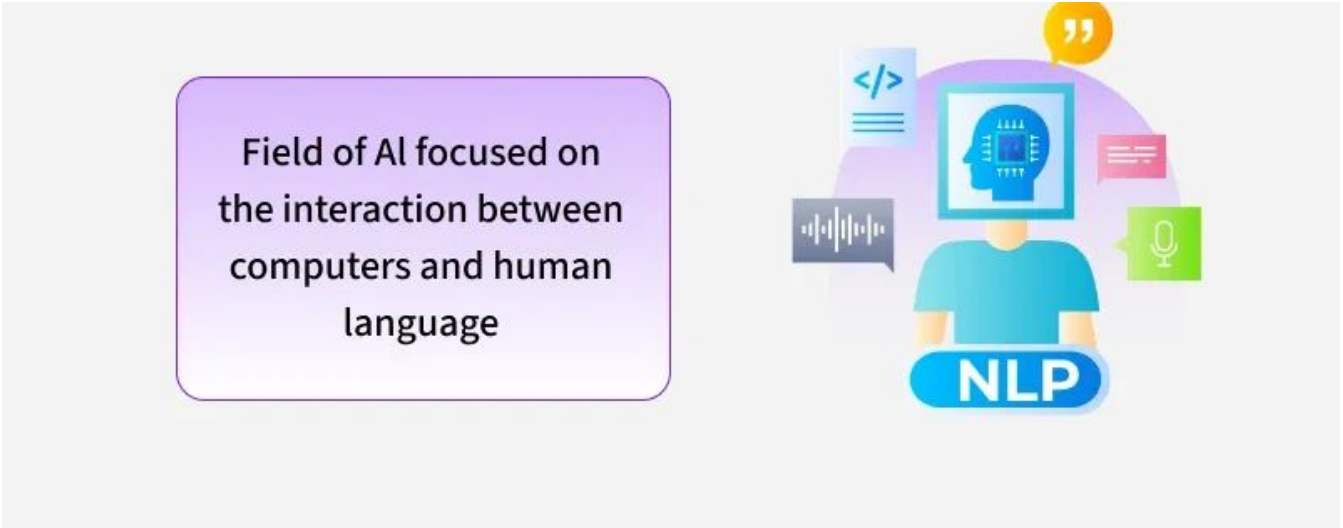
What is Human Language?

- A complex system for communication involving structured components (grammar, vocabulary).
- It is nuanced, contextual, and constantly evolving (e.g., new slang, technologies).

What is Natural Language Processing (NLP)?

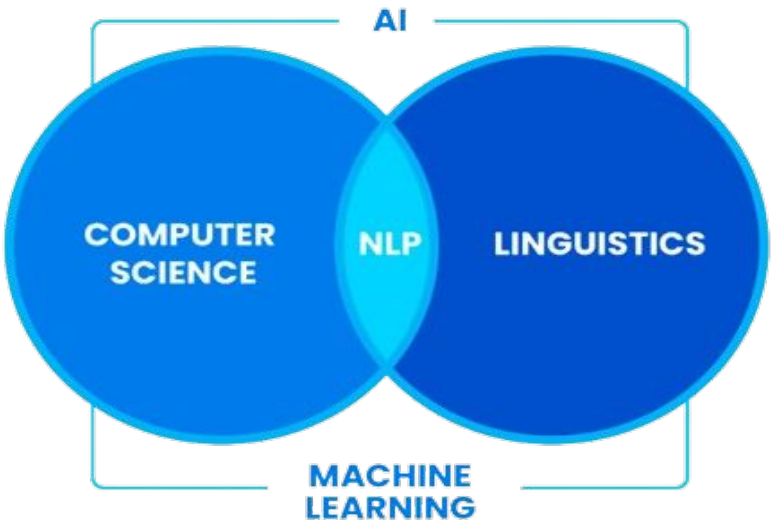
- Definition: Teaching computers to read, understand, interpret, and generate human languages (natural languages).
- It’s the field that allows machines to communicate with us.



NLP: An Interdisciplinary Field

NLP sits at the intersection of three major areas:

- Computer Science: Algorithms and data structures.
- Artificial Intelligence: Machine Learning and Deep Learning models.
- Linguistics: Grammar, semantics (meaning), and syntax (structure).



The Core Goal of NLP

- To convert unstructured human text (raw data like emails, articles, or tweets) into structured, machine-readable data.
- This structured data can then be analyzed, categorized, and acted upon.

The Problem: Why is NLP Hard?

- Text is inherently messy, informal, and ambiguous.
- Humans use context, world knowledge, and intuition; computers only see symbols.

Lexical Ambiguity

The presence of two or more possible meanings within a single word.

"I saw her duck."

Syntactic Ambiguity

The presence of two or more possible meanings within a single sentence or sequence of words.

"The chicken is ready to eat."

Example 1: Lexical Ambiguity (Words)

The word "bank" can mean:

A financial institution ("I put money in the bank.")

The edge of a river ("The river bank overflowed.")

The computer needs context (the surrounding words) to differentiate.

Example 2: Syntactic Ambiguity (Structure)

- Consider: *“The man saw the woman with the telescope.”*
- Did the man use the telescope? Or did he see a woman who possessed the telescope?
- Computers struggle with structural parsing without clear rules.

Example 3: Referential Ambiguity: When the subject is pointed to more than once in a sentence.

e.g., the boy told his father about the theft. He was very upset.

Here, who does "he" refer to??

Example 3: Anaphoric Ambiguity: A phrase or word refers to something previously mentioned, but there is more than one possibility.

e.g. - Margaret invited Susan for a visit, but she told her she had to go to work.

Here "she" and "her" can be used for Susan and Margaret interchangeably, which gives two different meanings.

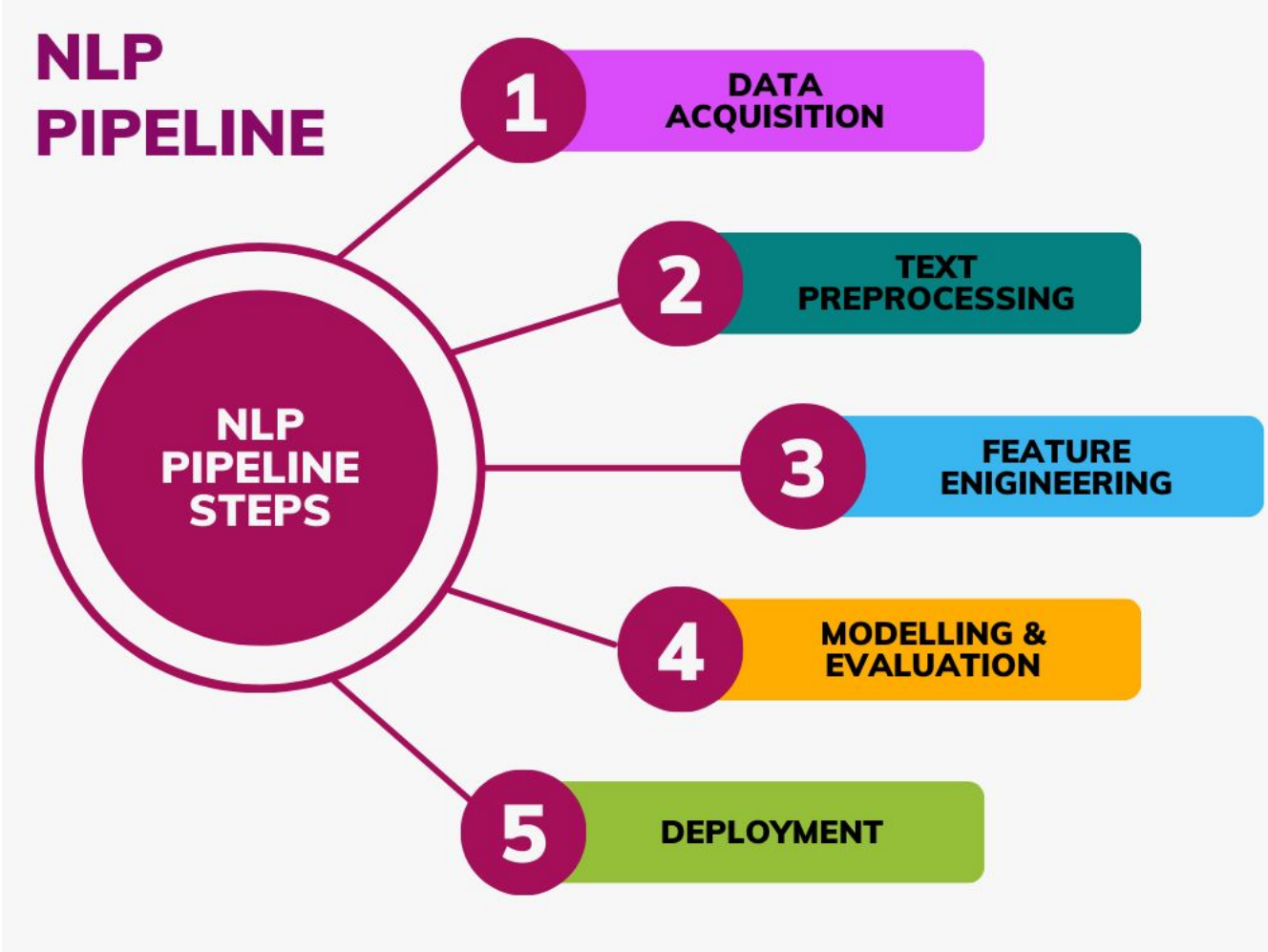
Example 4: Pragmatic Ambiguity: when the statement is not specific, and the context does not provide the information needed to clarify the statement. (Some Information is Missing)

e.g. - “The police are coming.”

Does this mean they are coming for you/me (I hope they better not 👀), or are they coming down the road ?

Solutions in NLP

- **Contextual Models:** Word embeddings (like Word2Vec, BERT) capture word meaning from surrounding text.
- **Disambiguation Techniques:** Algorithms like Part-of-Speech (POS) tagging and Word Sense Disambiguation (WSD) determine the correct meaning.
- **Statistical & Neural Parsers:** Use patterns and context to build the most probable sentence structure.
- **Attention Mechanisms:** Help models focus on relevant parts of the input to resolve ambiguity.


