Introduction to Natural Language Processing (NLP)

From Symbolic Rules to Deep Neural Representations

Paul MINCHELLA
paul.minchella@lyon.unicancer.fr

What is NLP?

Natural Language Processing (NLP) is a field at the intersection of **computer science**, **linguistics**, and **statistics**. Its goal is to enable machines to **understand**, **generate**, and **interact** with human language.

Applications

- Machine Translation (Google Translate, DeepL)
- Sentiment Analysis and Opinion Mining
- Chatbots and Conversational Agents (ChatGPT, Claude, Gemini)
- Information Extraction, Named Entity Recognition (NER)
- Speech-to-Text and Text-to-Speech

Core Challenge

Human language is **ambiguous**, **contextual**, and **non-linear**. To model it, we must move from discrete symbols (words) to **continuous representations** in vector spaces. These embeddings make it possible to perform meaningful *mathematical operations on words*, e.g.:

$$\text{King} - \text{Man} + \text{Woman} \approx \text{Queen}.$$

Era	**Representation Type**	**Goal**
Symbolic (1950–1990)	Grammars, trees, logic	Capture structure
Statistical (1990–2010)	Probabilities, frequencies	Capture local dependencies
Neural (2010–Nowadays)	Continuous vectors (embeddings)	Capture meaning and context

The Symbolic Era (1950–1990)

- Language modeled through **rules**, **grammars**, and **logic**.
- Example: Chomsky's *Context-Free Grammars* – formal systems to generate syntactically valid sentences.
- NLP systems like **ELIZA** Mikolov et al., 2013 or **SHRDLU** Winograd, 1972 relied on handcrafted rules.

Limitation

Rule-based systems failed to scale – they lacked robustness and could not generalize beyond their handcrafted logic.

The Statistical Era (1990–2010)

- Data-driven methods replace rigid rules.
- Probabilistic models (e.g., n-grams, Hidden Markov Models) estimate

$$\mathbb{P}(w_t \mid w_{t-n+1}, \dots, w_{t-1})$$

  to capture local dependencies between words Shannon, 1948; Chen and Goodman, 1999.

  $\mathbb{P}(\text{sentence}) = \mathbb{P}(w_1)\mathbb{P}(w_2 \mid w_1)\mathbb{P}(w_3 \mid w_1, w_2) \dots$
  $\mathbb{P}(w_t \mid w_{t-1})$ (bigram model),
  $\mathbb{P}(w_t \mid w_{t-2}, w_{t-1})$ (trigram model)
- Key innovation: using large corpora to learn frequencies and co-occurrence patterns.

Limitation

Statistical models capture surface patterns but **ignore meaning and context**. They cannot distinguish between semantically related words or infer deeper linguistic relationships.

Era	Representation	Core Idea	Main Limitation
Symbolic (1950–1990)	Logical rules	Handcrafted syntax and semantics	No generalization; rules do not scale to real-world variability.
Statistical (1990–2010)	Probabilistic counts	Learning from data frequencies; use of $\mathbb{P}(w_t \mid w_{t-n+1}, \ldots, w_{t-1})$ to capture local dependencies	No semantics; fails to represent meaning or context.
Neural (2010–Nowadays)	Continuous embeddings	Learning distributed meaning via differentiable representations and optimization	Data- and computation-intensive; interpretability remains limited.

Core Idea

The goal of **Maximum Likelihood Estimation (MLE)** is to find the parameter values that make the observed data the most **probable** under a chosen model.

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \ \mathbb{P}_{\theta}(Y_1, \ldots, Y_n)$$

Among all models, we select the one that would most likely have produced our data.

Example – Normal Model

Suppose that $Y_1, \ldots, Y_n \sim \mathcal{N}(\mu, \sigma^2)$, but μ and σ^2 are unknown. MLE chooses $(\hat{\mu}, \hat{\sigma})$ that maximize the joint probability:

$$L(\mu, \sigma) = \prod_{i=1}^{n} f(y_i; \mu, \sigma).$$

Thus, MLE gives the most plausible parameters for the data we observed.

Key Intuition

MLE inverts the usual reasoning: we start from the **data** and infer which model is the most plausible to have generated it.

Likelihood and Log-Likelihood

For an i.i.d. sample $Y_1, \ldots, Y_n \sim f(y;\theta)$:

$$L(\theta) = \prod_{i=1}^{n} f(y_i;\theta), \quad \ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(y_i;\theta).$$

MLE:

$$\hat{\theta}_{\mathrm{MLE}} = \arg\max_{\theta} \ell(\theta)$$

Why Use the Log-Likelihood?

