* is a single-word **Person entity**, so it gets **B-PER**.

- *Google* begins a new **Organization entity**, marked **B-ORG**.
- *Bengaluru* begins a **Location entity**, marked **B-LOC**.
- All other words are **O** — they are not part of any entity.

Example: Multi-Word Entities

Sentence:

“Elon Musk founded Space Exploration Technologies.”

Word	IOB Tag
Elon	B-PER
Musk	I-PER
founded	O
Space	B-ORG
Exploration	I-ORG
Technologies	I-ORG
.	O

Explanation

- “Elon Musk” → two-word *Person entity*: B-PER, I-PER.
- “Space Exploration Technologies” → multi-word *Organization entity*: B-ORG, I-ORG, I-ORG.

Variants of IOB Tagging

There are a few related formats that appear in different corpora:

Scheme	Tags Used	Description
IOB1 / IOB2	B, I, O	Standard format – every entity starts with **B-**.
BIOES	B, I, O, E, S	Adds **E** (End) and **S** (Single) for finer boundaries.
BILOU	B, I, L, O, U	Similar idea — L = Last, U = Unit.

Most modern corpora (e.g., **CoNLL NER datasets**) follow the **IOB2 scheme**.

Why IOB Labelling is Important

- Provides **clear and consistent boundaries** for entities or chunks.
- Converts complex text segmentation into a **token-level classification problem**.
- Makes it possible to train models like **HMMs, CRFs, and sequence taggers** for tasks such as NER or SRL.
- Enables easy evaluation using **Precision, Recall, and F1-score** at the entity level.

Example: IOB in Semantic Role Labelling

Sentence:

"Riya gave a book to Meera."

Word	Role	IOB Tag
Riya	Agent	B-ARG0
gave	Predicate	O

a	–	O
book	Theme	B-ARG1
to	–	O
Meera	Recipient	B-ARG2

Here, ARG0, ARG1, and ARG2 correspond to different semantic arguments:

- ARG0 → *Agent*
- ARG1 → *Theme*
- ARG2 → *Recipient*

Applications

Task	Use of IOB Labels
Named Entity Recognition	Marking boundaries of named entities.
Chunking / Phrase Detection	Identifying noun and verb phrase boundaries.
Semantic Role Labelling	Marking arguments of a predicate.
Information Extraction	Structuring data from raw text into labeled entities.

Advantages

Aspect	Benefit
Simplicity	Easy to implement and interpret.

Consistency	Standardised format across multiple NLP tasks.
Compatibility	Works well with sequence models like HMMs and CRFs.
Evaluation-Friendly	Enables token- and entity-level performance tracking.

Limitations

Aspect	Challenge
Complex Entities	Nested or overlapping entities are difficult to represent.
Boundary Errors	Mislabelled beginnings or endings affect full entity accuracy.
Limited Hierarchy	Does not capture sub-entity structure or fine-grained types.

Conditional Random Fields (CRF)

Definition

Conditional Random Fields (CRF) are **probabilistic models** used for **sequence labelling tasks**, where the goal is to assign labels (like IOB tags) to each element in a sequence (such as words in a sentence).

Unlike simpler models that label each word independently, CRFs consider the **entire context** of the sentence — ensuring that the predicted sequence of labels is **globally consistent.**

Example:

In Named Entity Recognition, CRFs help ensure that tags like

B-PER → I-PER → O

appear in valid order, avoiding illogical tag sequences like

I-PER → O → B-PER.

Why CRFs are Needed

Earlier models such as **Naive Bayes** or **Maximum Entropy** predicted each label **independently**, ignoring the dependencies between neighbouring tags.

However, in sequence tasks like NER or POS tagging, **adjacent words and their labels are correlated**.

Example:

Sentence: "Riya works at Google."

If "Riya" is tagged as B-PER, the next token "works" is unlikely to be I-PER.

A CRF captures this dependency between labels.

Core Idea

CRFs model the **conditional probability** of a label sequence Y given an observation sequence X:

$$P(Y/X)$$

- X = sequence of words or features
- Y = corresponding sequence of labels

The model defines how **contextual features** (like capitalization, POS tags, neighbouring words) influence the probability of label assignments, **while enforcing dependencies** between neighbouring labels.

Comparison with HMM

Aspect	Hidden Markov Model (HMM)	Conditional Random Field (CRF)

Type	Generative model	Discriminative model
Goal	Model joint probability $P(X, Y)$	Model conditional probability $P(X/Y)$
Feature Use	Limited to observed tokens	Can use arbitrary overlapping features
Context Handling	Considers transitions between hidden states	Considers both features and transitions
Flexibility	Strong independence assumptions	More flexible, fewer assumptions

Key advantage:

CRFs directly model the conditional distribution P(Y|X), allowing the inclusion of many informative and overlapping features.