NLP
PIPELINE
NLP
PIPELINE
STEPS
1
DATA
ACQUISITION
2
TEXT
PREPROCESSING
3
FEATURE
ENIGINEERING
4
MODELLING &
EVALUATION
5
DEPLOYMENT

The NLP Pipeline Overview

The general flow for almost every NLP task:

1. Input (Raw Text)
2. Preprocessing (Cleaning)
3. Feature Extraction (Converting to Numbers)
4. Modeling (Machine Learning)
5. Output (Result/Prediction)

Text Preprocessing

The Necessity: Raw text is noisy, inconsistent, and full of irrelevant information.

Preprocessing is the vital first step to clean and normalize the data for the model.

Step 1: Document/Sentence Segmentation

- Breaking a large document (e.g., a book or an article) into manageable units.
- First into paragraphs, and then into individual sentences.
- Models often process sentences one at a time.

Step 2: Tokenization

- The fundamental process of splitting text into smaller, discrete units called tokens.
- Tokens are the base units of text we feed to the machine.

Tokens are usually words.

The simplest form: splitting by whitespace.

Example: "Hello world!" : [Hello, world, !]

Punctuation is often tokenized separately.

Tokens can be Subwords (BPE)

For dealing with a very large or rare vocabulary, we split words into meaningful parts.

Example: 'unbelievable': [un, believe, able]

This keeps the total vocabulary size small and manageable.

Tokens can be characters.

- Sometimes, we treat every letter as a token.
- Useful for tasks like spelling correction or recognizing non-standard words (e.g., URLs, usernames).

Step 3: Lowercasing

- Converting all tokens to lowercase (e.g., 'Apple' : 'apple').
- Ensures the model treats different capitalizations of the same word equally, reducing feature space.

When to AVOID Lowercasing

- If the task is Named Entity Recognition (NER).
- Capitalization is a strong clue: "May" (month) vs. "may" (verb) or *"Apple" (company) vs. "apple" (fruit).

Step 4: Removing Noise (Punctuation)

- Removing commas, periods, quotes, or HTML tags if they don't contribute to the core meaning.
- Reduces noise, especially for simple classification tasks.

When to KEEP Punctuation

- For Sentiment Analysis, punctuation provides emotional context.
- "This is good." is different from "This is good!" or "This is good..."

Step 5: Stop Word Removal

- Stop words are extremely common, high-frequency words (e.g., a, an, the, is, of).
- They often carry little unique semantic value.

Why Remove Stop Words?

- Reduces the total number of features (vocabulary size).
- Speeds up training by forcing the model to focus on the content-bearing keywords.

When to AVOID Stop Word Removal

- For grammar-dependent tasks like machine translation or text generation.
- The absence of stop words can destroy grammatical fluency.

Step 6: Stemming

1. A rule-based, heuristic process of reducing a word to its root or stem.
2. The stem is often not a real dictionary word.

- Stemming Example (Porter Algorithm)
  - Running: run
  - Universities: univers
  - Finally: final

It's fast but crude and linguistically inaccurate.

Step 7: Lemmatization

- A more sophisticated process using vocabulary and morphological analysis.
- Reduces a word to its base, dictionary form (lemma).

Lemmatization Example (WordNet)

running : run

better : good

was : be

The output is always a valid word.

Stemming vs. Lemmatization

- Stemming: Faster, less accurate, rule-based.
- Lemmatization: Slower, uses linguistic context, more accurate.
- Choose lemmatization when precision is critical.

Normalization & Other Cleanups

- Removing numbers (if irrelevant),
- correcting common misspellings (e.g., loooove : love).
- Expanding contractions (don't: do not).
- Handling or translating emojis.

Preprocessing Summary

- We have converted messy, raw text into a clean, normalized, and consistent list of tokens.
- This clean list is the essential foundation for all subsequent steps.

Feature Engineering

From Words to Numbers

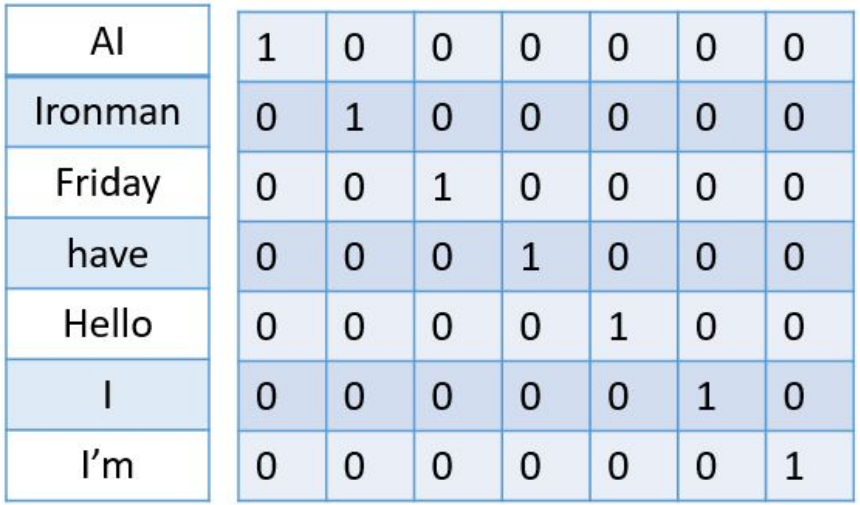
- The Core Problem: Machine learning models only work with numerical data (vectors/matrices).
- We must convert our tokens into a machine-understandable numerical representation (feature vectors).

Representation 1: One-Hot Encoding

- Each word is represented by a vector where only one dimension is '1' (its index) and all others are '0'.
- The vector size equals the vocabulary size.

Hello, I'm Ironman. I have Friday AI

One Hot Encoding



AI	1	0	0	0	0	0	0
Ironman	0	1	0	0	0	0	0
Friday	0	0	1	0	0	0	0
have	0	0	0	1	0	0	0
Hello	0	0	0	0	1	0	0
I	0	0	0	0	0	1	0
I'm	0	0	0	0	0	0	1

One-Hot Encoding Issues

- Sparsity: Vectors are mostly zeroes (inefficient).
- Scale: Vocabulary size can be enormous (100,000+ dimensions).
- No Semantic Meaning: 'Dog' and 'Cat' are equally distant (no similarity captured).

Representation 2: Bag of Words (BoW)

- Documents are represented by the frequency of words they contain.
- Crucially, the order of words and grammar are completely ignored ("bag" of words).

BoW: The Vocabulary Matrix

- A matrix is created where rows are documents and columns are unique words in the vocabulary.
- Values are the counts of each word in each document.

Bag of Words Model explanation

Corpus

A collection of text documents

↓

Tokenize

Divide the text into smaller units called tokens, usually words or phrases

↓

Count word frequencies

Create a vocabulary of unique words and count the number of times each word appears in each document.

↓

Encode the data

Encoding the text data as numerical values by creating a vector for each document, with each element of the vector representing the frequency count of a particular word in the document

Example

Corpus:

The dog is happy. The child makes the dog happy. The dog makes the child happy

↓

Tokenization:

D1: [The] [dog] [is] [happy]
D2: [The] [child] [makes] [the] [dog] [happy]
D3: [The] [dog] [makes] [the] [child] [happy]

↓

Documents	Counting word frequencies
D1	the: 1, dog: 1, is: 1, happy: 1
D2	the: 2, dog: 1, makes: 1, child: 1, happy: 1
D3	the: 2, child: 1, makes: 1, dog: 1, happy: 1

↓

Encode	child	dog	happy	is	makes	the	BoW Vector representations
D1	0	1	1	1	0	1	**[0,1,1,1,0,1]**
D2	1	1	1	0	1	2	**[1,1,1,0,1,2]**
D3	1	1	1	0	1	2	**[1,1,1,0,1,2]**

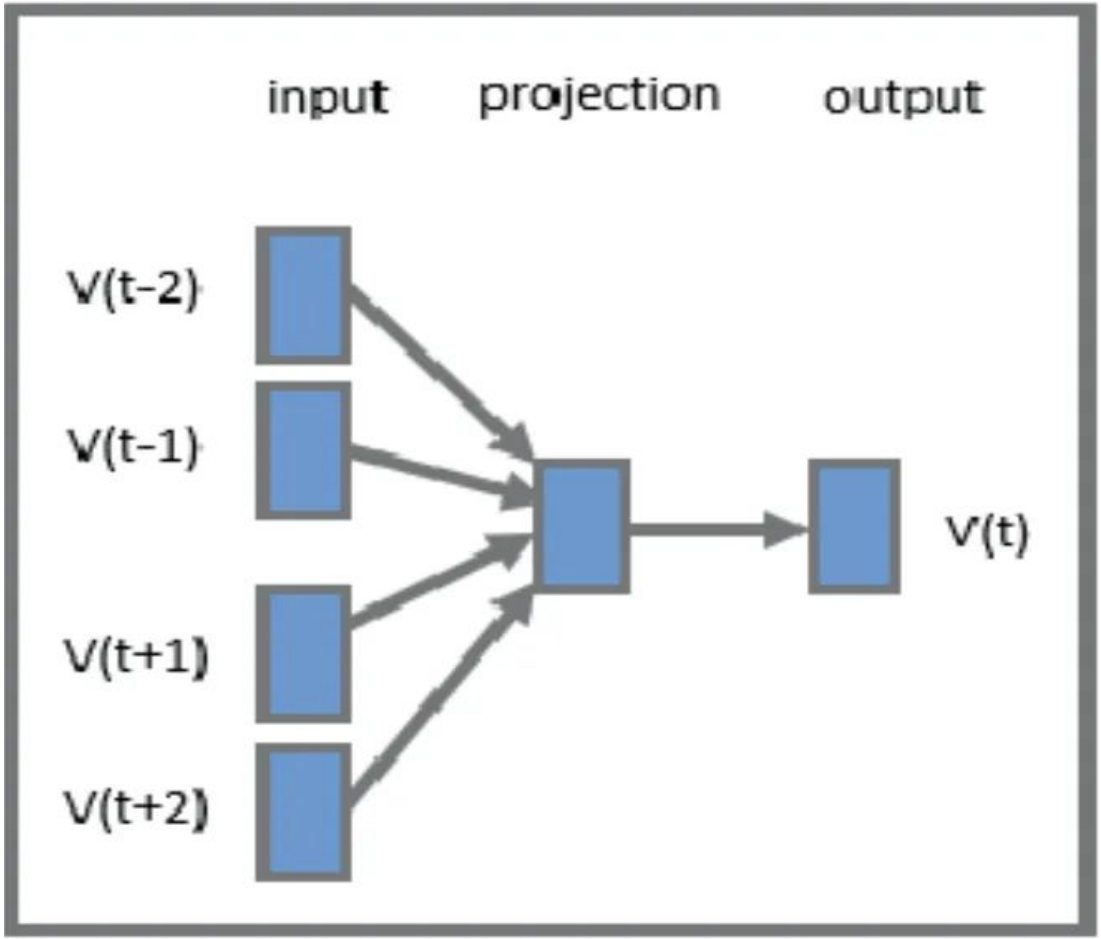
BoW Example: "The book is good."