- Converts products into sums $\rightarrow$ easier to manipulate;
- Logarithm preserves the maximizer;
- More stable numerically.

Optimality Condition

At the maximum:

$$\nabla_\theta \ell(\hat{\theta}) = 0, \quad \nabla_\theta^2 \ell(\hat{\theta}) \text{ is negative definite.}$$

Fisher Information Matrix

$$\mathcal{I}(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log f(Y;\theta)\right)\left(\frac{\partial}{\partial\theta}\log f(Y;\theta)\right)^{\top}\right]$$

Measures how sensitive the likelihood is to small changes in θ. A sharp curvature means highly informative data; a flat curvature means uncertainty.

Asymptotic Properties

- **Consistency:** $\hat{\theta}_{\text{MLE}} \to \theta^{\star}$
- **Normality:** $\sqrt{n}(\hat{\theta} - \theta^{\star}) \sim \mathcal{N}(0, \mathcal{I}(\theta^{\star})^{-1})$
- **Efficiency:** reaches Cramér–Rao lower bound.

Connection with NLP

Language models maximize:

$$L(\theta) = \prod_t \mathbb{P}_\theta(w_t \mid w_{<t}),$$

which leads to the **cross-entropy loss** in neural NLP. Models like Word2Vec, GloVe, and BERT are all practical MLEs – their goal is to learn parameters θ that make observed sentences the most likely.

Definition

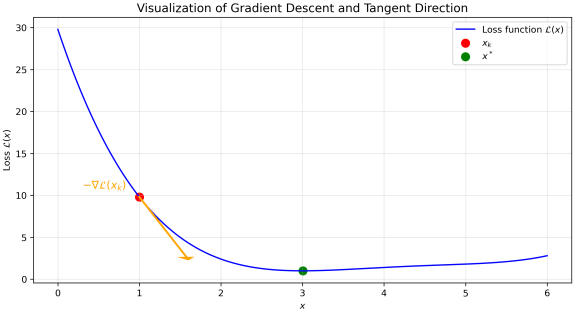
Let $\theta \in \mathbb{R}^d$ be the vector of parameters and $\mathcal{L}(\theta)$ a differentiable loss. Gradient Descent iteratively updates:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta^{(t)}),$$

where $\eta > 0$ is the **learning rate**. The process continues until convergence, when the loss stops decreasing significantly.

Geometric Intuition

Gradient Descent can be visualized as a *ball rolling down a loss landscape*, always moving in the direction of the steepest descent.



The gradient $-\nabla\ell(x_k)$ (orange arrow) points from the current point x_k (red) toward the minimum x^* (green).

Core Idea

Each iteration adjusts parameters to reduce the "energy" of the system – learning can thus be viewed as an **energy minimization process**.

Why It Is So Useful

Gradient Descent is the backbone of nearly all modern learning algorithms:

- Works for complex, non-linear losses with no closed-form solution;
- Requires only gradients, not the explicit form of the minimum;
- Scales to large datasets via variants such as **SGD** Bottou, 2012, **Adam** Kingma and Ba, 2014, and **RMSProp** Hinton, 2012.

Interpretation

Learning = Energy Minimization. The model progressively decreases its "potential energy" (loss) until reaching equilibrium at optimal parameters. This gives a geometric and physical interpretation of training.

Connection to NLP

All NLP models – from **Word2Vec** to **BERT** – are trained by **minimizing a loss** (e.g., cross-entropy, energy-based objectives) using **gradient descent**. Understanding this process is essential to interpret how models learn semantic structures and contextual embeddings.

Definition

Given scores $z = (z_1, \ldots, z_K)$, the **softmax** maps them to a probability distribution:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.$$

It ensures: $p_i > 0$, $\sum_i p_i = 1$, and monotonicity $(z_i > z_j \Rightarrow p_i > p_j)$.

Intuition

Softmax acts as a smooth $\arg\max$ – higher scores yield higher probabilities, but transitions remain continuous and differentiable.

Interpretation

Softmax is a bridge between:

- **Scores** (model outputs)
- **Probabilities** (interpretable predictions)

and thus the core normalization used in NLP models.

Statistical View

Softmax generalizes logistic regression to multiple classes:

$$\mathbb{P}(y = k \mid x) = \frac{e^{w_k^\top x}}{\sum_j e^{w_j^\top x}}.$$

It yields the **maximum-entropy** distribution consistent with observed data.

Physical View (Boltzmann Distribution)

$$\mathbb{P}(i) = \frac{e^{-\beta E_i}}{\sum_j e^{-\beta E_j}},$$

where low energy $\Rightarrow$ high probability. Temperature $\tau = 1/\beta$ controls how peaked the distribution is.