Intuitive Example

Suppose we are tagging entities in the sentence:

"Riya works at Google."

We want the model to output:

Riya → B-PER

works → O

at → O

Google → B-ORG

CRFs learn that:

- A word starting with a capital letter + verb following → more likely PERSON.
- The pattern "at [Capitalized Word]" → likely ORGANIZATION.
- The tag *B-ORG* rarely follows *I-PER* directly → structural dependency learned automatically.

Thus, the CRF balances **word-level evidence** and **label sequence consistency**.

Mathematical Form (Simplified)

CRFs define a conditional probability:

$$P(Y/X) = \frac{1}{Z(X)} \exp\left(\sum_t \sum_k \lambda_k f_k(y_t, y_{t-1}, X, t)\right)$$

Where:

- f_k : feature functions (e.g., capitalisation, previous tag)
- λ_k : learned feature weights
- $Z(X)$: : normalization constant ensuring probabilities sum to 1
- y_t : current label
- y_{t-1} : previous label

The exponential term combines evidence from **features and transitions**, producing the most probable label sequence.

Example of Features Used in NER

Feature Type	Example	Purpose
Word Form	Word = "Riya"	Identifies capitalized names
Prefix / Suffix	"Ltd.", "Corp."	Detects organizations
Part of Speech (POS)	Proper Noun (NNP)	Distinguishes entities
Context Window	Previous / next words	Captures local context
Previous Tag	B-PER → I-PER	Enforces tag continuity

Each feature contributes to the final tagging decision, weighted according to its learned importance.

Advantages of CRFs

Aspect	Benefit

Context-Aware	Uses neighbouring words and tags for coherent predictions.
Feature-Rich	Can include multiple linguistic or orthographic features.
Global Optimization	Finds best label sequence for the whole sentence.
Widely Used	Standard in NER, POS tagging, chunking, and SRL.

Limitations

Aspect	Challenge
Training Complexity	Computationally intensive for large datasets.
Feature Engineering	Requires manual selection and crafting of features.
Limited Long Context	Handles short dependencies; not suitable for long-distance relations.
Data Requirement	Needs annotated corpora for supervised learning.

Applications

Task	How CRFs Are Used
Named Entity Recognition	Tagging entities with IOB labels.
Part-of-Speech Tagging	Assigning grammatical labels to tokens.
Chunking	Identifying noun and verb phrase boundaries.

Semantic Role Labelling	Recognising argument roles around predicates.

Coreference Resolution

Definition

Coreference Resolution is the task of identifying when **different words or phrases refer to the same entity** in a text.

It allows a system to connect pronouns or repeated mentions back to the correct noun phrase, enabling deeper understanding of meaning and continuity in discourse.

Example:

"Riya bought a book. She loved it."

- *Riya* → *She* (same person)
- *book* → *it* (same object)

By resolving such references, a machine can interpret who or what each mention refers to — a crucial step in understanding narratives and extracting information.

Core Idea

Human language uses **anaphora** (later mentions referring back to earlier ones) to avoid repetition.

Coreference resolution bridges these mentions, linking:

- **Pronouns** → their antecedents (*she*, *he*, *it*, *they*).
- **Noun phrases** → earlier mentions (*the company* → *Google*).
- **Definite expressions** → specific entities (*the teacher* → *Mrs. Mehta*).

Example

Sentence 1: "Priya went to the store because she wanted milk."

– she → Priya

Sentence 2: “Google announced a new product. The company plans to launch it next month.”

– The company → Google

– it → product

Such linking helps models maintain coherence across sentences.

Types of References

Type	Description	Example
Pronominal Reference	Pronouns referring to earlier entities.	*Riya said she will come.*
Nominal Reference	Noun phrase referring back to another noun phrase.	*The teacher entered. The lady was smiling.*
Demonstrative Reference	Uses “this,” “that,” “these,” “those.”	*I liked the film. That was amazing.*
Cataphora	Reference points **forward** instead of backward.	*When she arrived, Riya was tired.*

Rule-Based Coreference Resolution

Early NLP systems used **rule-based approaches**, which rely on **syntactic and semantic heuristics** to decide whether two mentions refer to the same entity.

Common Rules and Heuristics

1. **Gender Agreement:**

Pronoun and noun must match in gender.

Riya → *she*, not *he*.

2. **Number Agreement:**

Singular pronouns map to singular nouns.
Students → *they* ✓; *Students* → *it* ✕

3. **Person Agreement:**

I ↔ first person, *he/she* ↔ third person.