- If the vocabulary is {the, book, is, good, terrible}.
- The document vector is **[1, 1, 1, 1, 0]**. (1 count for each word present).

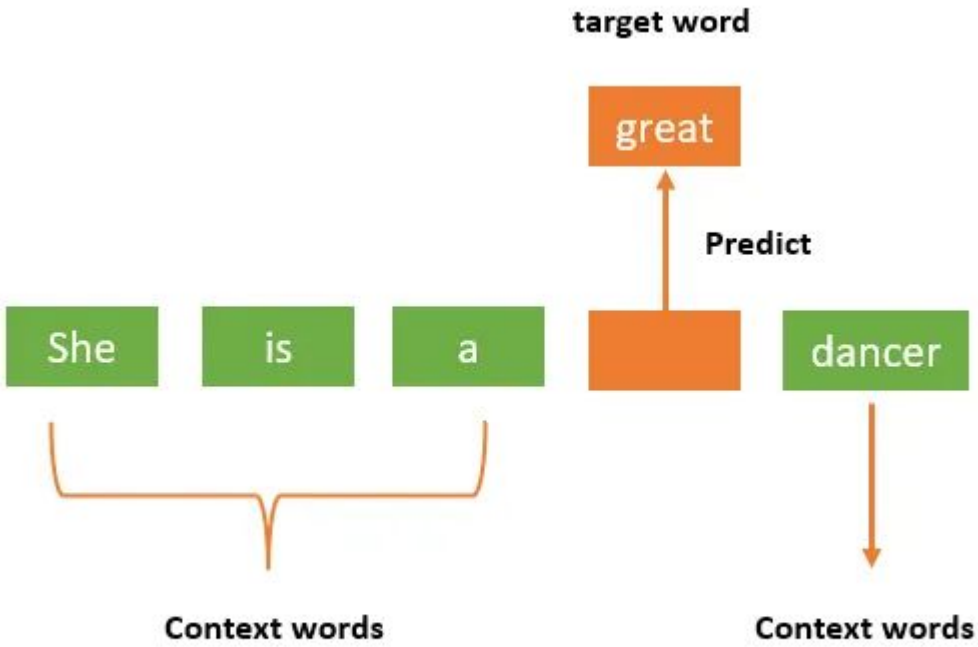
Limitations of BoW

- **Loss of Order:** "The dog bit the man" is identical to "The man bit the dog."
- **Lack of Context:** Fails on negations and sarcasm (e.g., "The food was *not* good" still contains the word "good").

Word2Vec Embedding



Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.


target word
great
Predict
She
is
a
dancer
Context words
Context words

Capturing Context:

- N-grams Sequences of N consecutive words are used as tokens/features instead of just single words. This approach captures local word order.

Unigrams ($N=1$) Individual words (the same as the basic BoW approach).

Example: [I, am, happy]

Bigrams (N=2)

- Pairs of adjacent words.
- *Example: "I am happy" $\rightarrow$ [I am, am happy]*
- Helpful for capturing phrases like "New York" or "not good."

Trigrams (N=3)

- Sequences of three adjacent words.
- Captures even more local context but increases the feature space significantly.

- In summary, Continuous Bag of Words (CBOW) is a Word2Vec architecture that predicts a target word based on the context words within a fixed window size.

- It learns word embeddings by training a neural network to maximize the likelihood of predicting the target word given its context.

BoW Refinement: Term Frequency (TF):

The raw count of a word t in a document d.

Often normalized by total words: $TF(t, d) = \text{Count}(t)/\text{Total words}.$

Document Frequency (DF)

- The number of documents in the entire corpus that contain the word t.
- Common words (like stop words) will have a high DF.

Inverse Document Frequency (IDF)

- Measures how unique or important a word is across the whole corpus.
- Common words get a very low IDF score. Rare words get a high IDF score.

The IDF Formula

- The formula downweights words that appear in many documents.
- $IDF(t) = \log\left(\frac{N}{DF(t)}\right)$
  - N: Total documents.
  - $DF(t)$: Documents containing term t.
- $$\mathrm{IDF}(t) = \log\left(\frac{N}{DF(t)}\right)$$

TF-IDF (Term Frequency-Inverse Document Frequency)

- The final weighting score: $TFIDF(t,d) = TF(t,d) \times IDF(t)$.
- $$\mathrm{TFIDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

What TF-IDF Does

- It emphasizes words that are frequent in a specific document (high TF) but rare across the entire corpus (high IDF).
- Excellent for keyword extraction and document similarity.

The Semantic Problem Persists

- All count-based models (BoW, TF-IDF) still treat words like 'King' and 'Queen' as completely unrelated features.
- There is no representation of the conceptual similarity.

The Solution: Word Embeddings

- The crucial shift from high-dimensional, sparse, count-based vectors to low-dimensional, dense, meaning-based vectors.

Feature Engineering is Complete.

We have learned how to convert text numerically using counts (BoW/TFIDF) and introduced the concept of dense meaning-vectors (Embeddings).

Word Embeddings

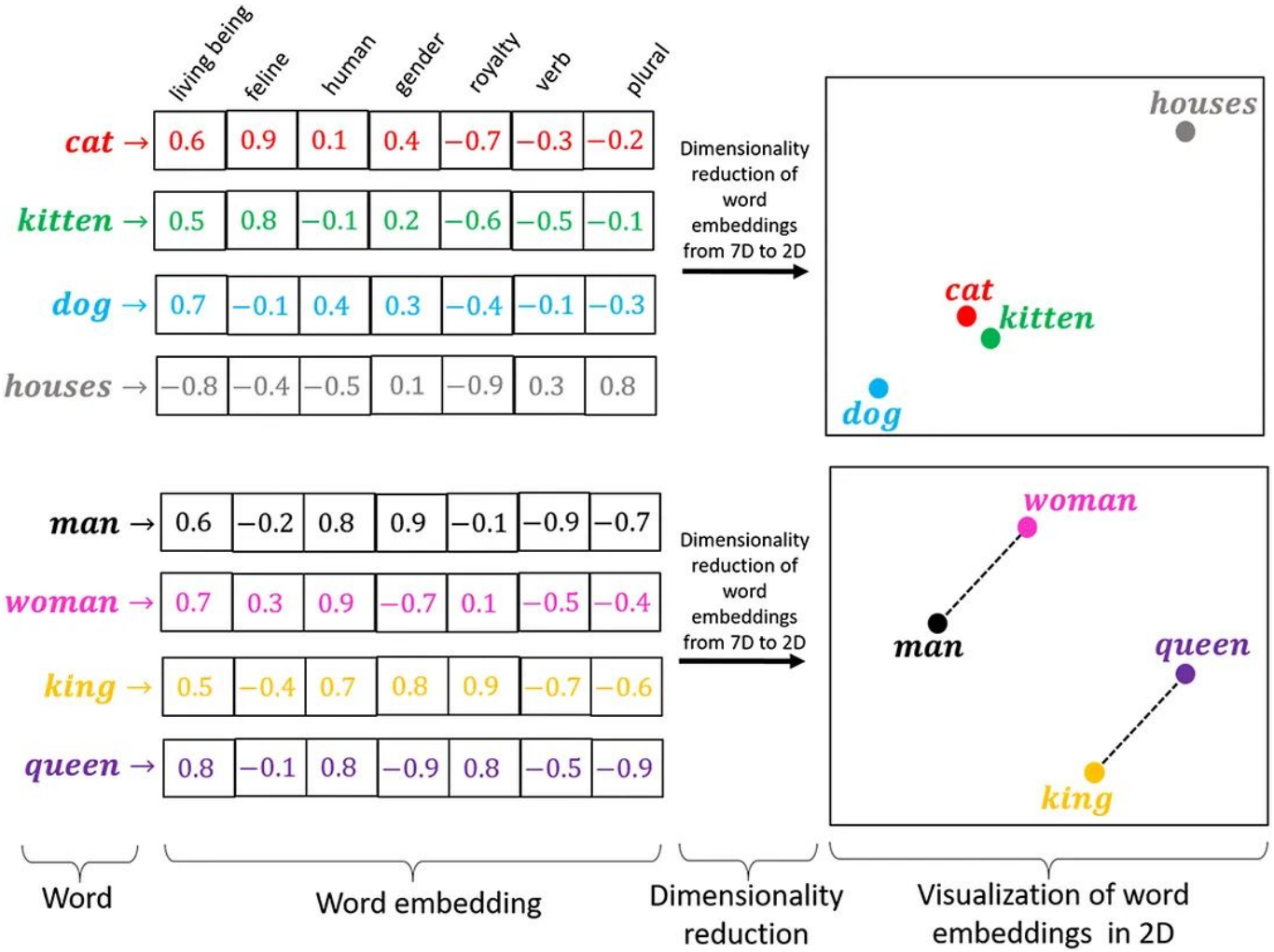
Goal: To create a vector space where words with similar meanings (semantics) are located closer together.

What is a Word Vector?