Geometric View

Softmax maps $\mathbb{R}^K$ to the probability simplex $\Delta^{K-1} = \{p_i \geq 0, \sum_i p_i = 1\}$, preserving ordering and scale invariance.

Computation and Stability

For numerical stability, we use the **log-sum-exp trick**:

$$\log \sum_i e^{z_i} = z_{\max} + \log \sum_i e^{z_i - z_{\max}}.$$

Derivative:

$$\frac{\partial p_i}{\partial z_j} = p_i(\delta_{ij} - p_j).$$

Connection to Optimization

Combined with the cross-entropy loss:

$$\mathcal{L} = -\sum_i y_i \log p_i,$$

it provides a smooth, differentiable objective for neural models.

Why It Matters in NLP

Softmax ensures that:

- output scores become interpretable probabilities,
- learning remains differentiable,
- training objectives (MLE, cross-entropy) stay mathematically consistent.

Definition

An **Energy-Based Model** assigns an energy $E_\theta(x, y)$ to each pair (x, y):

$$\mathbb{P}_\theta(y \mid x) = \frac{e^{-E_\theta(x,y)}}{Z_\theta(x)}, \quad Z_\theta(x) = \sum_{y'} e^{-E_\theta(x,y')}.$$

Low energy $=$ high compatibility.

Connection with Softmax

If $E_\theta(x, y) = -f_\theta(x, y)$:

$$\mathbb{P}_\theta(y \mid x) = \frac{\exp(f_\theta(x, y))}{\sum_{y'} \exp(f_\theta(x, y'))} = \text{softmax}(f_\theta(x, y)).$$

Thus, every softmax classifier is a normalized energy model.

References. LeCun et al. (2006), Mnih and Hinton (2008) introduce scalable formulations of hierarchical and energy-based language models.

Language Modeling as Energy Minimization

Predicting the next word w_t given context c_t:

$$E(c_t, w_t) = -f_\theta(c_t, w_t), \quad \mathbb{P}(w_t \mid c_t) = \frac{e^{f_\theta(c_t, w_t)}}{\sum_{W'} e^{f_\theta(c_t, W')}}.$$

Training reduces the energy of observed pairs.

Applications

- **Word2Vec (Mikolov, 2013)** – contrastive energy learning via negative sampling.
- **Hierarchical LMs (Mnih & Hinton, 2008)** – scalable softmax approximation.
- **Transformers** – attention weights as normalized energy distributions.

Key Message

Softmax and EBMs form the mathematical core of NLP: learning = minimizing energy, prediction = choosing low-energy configurations.

- ✓ **Softmax:** turns scores into probabilities (normalization layer).
- ✓ **Energy-Based Models:** define compatibility via energy functions.
- ✓ **Training:** reduce the energy of real examples (MLE, contrastive learning).
- ✓ **Inference:** select configurations with minimal energy.
- ✓ **In NLP:** used in Word2Vec, BERT, GPT, and attention mechanisms.

Main core for Modern NLP

Mathematical structure underlying modern NLP systems: the **Artificial Neural Network (ANN)**.

Key Idea

By composing layers of simple transformations, neural networks can *learn to represent complex relationships in data*, including hierarchical linguistic patterns.

Universal Approximation Theorem Cybenko, 1989; Hornik, 1991

A feed-forward neural network with a single hidden layer containing a finite number of neurons can approximate **any continuous function** on compact subsets of $\mathbb{R}^n$, given a suitable activation function.

Interpretation

This result formalizes the expressive power of neural networks: they are universal function approximators.

In NLP

It justifies using neural architectures as general-purpose models for:

- word and sentence embeddings,
- sequence encoding and contextualization,
- large language models (LLMs).

Mathematical Definition

Given inputs $x = (x_1, \ldots, x_d)$:

$$h = \sigma(w^\top x + b),$$

where w are weights, b a bias, and σ a nonlinear activation.

Activation Functions

$$\sigma(x) = \tanh(x), \quad \sigma(x) = \frac{1}{1 + e^{-x}} \text{ (sigmoid)}, \quad \sigma(x) = \max(0, x) \text{ (ReLU)}.$$

Role of Nonlinearity

Without activation functions, stacked layers remain linear – nonlinearity is what enables expressive, hierarchical representations.