4. **Syntactic Proximity:**

The antecedent usually appears **closest** before the pronoun.

5. **Semantic Compatibility:**

The antecedent and pronoun should be **logically compatible**.
The car stopped because it was broken ✓; *The car stopped because he was broken* ✕

Example Application

Text: "Riya met Priya at the café. She ordered coffee."

Possible antecedents for *She*: *Riya* or *Priya*.

Rule-based systems check:

- Both are singular + female → Gender match ✓
- Nearest antecedent = *Priya* → chosen antecedent

Output: "She → Priya"

Statistical and Feature-Based Approaches

Later systems learned from annotated corpora to decide whether a pair of mentions are **coreferent** using features such as:

- String match (exact or partial)
- Head noun agreement
- Semantic type (Person, Location, Organization)
- Context distance (number of sentences apart)
- Grammatical role (subject, object)

Each pair is evaluated, and clusters of referring expressions are formed.

Example Feature: If both mentions share the same head noun + agree in number → likely coreferent.

Output Representation

Coreference resolution produces **clusters of mentions** that refer to the same entity.

Example Text:

“Riya bought a book. She read it immediately.”

Cluster ID	Mentions	Entity
1	Riya, She	Person
2	book, it	Object

Applications

Domain	Use Case
Information Extraction	Consolidate facts about the same entity.
Question Answering	Maintain context across sentences.
Summarisation	Merge redundant references.
Dialogue Systems	Track who or what is being talked about.
Machine Translation	Preserve gender and number agreements across languages.

Challenges

Aspect	Example Issue
Ambiguity	“Riya told Priya that she was late.” → Who is *she*?
Long-Distance Dependencies	“Riya loves books. They are her escape from reality.”
Common-Sense Reasoning	“The cup didn’t fit in the bag because it was too small.” → Which was small?
Plural and Collective References	“The team won. They celebrated.”

Advantages

Aspect	Benefit
Improves Coherence	Enables contextual understanding of text.
Supports Higher NLP Tasks	Essential for QA, summarisation, and dialogue.
Interpretable	Rule-based systems allow transparent explanation of links.

Limitations

Aspect	Challenge

Ambiguity Resolution	Often needs world knowledge to decide.
Limited Scalability	Hand-crafted rules fail on long texts or open domains.
Pronoun Complexity	Gender-neutral or implicit pronouns hard to link.

1) Bag-of-Words (BoW)

Overview

Bag-of-Words converts text into numbers by counting how often each word appears, ignoring grammar and word order. It's simple, fast, and often strong enough for keyword-driven tasks (spam filters, quick classifiers).

Key Ideas

- Build a vocabulary from the corpus (unique tokens after preprocessing).
- Create a Document–Term Matrix (DTM): rows = documents, columns = terms, cells = counts.
- Common preprocessing: lowercase, tokenize, remove stopwords (e.g., "the", "and"), optionally stem/lemmatize.

Worked Numerical Example

Corpus (3 headlines)

1. "The team scored an early goal."

2. "Fans cheered as the goal was scored."

3. "The early goal helped the team win."

After stopword removal and lowercasing:

D1: team, scored, early, goal

D2: fans, cheered, goal, scored

D3: early, goal, helped, team, win

Vocabulary (sorted): [cheered, early, fans, goal, helped, scored, team, win]

DTM (counts):

Doc \ Term	cheered	early	fans	goal	helped	scored	team	win
D1	0	1	0	1	0	1	1	0
D2	1	0	1	1	0	1	0	0
D3	0	1	0	1	1	0	1	1

Reading: cell (D1, scored) = 1 means the word 'scored' appears once in Document 1.

When BoW Works Well

- Keyword-rich tasks: spam detection, domain tagging, quick topic hints.
- As a base representation before weighting (e.g., TF-IDF).

Pitfalls

- No word order or context (“dog bites man” = “man bites dog”).
- High-dimensional and sparse for large vocabularies.
- Synonyms are unrelated; polysemy is ignored.

2) TF-IDF

Overview

TF-IDF reweights BoW counts to highlight terms that are frequent in a document but rare across the corpus—great for search and classification.

Key Ideas

- TF(t,d) = f(t,d) / sum_{t'} f(t',d).
- IDF(t) = ln(N / df(t)), where N = #documents and df(t) = #documents containing t.
- TF-IDF(t,d) = TF(t,d) × IDF(t). The log base only rescales values; rankings remain the same.

Worked Numerical Example

Use the same 3 headlines after stopword removal:

D1: team, scored, early, goal (4 tokens)

D2: fans, cheered, goal, scored (4 tokens)

D3: early, goal, helped, team, win (5 tokens)

Task: Compute TF-IDF for term “scored” in D1 using natural log.