- A dense, low-dimensional vector (e.g., 50, 100, or 300 dimensions).
- Each number in the vector contributes to the word's total meaning.

"You shall know a word by the company it keeps" (J. R. Firth).

Words that appear in similar contexts (surrounded by similar words) have similar meanings.


living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Visualization in 2D

- If we could project the 300D vectors into 2D, we would see clusters of related concepts.
- Distance in this space correlates with semantic similarity.

Word2Vec

- Developed by Google (Mikolov et al., 2013).
- A shallow, two-layer neural network designed specifically to learn word associations from a large text corpus

.Word2Vec Architecture 1: CBOW

Continuous Bag of Words.

The model predicts the target word from a window of surrounding context words.

Word2Vec Architecture 2: Skip-Gram

The model predicts the surrounding context words given the target word.

Skip-gram is generally better for capturing rare word semantics.

The Most Famous Property: Vector Arithmetic

The relationships between words are preserved in the geometry of the vectors.

This property revealed deep, learned semantic knowledge.

- $Vector(\text{King}) - Vector(\text{Man}) + Vector(\text{Woman}) \approx Vector(\text{Queen}).$
- The model has learned the **gender** and **royalty** axes.

GloVe (Global Vectors for Word Representation)

- An alternative, popular model developed at Stanford.
- Trains on global word co-occurrence statistics across the entire corpus, combining the ideas of matrix factorization and neural methods.

Pre-trained Embeddings

- We rarely train Word2Vec/GloVe from scratch.
- Instead, we use weights trained on massive corpora (e.g., billions of tokens from Wikipedia).
- Saves time and data; a form of transfer learning.

Document Embedding: Simple Averaging

- How do we get a vector for a whole sentence or document?
- The simplest way is to average the vectors of all the words it contains.

The Flaw of Static Embeddings (Word2Vec, GloVe)

- Static embeddings mean the word 'bank' has *one* single vector, regardless of its context.
- They cannot handle the ambiguity we saw.

The Need for Contextual Embeddings

The representation (vector) of a word must dynamically change based on the sentence it appears in.

The vector for 'bank' in "river bank" must be different from "money bank."

Contextual Embeddings: A Teaser

This challenge led to the creation of the Transformer architecture .

Modern models (BERT, GPT) generate a unique vector for every word in every sentence.

Core NLP Tasks

Overview of the fundamental problems NLP systems are designed to solve.

These are the applications that drive industry value.

Task 1: Sentiment Analysis (SA)

Classifying the subjective feeling, opinion, or attitude expressed in a piece of text.

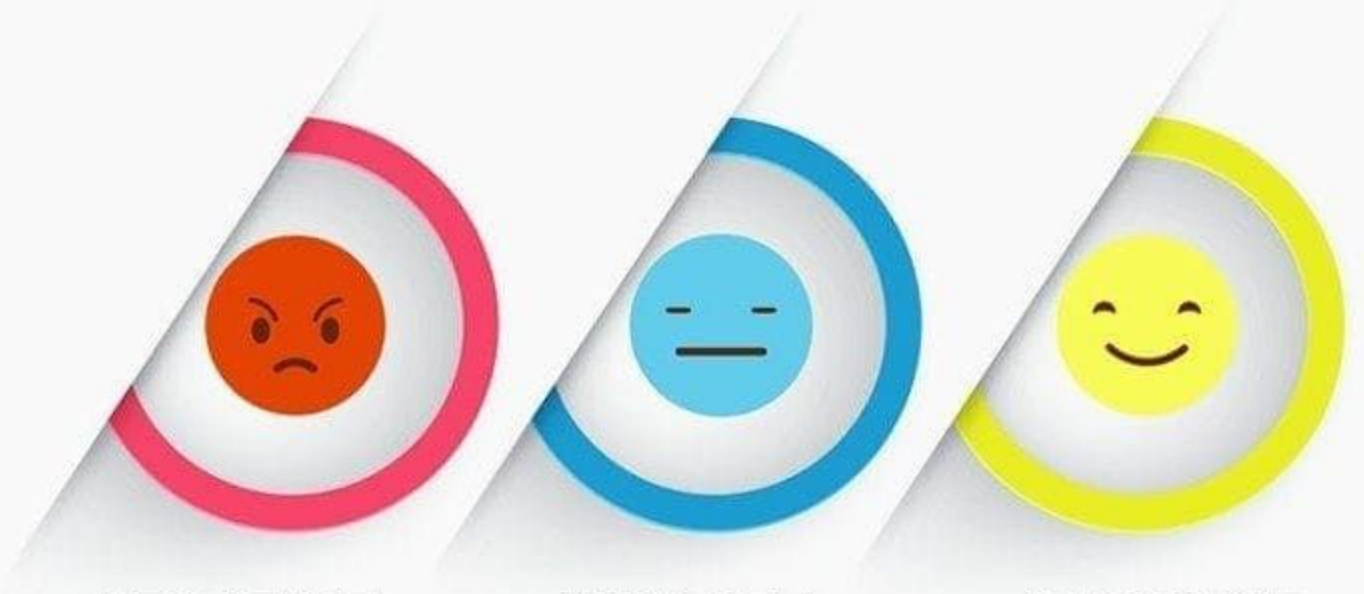
Used for brand monitoring, customer service triage, and market research.

SA: Polarity Classification

The simplest form: Binary (Positive/Negative).

More complex: Multi-Class (Happy/Sad/Angry/Neutral).

SENTIMENT ANALYSIS



NEGATIVE

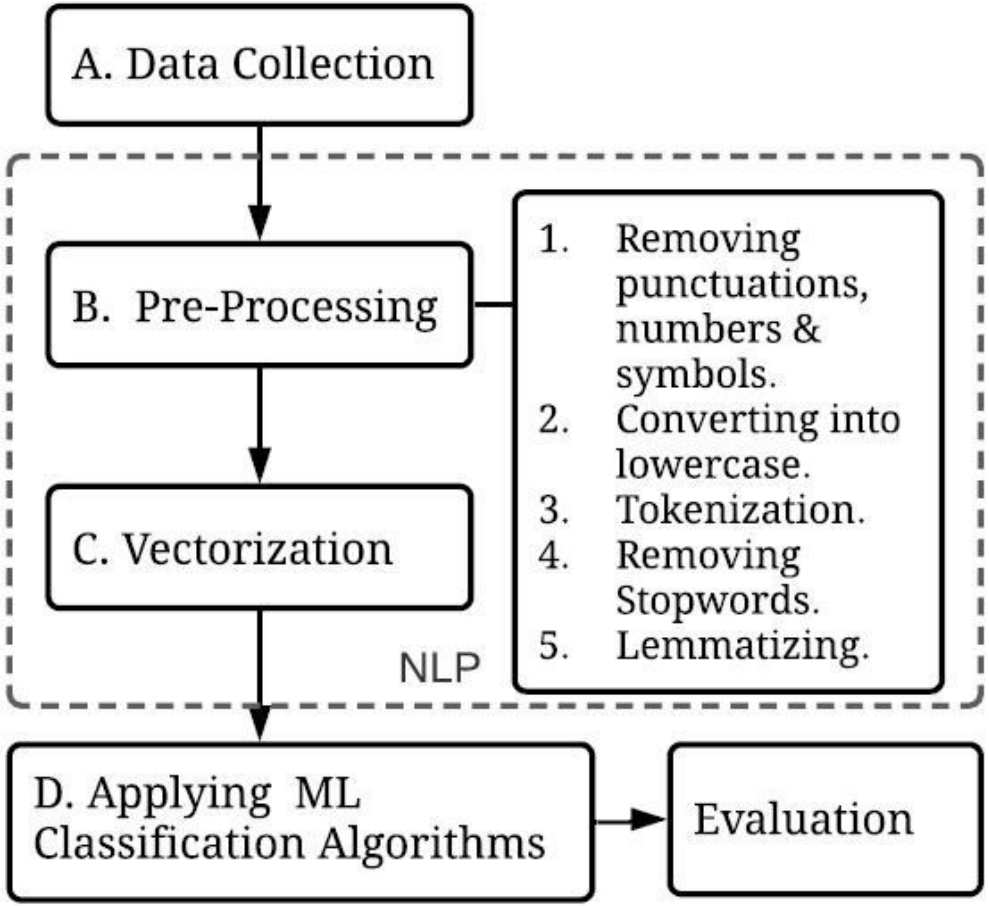
Totally dissatisfied with the service. Worst customer care ever.

NEUTRAL

Good Job but I will expect a lot more in future.

POSITIVE

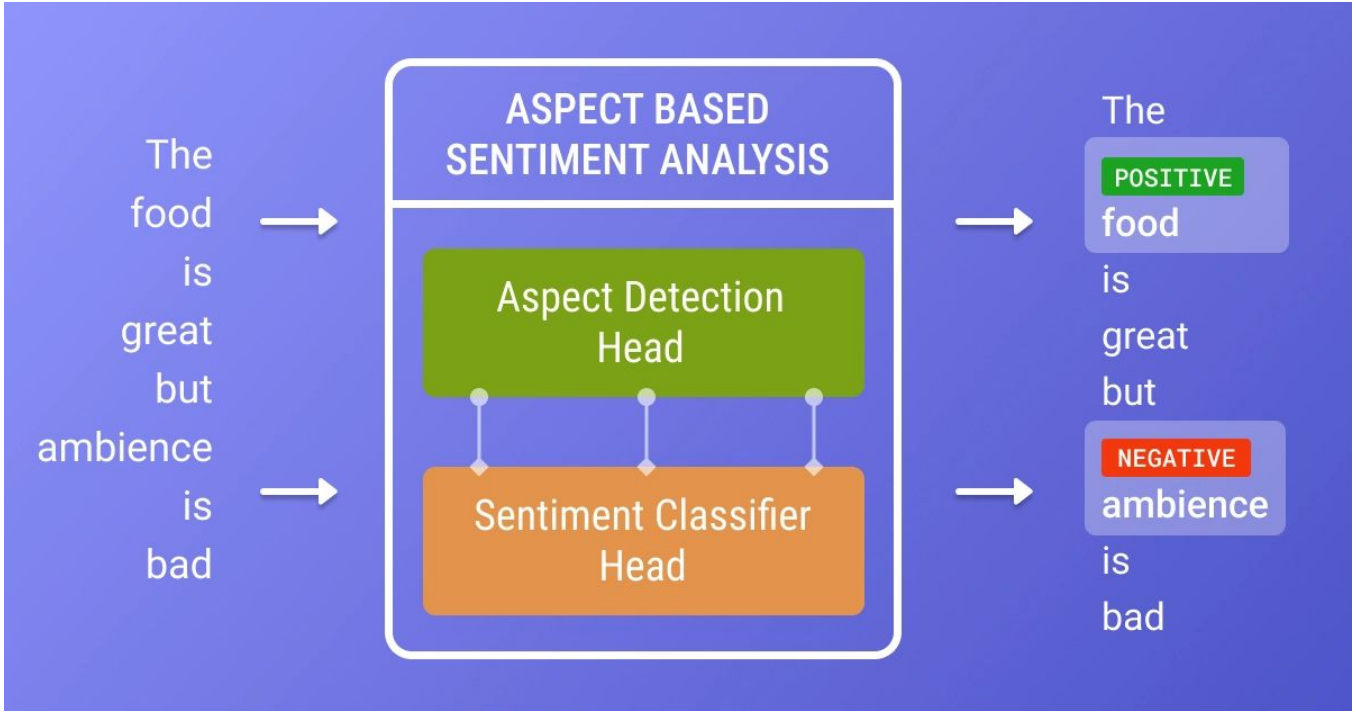
Brilliant effort guys! Loved Your Work.