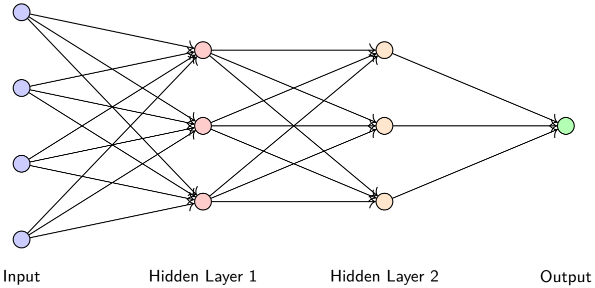
Definition

A **feedforward neural network** (or multilayer perceptron) stacks layers:

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}), \quad \hat{y} = W^{(L)} h^{(L-1)} + b^{(L)}.$$

Each layer learns intermediate representations.

- Early layers capture lexical or syntactic patterns.
- Deeper layers encode meaning and context.



Stacking Nonlinear Layers

Each layer applies a nonlinear transformation:

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}).$$

Composing many layers allows the network to model complex feature interactions.

Representation Hierarchy in NLP

- **Lower layers:** lexical and syntactic cues.
- **Higher layers:** semantic and contextual meaning.

Intuition

This mirrors human language processing: from characters $\rightarrow$ words $\rightarrow$ phrases $\rightarrow$ discourse.

Goal

Minimize a loss function $\mathcal{L}(\theta)$ using gradient descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}.$$

Challenge

Deep networks involve nested compositions:

$$\mathcal{L} = \ell(W^{(L)} h^{(L-1)} + b^{(L)}, y),$$

making manual differentiation infeasible.

Solution: Backpropagation

Backprop systematically applies the chain rule to compute gradients efficiently, layer by layer, from output to input.

1. Forward Pass

Compute activations layer by layer:

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}).$$

2. Backward Pass

Propagate errors backward:

$$\delta^{(l)} = ((W^{(l+1)})^\top \delta^{(l+1)}) \odot \sigma'(a^{(l)}), \quad \nabla_{W^{(l)}} \mathcal{L} = \delta^{(l)} (h^{(l-1)})^\top.$$

Efficiency

All gradients are computed with roughly twice the cost of one forward pass.

Core Benefits

- Enables **end-to-end learning**.
- Scales to millions of parameters (automatic differentiation).
- Provides a unified optimization framework across architectures.

In NLP Practice

Backprop drives the learning of:

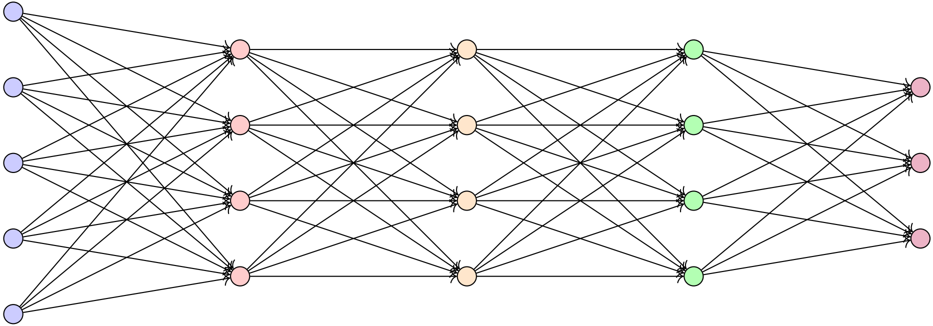
- Word embeddings (Word2Vec, GloVe),
- Sequence models (LSTM, GRU),
- Contextual models (BERT, GPT).

Summary

Forward pass + backward pass = representation learning. This is the mathematical engine behind every modern NLP system.

Word Embedding Layer

The final layer is a mapping that outputs the word's embedding vector, which captures the word's meaning in a high-dimensional space.



Definition of a Token

A **token** is the smallest atomic unit of text processed by an NLP model. Depending on the task, it may represent:

- a **word**: “hospital”, “patient”;
- a **subword or morpheme**: “play” and “-ing” in “playing”;
- or a **character or punctuation mark**.

Tokenization converts raw text into a discrete sequence:

$$\text{"The patient recovered."} \longrightarrow [\text{"The"}, \text{"patient"}, \text{"recovered"}, \text{"."}]$$

The Vocabulary Set

All distinct tokens form the finite set:

$$\mathcal{V} = \{v_1, v_2, \dots, v_N\}, \quad N = |\mathcal{V}|.$$

Typical vocabularies include:

- `[UNK]` – unknown tokens,
- `[PAD]` – padding for equal-length sequences.