1. TF(scored, D1) = 1/4 = 0.25.
2. df(scored) = 2 (D1 and D2), N = 3.
3. IDF = ln(3/2) ≈ 0.405465108.
4. TF-IDF = 0.25 × 0.405465108 ≈ 0.101366277.

Edge case: if a term appears in all documents, df = N ⇒ IDF = 0 and TF-IDF weight is zero (uninformative).

When TF-IDF Shines

- Search ranking, information retrieval, text classification, clustering.
- Reduces the dominance of ubiquitous terms.

Pitfalls

- Still ignores order/context; synonyms treated separately.
- Corpus-dependent: adding new documents changes IDF (recompute needed).

3) Word Embeddings

Overview

Embeddings map words to dense vectors so geometry reflects meaning: similar words are near each other. Unlike sparse BoW/TF-IDF, embeddings capture distributional similarity (“you shall know a word by the company it keeps”).

Key Ideas

- Learned from context: words co-occurring in windows/sentences → similar vectors.
- Common methods: Word2Vec (skip-gram/CBOW), GloVe (from global co-occurrence/PPMI).
- Cosine similarity is commonly used to measure closeness.

Tiny Numerical Example (PPMI-style intuition)

Toy co-occurrence counts (word × context):

word \ context	pet	bark	meow	fruit	total
cat	30	1	20	0	51
dog	28	18	0	0	46
apple	0	0	0	40	40
total	58	19	20	40	137

Compute probabilities and PPMI for a couple of entries:

5. P(cat, pet) = 30/137; P(cat) = 51/137; P(pet) = 58/137 → PMI(cat,pet) = log[(30/137)/((51/137)(58/137))] ≈ log(1.39) ≈ +0.33 → PPMI = +0.33.
6. P(dog, bark) = 18/137; P(dog) = 46/137; P(bark) = 19/137 → PMI ≈ log[(18/137)/((46/137)(19/137))] ≈ log(2.82) ≈ +1.04 → PPMI ≈ +1.04.
7. P(apple, pet) = 0 → PPMI = 0 (by convention).

Treat each word as a 4D vector [PPMI with pet, bark, meow, fruit]. Cosine similarity makes cat close to dog (both tied to pet), and apple far from both (tied to fruit).

Pitfalls

- Encodes corpus biases; requires sufficient data.
- Out-of-vocabulary (OOV) handling needed for unseen words.

4) Topic Modeling with LDA

Overview

Latent Dirichlet Allocation (LDA) is a probabilistic model where each document is a mixture of topics, and each topic is a distribution over words. We infer hidden topics and per-document topic proportions from the observed words.

Generative Story (plain)

- For each document d: draw topic proportions θ_d ~ Dir(α).
- For each topic k: draw word distribution ϕ_k ~ Dir(β).
- For each token position n in doc d: pick a topic z_dn ~ Cat(θ_d); then a word w_dn ~ Cat(ϕ_{z_dn}).

Worked Numerical Example (one collapsed Gibbs update)

Setup: 2 topics, vocabulary = {team, election}. Symmetric priors $\alpha = 0.5$, $\beta = 0.1$. Update the topic for a token 'team' in document d.

Current counts (excluding this token):

	Sports	Politics
Doc d topic counts n_{d,k}	3	1
Word counts for 'team'	20	2
Word counts for 'election'	5	25
Topic totals N_k	25	27

Collapsed Gibbs update (proportional): $p(z=k|\cdot) \propto (n_{d,k}+\alpha) \times (n_{k,v}+\beta)/(N_k + V\beta)$, V=2, v='team'.

8. Sports: $(3+0.5) \times (20+0.1)/(25 + 2\times0.1) = 3.5 \times 20.1/25.2 \approx 2.7917$.
9. Politics: $(1+0.5) \times (2+0.1)/(27 + 2\times0.1) = 1.5 \times 2.1/27.2 \approx 0.1158$.
10. Normalize: Sum $\approx 2.9075 \rightarrow$ P(Sports) $\approx 2.7917/2.9075 \approx 0.9602$; P(Politics) ≈ 0.0398.

Interpretation: Given current counts, 'team' is much more likely to belong to the Sports topic on this update.

Practical Tips

- Choose K via coherence metrics/perplexity and validation.
- After fitting, label topics via top-weighted words; inspect per-doc θ_d.
- Use tools like pyLDAvis to examine term relevance and topic separation.

Pitfalls

- Sensitive to preprocessing (bigrams, vocabulary pruning); short docs can be noisy.
- Topic interpretability varies; multiple good local optima.