A. Data Collection
B. Pre-Processing
C. Vectorization
NLP
1. Removing punctuations, numbers & symbols.
2. Converting into lowercase.
3. Tokenization.
4. Removing Stopwords.
5. Lemmatizing.
D. Applying ML Classification Algorithms
Evaluation

SA: Aspect-Based

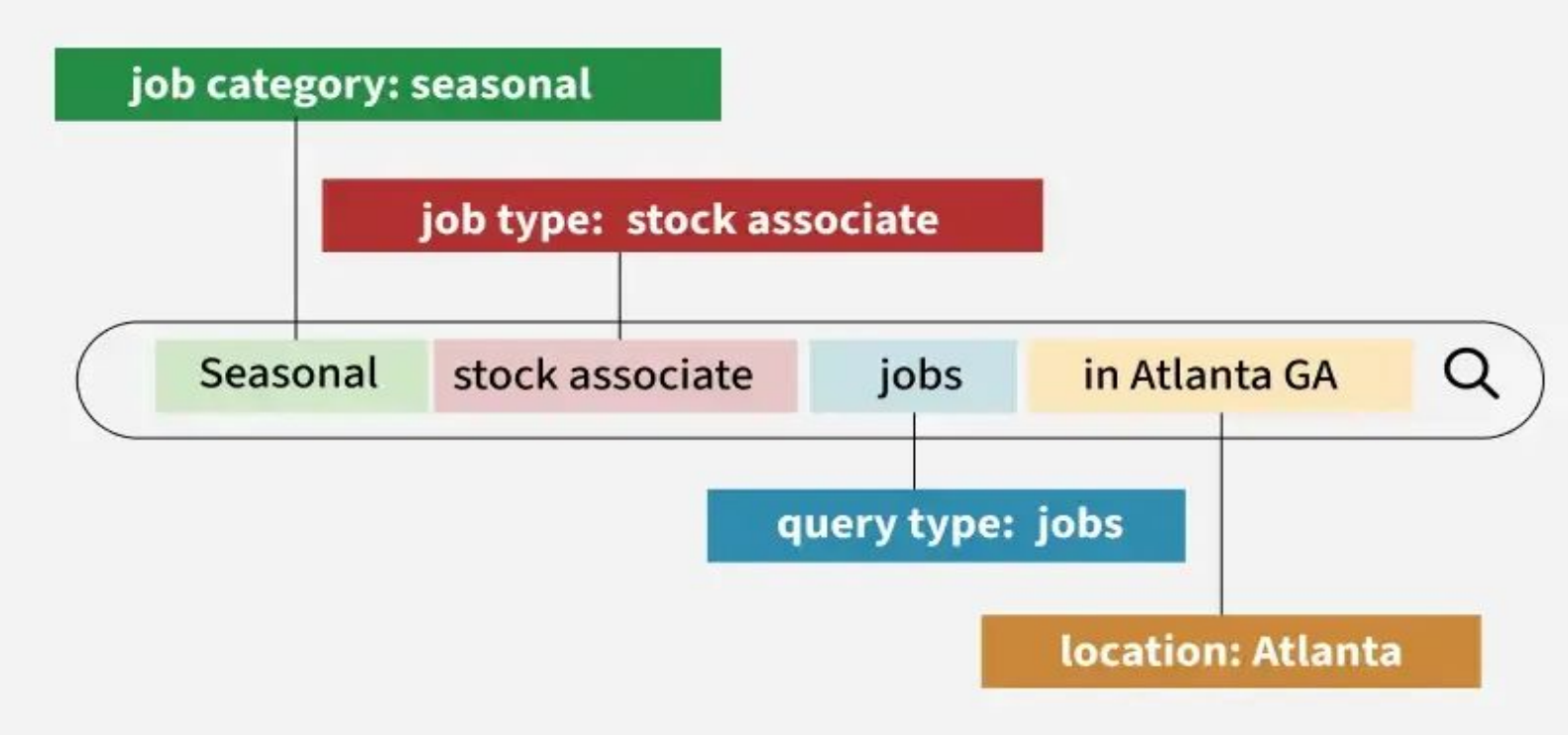
Identifying the sentiment directed towards specific aspects of an entity.

Example: "The food was great (positive), but the service was terrible (negative)."



Task 2: Named Entity Recognition (NER)

- Locating and classifying named entities into predefined categories.
- It's a sequence labeling task.



NER Categories

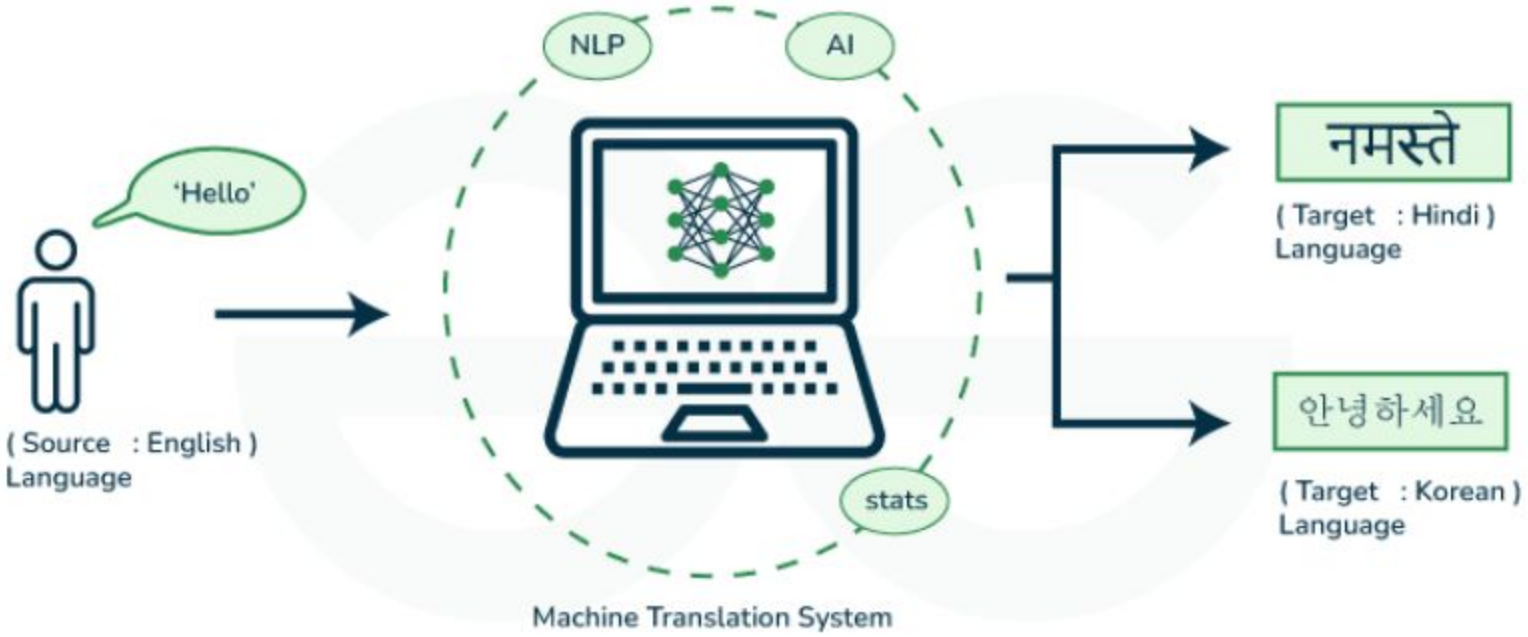
- Common categories:
  - **PER** (Person: *Jeff Bezos*)
  - **ORG** (Organization: *Amazon*)
  - **LOC** (Location: *Seattle*)
  - **DATE**, **MONEY**, etc.

Task 3: Part-of-Speech (POS) Tagging

- Assigning a grammatical category (e.g., noun, verb, adjective, adverb) to every word in a sentence.

POS Tagging Example

- *The (DT: Determiner) quick (JJ: Adjective) brown (JJ: Adjective) fox (NN: Noun) jumps (VBZ: Verb).*
- Crucial for advanced parsing and linguistic analysis.


NLP
AI
'Hello'
(Source : English) Language
stats
Machine Translation System
नमस्ते
(Target : Hindi) Language
안녕하세요
(Target : Korean) Language

Task 4: Machine Translation (MT)

- Converting text from a source language to a target language while preserving the meaning and style.
- Requires both language understanding and language generation.

Neural MT

- Modern MT relies almost exclusively on **Sequence-to-Sequence (Seq2Seq)** models.
- These models are composed of an **Encoder** (understands source) and a **Decoder** (generates target).