From Words to Vectors

Each token $v_i \in \mathcal{V}$ is associated with a dense vector $w_i \in \mathbb{R}^d$. All embeddings form the learnable matrix:

$$W = [w_1 \ w_2 \ \ldots \ w_N]^\top \in \mathbb{R}^{N \times d}.$$

Softmax-Based Learning Objective

During training, embeddings are optimized through a probabilistic objective:

$$\mathbb{P}(w_t \mid \text{context}) = \frac{\exp(v_{w_t}^\top h_t)}{\sum_{w \in \mathcal{V}} \exp(v_w^\top h_t)},$$

where h_t is the contextual hidden state. The model minimizes:

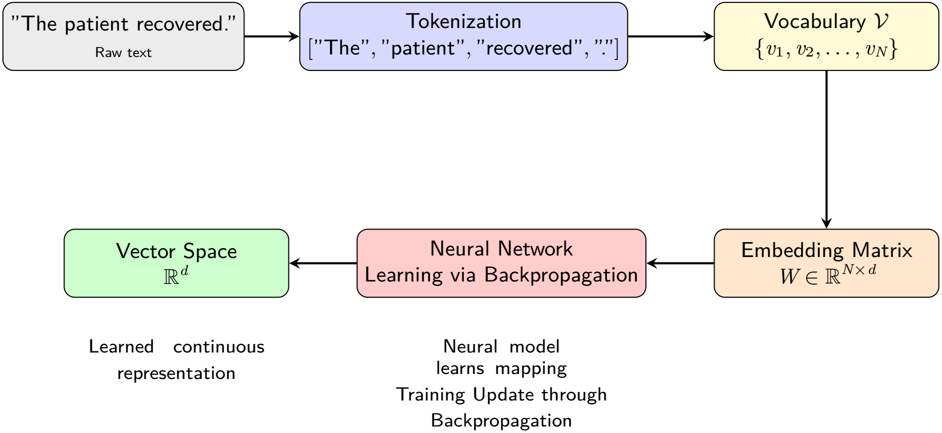
$$\mathcal{L} = -\sum_t \log \mathbb{P}(w_t \mid \text{context}).$$

Result: Embeddings in a Vector Space

Optimization via gradient descent gradually organizes W so that:

- words with similar contexts have nearby vectors,
- meaning and analogy emerge geometrically in $\mathbb{R}^d$.

Language thus becomes **geometry**: continuous, measurable, and differentiable.


"The patient recovered."
Raw text
Tokenization
["The", "patient", "recovered", "."]
Vocabulary $\mathcal{V}$
$\{v_1, v_2, \ldots, v_N\}$
Embedding Matrix
$W \in \mathbb{R}^{N \times d}$
Neural Network
Learning via Backpropagation
Vector Space
$\mathbb{R}^d$
Learned continuous representation
Neural model learns mapping
Training Update through Backpropagation

Observation

Embedding spaces exhibit **linear regularities** that correspond to semantic and syntactic relationships:

$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}.$$

Interpretation

- $v_{\text{king}} - v_{\text{man}}$ encodes the concept of "royalty".
- $v_{\text{woman}} - v_{\text{man}}$ captures "gender".
- Similar analogies (e.g., Paris $-$ France $+$ Italy $\approx$ Rome) encode geographic relations.

Geometric Meaning

The embedding space decomposes into **subspaces** reflecting conceptual axes (gender, tense, number, royalty, profession, etc.). The geometry of the space mirrors the latent structure of human semantics.

Intuitive Picture

Each word embedding $v_w \in \mathbb{R}^d$ acts as a coordinate of meaning in a high-dimensional space.

Semantic Clustering

- Animals: *dog*, *cat*, *lion* cluster together.
- Professions: *doctor*, *nurse*, *teacher*.
- Emotions: *love*, *hate*, *joy*.

The topology of this space – distances and angles – encodes how meanings relate or diverge.

Key Idea

Embeddings transform language into **geometry**: meaningful relations become measurable through distances and directions.

Euclidean Distance

Measures absolute distance between embeddings:

$$d_{\mathsf{Euc}}(v_i, v_j) = \|v_i - v_j\|_2 = \sqrt{\sum_{k=1}^{d} (v_{i,k} - v_{j,k})^2}.$$

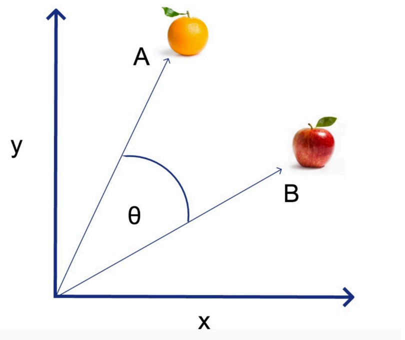
Sensitive to vector norms – focuses on spatial proximity.

Cosine Similarity

Focuses on the **angle** between vectors:

$$\text{cosine\_sim}(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \, \|v_j\|} = \cos(\theta_{ij}).$$

- $\cos(\theta_{ij}) = 1$: identical meaning.
- $\cos(\theta_{ij}) = 0$: unrelated.
- $\cos(\theta_{ij}) = -1$: opposite meaning.



Why Cosine Similarity?

In high dimensions, it ignores magnitude and captures only **semantic direction**. It directly aligns with objectives used in models such as Word2Vec or BERT.

Regularities in the Embedding Space

Certain directions correspond to semantic relations:

$$v_{\text{king}} - v_{\text{man}} \approx v_{\text{queen}} - v_{\text{woman}}.$$

Subspace Interpretation

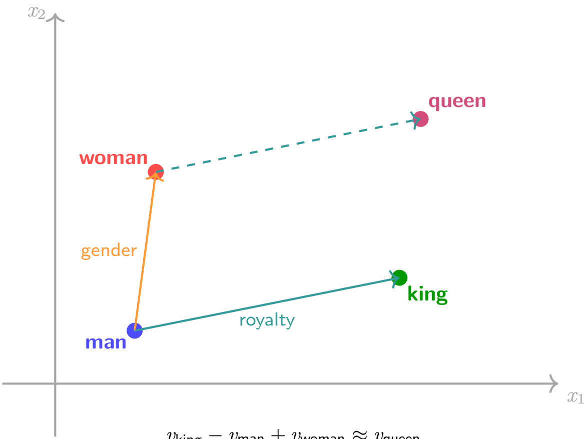
If U_{royalty} is the "royalty" subspace:

$$v_{\text{king}} - v_{\text{man}} \in U_{\text{royalty}}, \quad v_{\text{queen}} - v_{\text{woman}} \in U_{\text{royalty}}.$$

These vectors are approximately parallel – encoding the same conceptual relation.

Summary

- Words become algebraic objects in $\mathbb{R}^d$.
- Semantic relations correspond to linear transformations.
- Embedding geometry reveals meaning through direction and distance.



$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}$$

Core Idea

The **Word2Vec** model introduced by Mikolov et al., 2013 is a simple yet powerful way to learn word meaning from co-occurrence patterns in text. It relies on the **distributional hypothesis** (Firth, 1957):

"You shall know a word by the company it keeps."

Motivation

Words that appear in similar contexts tend to have similar meanings. For example:

$$\text{doctor} \leftrightarrow \text{hospital} \quad \text{(context: patient, nurse, medicine)}$$

Word2Vec learns to predict context words from a target word (or vice versa) and captures these semantic regularities as vector similarities.

Key Paradigm

No manual labels are needed – **the corpus supervises itself**. Each word provides training signals for its neighbors: this is **self-supervised learning**.

Distributional Hypothesis

At the core of Word2Vec lies the hypothesis that *semantic similarity arises from contextual similarity*. Words that appear in similar linguistic contexts tend to have similar meanings.

Mathematical Objective

Given a sequence $(w_1, w_2, \ldots, w_T)$ from a corpus, learn a mapping

$$f : \mathcal{V} \to \mathbb{R}^p,$$

that associates each word $w \in \mathcal{V}$ with a dense vector $f(w)$ capturing its syntactic and semantic regularities.

Goal

Find embeddings such that:

$$\text{similar meaning} \iff \text{similar context statistics.}$$

Training Principle

At each position t, the model observes a target word w_t and predicts each context word w_{t+j}, for $j \in \{-C, \dots, -1, 1, \dots, C\}$.

$$\mathbb{P}(w_{t+j} \mid w_t)$$

is the probability of observing w_{t+j} given the central word w_t.

Network Structure

$$x_w \xrightarrow{W} h \xrightarrow{W'} z$$

- $W \in \mathbb{R}^{p \times N}$: input embedding matrix;
- $W' \in \mathbb{R}^{N \times p}$: output embedding matrix;
- $h = W x_w$: hidden (embedding) representation;
- $z = W' h$: unnormalized output scores.

Softmax Layer

The output vector z is normalized via:

$$\mathbb{P}(w_i \mid w_t) = \frac{\exp(u_{w_i}^\top v_{w_t})}{\sum_{k=1}^{N} \exp(u_{w_k}^\top v_{w_t})},$$

where v_{w_t} and u_{w_i} are the input and output embeddings respectively.

Loss Function

For each word pair (w_t, w_{t+j}):

$$\mathcal{L}(w_t, w_{t+j}) = -\log \mathbb{P}(w_{t+j} \mid w_t).$$

Global objective:

$$\mathcal{L}_{\text{total}} = -\sum_{t=1}^{T} \sum_{\substack{j=-C \\ j\neq 0}}^{C} \log \mathbb{P}(w_{t+j} \mid w_t).$$

Learning Mechanism. By minimizing $\mathcal{L}_{\text{total}}$, the model adjusts embeddings v_{w_t} such that co-occurring words have higher dot products.