Task 5: Text Summarization

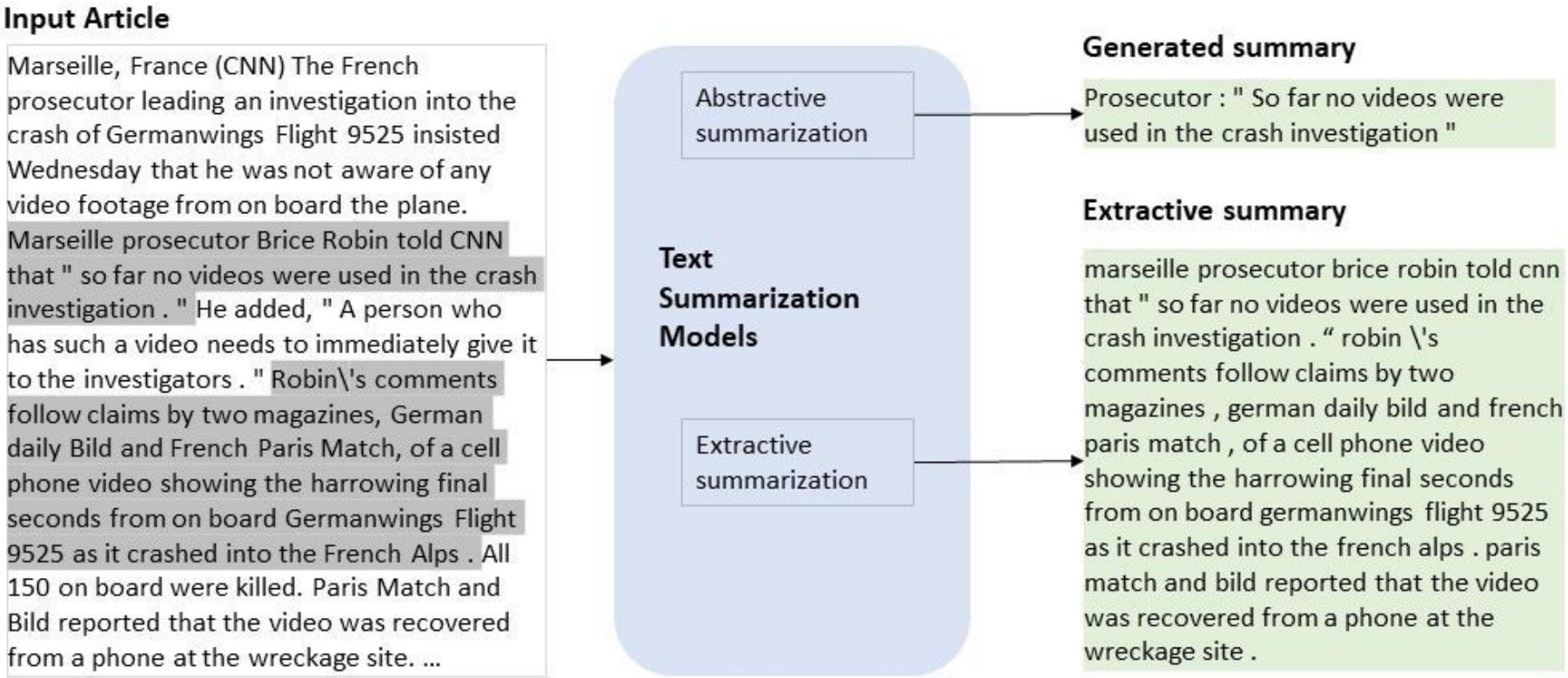
- Condensing a large document into a shorter, coherent, and informative version.
- Essential for managing information overload.

Summarization Type 1: Extractive

- Selects the most important sentences directly from the source text and stitches them together.
- Simpler and safer; guarantees grammatical correctness.

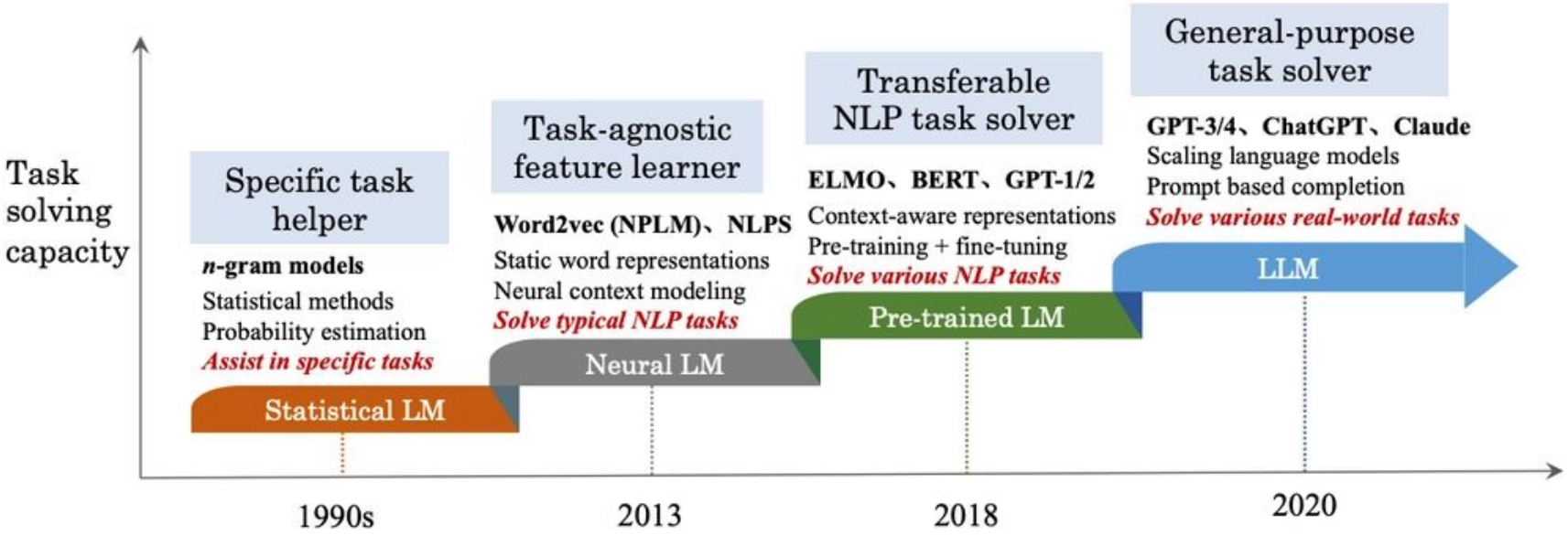
Summarization Type 2: Abstractive

- Generates completely new sentences and phrases, often rephrasing the source material.
- Requires deep understanding; more difficult but produces better quality.


Input Article
Marseille, France (CNN) The French prosecutor leading an investigation into the crash of Germanwings Flight 9525 insisted Wednesday that he was not aware of any video footage from on board the plane. Marseille prosecutor Brice Robin told CNN that " so far no videos were used in the crash investigation . " He added, " A person who has such a video needs to immediately give it to the investigators . " Robin\'s comments follow claims by two magazines, German daily Bild and French Paris Match, of a cell phone video showing the harrowing final seconds from on board Germanwings Flight 9525 as it crashed into the French Alps . All 150 on board were killed. Paris Match and Bild reported that the video was recovered from a phone at the wreckage site. ...
Text Summarization Models
Abstractive summarization
Extractive summarization
Generated summary
Prosecutor : " So far no videos were used in the crash investigation "
Extractive summary
marseille prosecutor brice robin told cnn that " so far no videos were used in the crash investigation . “ robin \'s comments follow claims by two magazines , german daily bild and french paris match , of a cell phone video showing the harrowing final seconds from on board germanwings flight 9525 as it crashed into the french alps . paris match and bild reported that the video was recovered from a phone at the wreckage site .

Task 6: Language Modeling (LM)

- The probabilistic task of predicting the next word in a sequence, given the preceding words
- Language modeling is the fundamental engine behind all modern text generation systems.
- This includes chatbots, code completion, and creative writing tools.


Task solving capacity
Specific task helper
n-gram models
Statistical methods
Probability estimation
Assist in specific tasks
Statistical LM
Task-agnostic feature learner
Word2vec (NPLM)、NLPS
Static word representations
Neural context modeling
Solve typical NLP tasks
Neural LM
Transferable NLP task solver
ELMO、BERT、GPT-1/2
Context-aware representations
Pre-training + fine-tuning
Solve various NLP tasks
Pre-trained LM
General-purpose task solver
GPT-3/4、ChatGPT、Claude
Scaling language models
Prompt based completion
Solve various real-world tasks
LLM
1990s
2013
2018
2020

Neural Networks for NLP

- We need powerful, complex models to process and learn from the sequential and contextual nature of language.
- Deep Learning is the current state-of-the-art.

The Simple Neuron (Perceptron)

- The basic building block: takes inputs, applies weights (w_i), sums them, adds a bias (b), and passes the result through an activation function.

Feed-Forward Networks (MLPs)

- Stacking layers of neurons. Useful for basic classification (e.g., BoW sentiment).
- **Limitation:** Treats all inputs independently; cannot handle sequences where order matters.

- Language data (text) is sequential: $w_1 \rightarrow w_2 \rightarrow w_3 \ldots$
- The meaning of w_3 depends on w_1 and w_2. We need memory.