Challenge

Softmax normalization is costly: $\mathcal{O}(N)$ per update. Two main solutions exist.

(i) **Hierarchical Softmax:** Factorizes $\mathbb{P}(w_i \mid w_t)$ via a binary tree, reducing complexity to $\mathcal{O}(\log N)$.

(ii) **Negative Sampling:** Replaces the softmax by a logistic objective distinguishing true and noise pairs:

$$\mathcal{L}_{\mathsf{NS}}(w_t, w_{t+j}) = -\log \sigma(u_{w_{t+j}}^\top v_{w_t}) - \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n}[\log \sigma(-u_{w_k}^\top v_{w_t})],$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$.

Sampling Strategy

P_n is a **noise distribution** over words used to draw negative examples.

$$P_n(w_i) = \frac{\mathrm{f}(w_i)^{3/4}}{\sum_{j=1}^{N} \mathrm{f}(w_j)^{3/4}}.$$

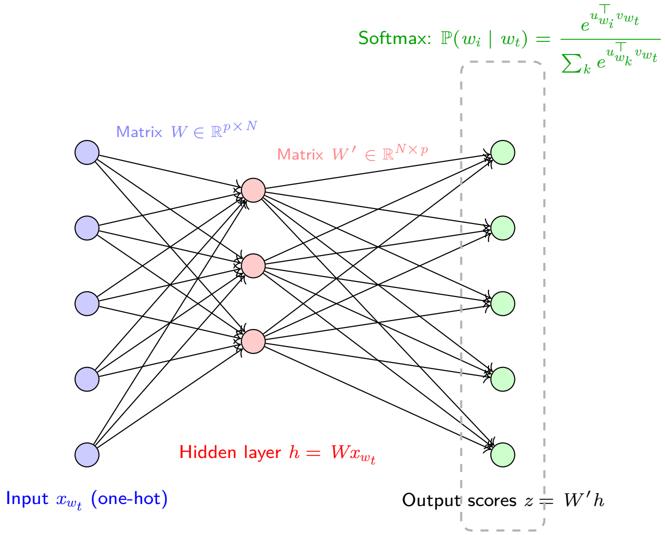
Empirical Rule

Raising frequencies to the $3/4$ power (Goldberg and Levy, 2014):

- down-weights extremely frequent words (e.g., *the*, *of*);
- ensures frequent words still appear often enough to shape meaningful contrasts.

Efficiency

Negative Sampling scales linearly with K rather than N, allowing efficient training on large vocabularies.


Softmax: $\mathbb{P}(w_i \mid w_t) = \frac{e^{u_{w_i}^\top v_{w_t}}}{\sum_k e^{u_{w_k}^\top v_{w_t}}}$
Matrix $W \in \mathbb{R}^{p \times N}$
Matrix $W' \in \mathbb{R}^{N \times p}$
Hidden layer $h = W x_{w_t}$
Input x_{w_t} (one-hot)
Output scores $z = W' h$

Summary

Core Idea

The **Continuous Bag of Words (CBOW)** model is the inverse of Skip-Gram: it predicts the central target word w_t from its surrounding context words.

$$\mathbb{P}(w_t \mid w_{t-C}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+C})$$

Distributional Intuition

A word's meaning can be inferred from the company it keeps. CBOW captures this intuition by averaging the embeddings of neighboring words to predict the target.

$$\text{Context: } (\text{doctor}, \text{nurse}, \text{patient}) \Rightarrow \text{Target: hospital.}$$

Conceptual Symmetry

CBOW and Skip-Gram share the same embedding matrices and probabilistic formulation — they differ only in the prediction direction.

Architecture Overview

Each context word w_{t+j} is represented by a one-hot vector $x_{t+j} \in \{0,1\}^N$.

1. Map each context word to its embedding:

$$h_{t+j} = W x_{t+j}, \quad W \in \mathbb{R}^{p \times N}.$$

2. Compute the mean context representation:

$$h = \frac{1}{2C} \sum_{\substack{j=-C \\ j \neq 0}}^{C} h_{t+j} = \frac{1}{2C} \sum_{\substack{j=-C \\ j \neq 0}}^{C} W x_{t+j}.$$

3. Compute output scores:

$$z = W' h, \quad W' \in \mathbb{R}^{N \times p}.$$

Interpretation

The vector h acts as a dense semantic representation of the entire context window.

Conditional Probability

The model defines:

$$\mathbb{P}(w_t \mid w_{t-C}, \ldots, w_{t+C}) = \frac{\exp(u_{w_t}^\top h)}{\sum_{i=1}^{N} \exp(u_i^\top h)},$$

where u_i is the i-th row of W' (the output embedding).