Recurrent Neural Networks (RNNs)

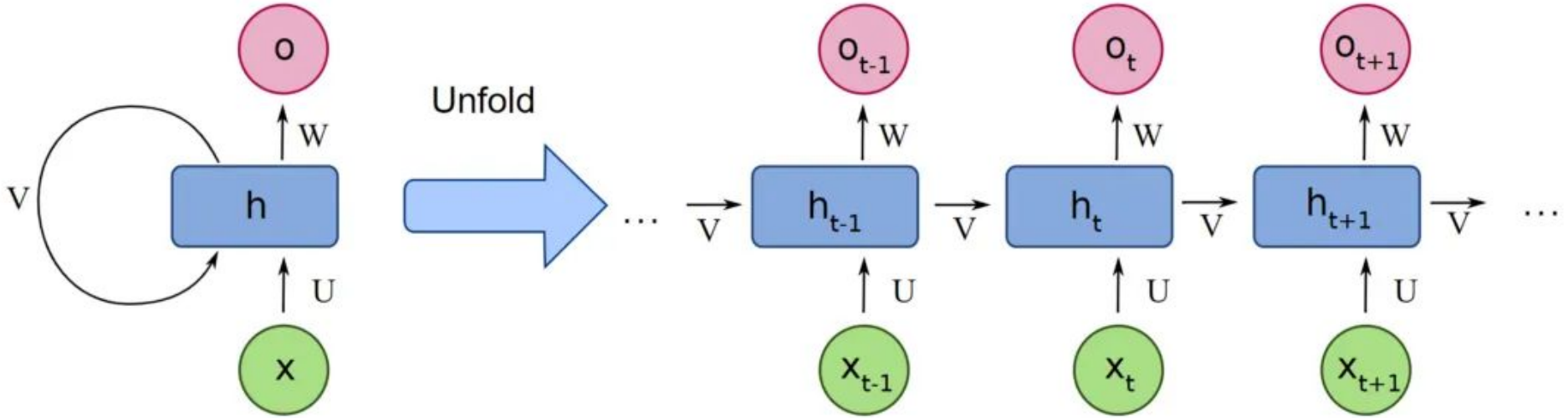
- Networks with a hidden state/memory loop.
- The output/hidden state from time step $t-1$ is passed as input to time step t.

he RNN Hidden State

- The 'memory' vector that attempts to capture everything learned so far in the sequence.
- $h_t = f(h_{t-1}, x_t)$.

The Vanishing Gradient Problem

- The 'memory' in simple RNNs is short-lived.
- Gradients become too small during training, causing the network to forget initial context (long-term dependencies).


o
W
V
h
U
x
Unfold
…
V
o_{t-1}
W
h_{t-1}
U
x_{t-1}
V
o_t
W
h_t
U
x_t
V
o_{t+1}
W
h_{t+1}
U
x_{t+1}
V
…

Long Short-Term Memory (LSTMs)

- A sophisticated type of RNN explicitly designed to combat vanishing gradients.
- They use a specialized cell structure to retain long-term memory.

LSTM: The Cell State

- An internal 'conveyor belt' that runs through the unit.
- Information can be added or removed from this state, allowing memory to persist across many time steps.

LSTM: The Three Gates

- LSTMs use internal gates (small neural networks) to control information flow:
  - **Forget Gate:** Decides what to discard from the cell state.
  - **Input Gate:** Decides what new information to store.
  - **Output Gate:** Decides what to output as the hidden state.

Attention Mechanism: The Breakthrough

- A fundamental idea: instead of processing all words equally, the model should dynamically focus on the *most relevant* parts of the input.
- "Pay Attention to what matters."

Attention in Seq2Seq

- In Machine Translation, the decoder pays attention to the most relevant source words while generating the next target word.
- *Example: generating "chat" in Spanish must focus on the source English word "cat."*

The Transformer Architecture (2017)

- Introduced in the paper "Attention Is All You Need."
- It completely abandoned recurrence (RNNs and LSTMs) and relied only on Attention.

The Core of the Transformer: Self-Attention

- Allows every word in the input sequence to attend to every other word in the *same* sequence.
- This instantly creates contextual embeddings for every word.

Self-Attention Example

- *Sentence: "The animal didn't cross the road because **it** was too tired."*
- Self-attention calculates that the word "**it**" attends strongly to "**animal**," resolving the ambiguity instantly.

Transformer Block: Multi-Head Attention

- The model runs the attention mechanism multiple times in parallel.
- Each "head" learns a different type of relationship (e.g., one for syntax, one for semantics).

BERT (Bidirectional Encoder)

- A groundbreaking model based on the Transformer **Encoder** block.
- Trained bidirectionally (left and right context). Excellent for understanding text (classification, NER, QA).

GPT (Generative Pre-trained Transformer)

- Based on the Transformer **Decoder** block.
- Trained autoregressively (predicting only forward). The engine of modern text generation and Large Language Models (LLMs).

Transfer Learning in NLP

- The modern paradigm:
  - **Pre-train** a large model (BERT, GPT) on a massive corpus (cheap text).
  - **Fine-Tune** the model on a small, specific, labeled dataset (expensive data).

- This saves massive amounts of computation time.

Summary & Next Steps

We covered the full NLP fundamentals pipeline:

- Preprocessing
- Feature Engineering (Embeddings)
- Deep Learning Models (Transformers).

And, then the “Beast” arrived in 2017!

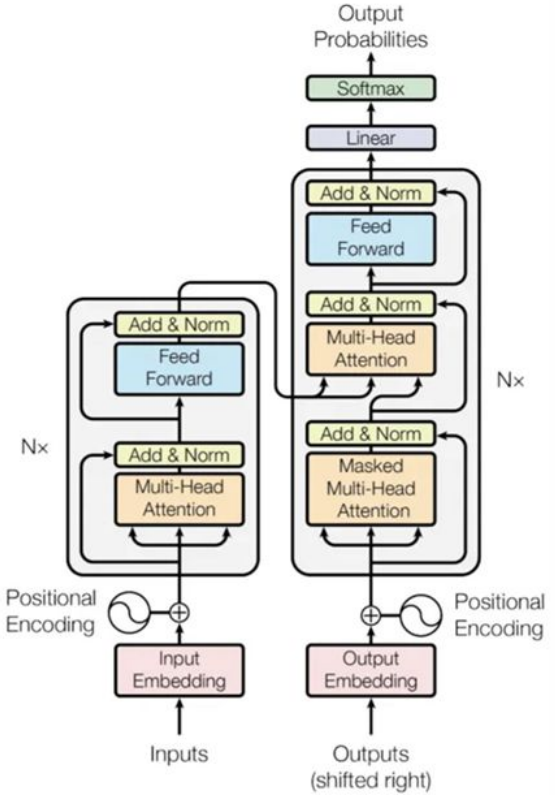
Attention is all you need

Authors: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin | Authors Info & Claims

NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems
Pages 6000 - 6010

Citations till Nov 2025: 2,05,000

- Replaced recurrence with *pure attention*
- Self-attention for sequence modelling
- Massive parallelism
- Positional Embeddings
- Multi-Head Attention
- Layer normalization + residual connections



“Large” Language Models

Think of LLMs as:

- A "super-autocomplete" on steroids
- A "linguistic chameleon" that has read and processed vast internet text

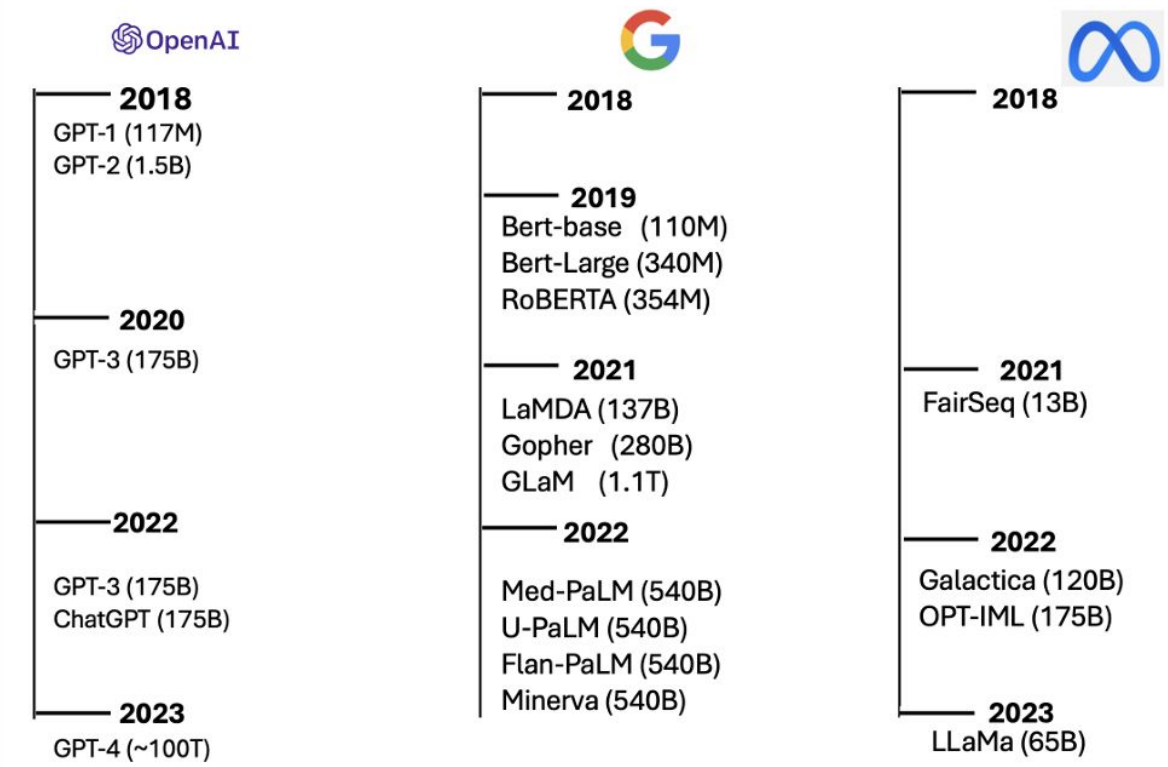
Key Characteristics

- Massive deep neural networks
- Trained on vast amounts of text data
- Understand, generate, and respond to human language

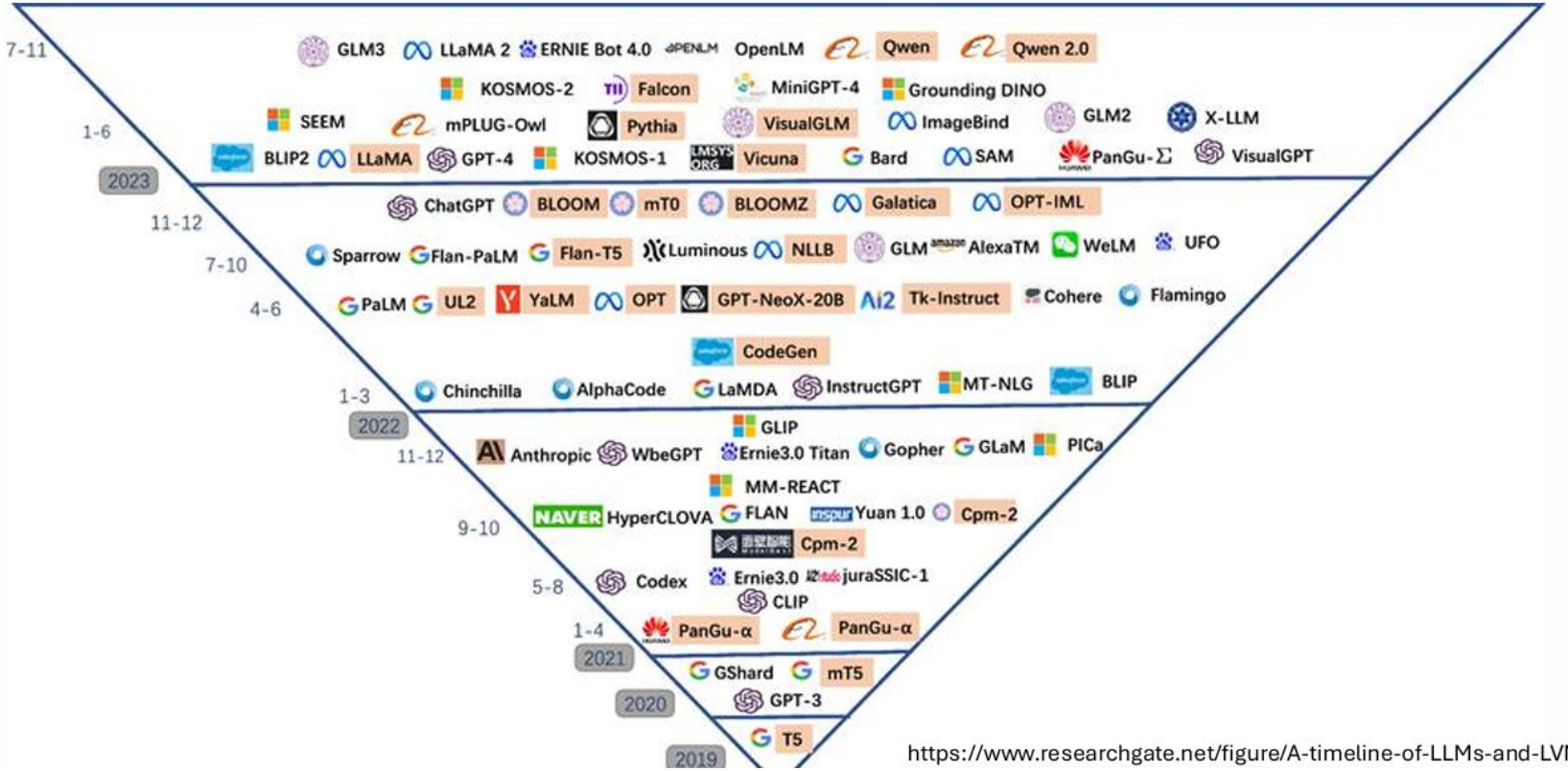
“LLMs learn relationships and patterns in language to perform tasks from writing essays to answering questions”

A Stochastic Parrot:
A system that statistically mimics text without real understanding [Bender et al., 2021]

And the “War” started !!

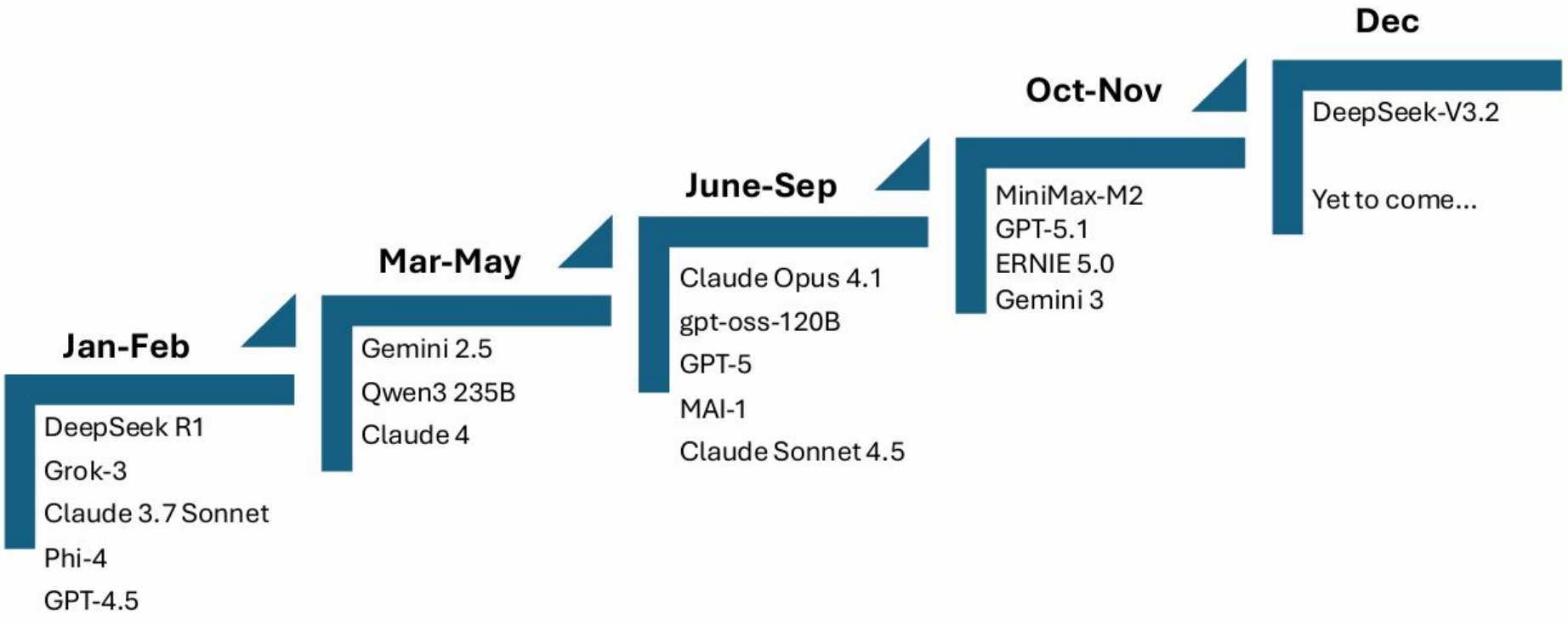


And the rest is history !!

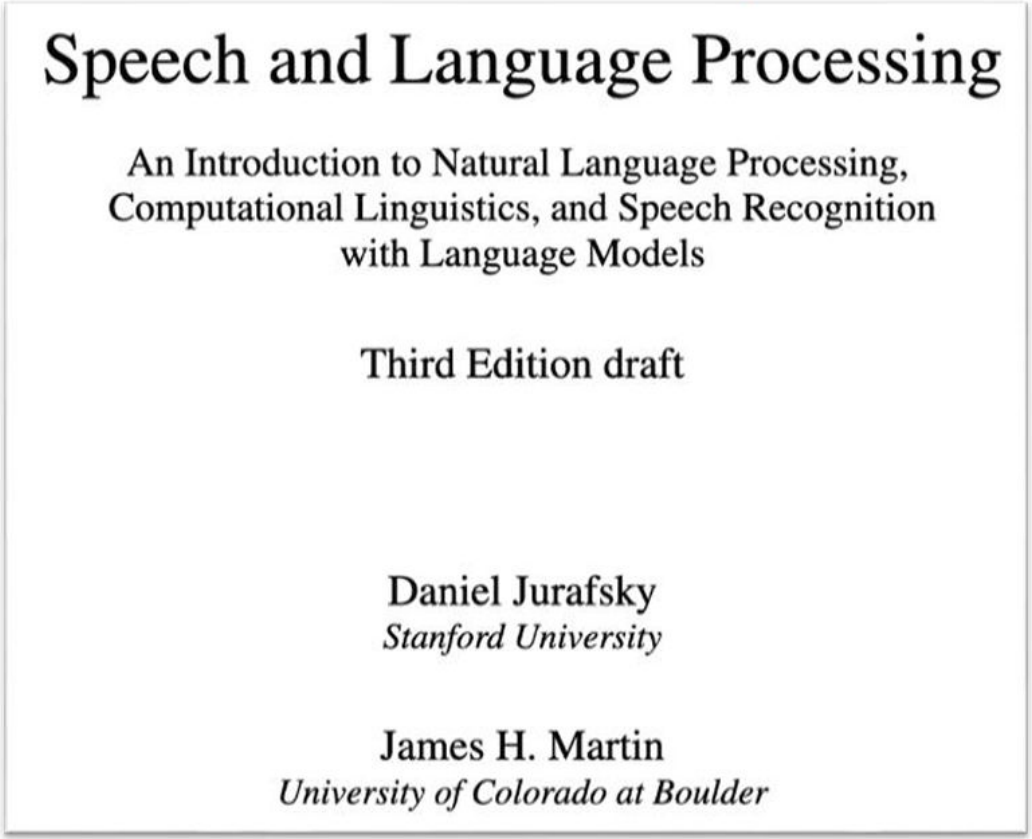


https://www.researchgate.net/figure/A-timeline-of-LLMs-and-LVMs-up-to-

Major LLMs in 2025: **Reasoning Models**



Jurafsky & Martin revised their book (3rd Edition)



Speech and Language Processing

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models

Third Edition draft

Daniel Jurafsky
Stanford University

James H. Martin
University of Colorado at Boulder

Volume I: Large Language Models

Other Resources

1. Jacob Eisenstein. Natural Language Processing
2. Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing
3. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning
4. Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Natural Language Processing with Transformers
5. Prof. Pushpak Bhattacharyya Lectures on NLP

Thank you.