Training Objective

For each position t:

$$\mathcal{L}(w_{t-C}, \ldots, w_{t+C}) = -\log \mathbb{P}(w_t \mid w_{t-C}, \ldots, w_{t+C}).$$

Global loss:

$$\mathcal{L}_{\text{total}} = -\sum_{t=1}^{T} \log \mathbb{P}(w_t \mid w_{t-C}, \ldots, w_{t+C}).$$

Learning Goal

Learn embeddings such that the averaged context representation h aligns closely with the target embedding u_{w_t}.

Prediction Direction

→ **Skip-Gram:** predicts context words given a target word.

→ **CBOW:** predicts the target word given its context.

Empirical Observations

- CBOW is faster to train — the averaging operation stabilizes learning.
- Skip-Gram handles rare words better — it processes each pair individually.

Computational Aspects

Both models face the same softmax bottleneck and therefore use:

1. **Hierarchical Softmax:** $\mathcal{O}(\log N)$.
2. **Negative Sampling:** logistic approximation distinguishing true and random pairs.

Motivation: Combining Local and Global Statistics

Unlike Word2Vec, which learns from local context windows, **GloVe** (Pennington et al., 2014) integrates **global corpus statistics**. It models meaning through the **ratios of co-occurrence probabilities** between words:

$$P_{ij} = \frac{X_{ij}}{X_i}, \qquad \frac{P_{ik}}{P_{jk}} \approx \text{semantic relation between } i, j.$$

Mathematical Objective

Let X_{ij} be the number of times word j appears in the context of word i. GloVe seeks embeddings $w_i, \tilde{w}_j$ such that:

$$w_i^\top \tilde{w}_j + b_i + \tilde{b}_j \approx \log X_{ij}.$$

The cost function is a weighted least-squares form:

$$\mathcal{L} = \sum_{i,j} f(X_{ij}) \left(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2.$$

Interpretation

- ✓ GloVe performs a **log-linear factorization** of the co-occurrence matrix.
- ✓ Combines Word2Vec's predictive modeling with statistical efficiency.
- ✓ Encodes global semantic structure – directions such as "royalty" or "gender" appear linearly:

$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}.$$

Motivation: Handling Morphology and Rare Words

FastText (Bojanowski et al., 2017) improves over Word2Vec and GloVe by incorporating **subword information**.
Instead of treating each word as atomic, FastText represents it as the sum of its character n-grams.
Each word w is represented as:

$$v_w = \sum_{g \in G_w} z_g,$$

where G_w is the set of character n-grams (e.g. for "apple" with $n = 3$: $\langle$ap, app, ppl, ple, le$\rangle$), and z_g their embeddings. The same **Skip-Gram with Negative Sampling** loss is used as in Word2Vec.

Advantages

- ✓ Learns **morphological regularities** (prefixes, suffixes, roots).
- ✓ Handles **rare or unseen words** through shared subwords.
- ✓ Works especially well for **morphologically rich languages**.
- ✓ FastText extends Word2Vec to the **subword level**, allowing models to generalize across inflected or unseen forms – a key step toward robust multilingual embeddings.

Motivation: Beyond Static Embeddings

Previous models (Word2Vec, GloVe, FastText) assign **one fixed vector per word**, regardless of context.
However, word meaning is **context-dependent**: *bank* (finance) $\neq$ *bank* (river).
ELMo (*Embeddings from Language Models*, Peters et al., 2018) introduced **contextual embeddings**, where each token's representation depends on the entire sentence.

Bidirectional LSTM Language Model

ELMo trains a forward and backward language model:

$$\mathbb{P}(w_1, \ldots, w_T) = \prod_{t=1}^{T} \mathbb{P}(w_t \mid w_{<t}) + \prod_{t=1}^{T} \mathbb{P}(w_t \mid w_{>t}).$$

Each layer of the bidirectional LSTM produces hidden states $h_{t,l}$ at different abstraction levels.

$$\text{ELMo}_t = \gamma \sum_{l=0}^{L} s_l h_{t,l},$$

where s_l are learned scalar weights and γ is a scaling factor.

Impact

- ✓ Introduced **contextual embeddings**: dynamic representations that change with sentence context.
- ✓ Enabled significant gains across NLP benchmarks (NER, QA, sentiment analysis).
- ✓ Marked the transition from *static geometry* to *contextualized language understanding*.

Motivation and Core Idea

BERT (*Bidirectional Encoder Representations from Transformers*, Devlin et al., 2019) integrates:

- → **Deep context modeling** (like ELMo),
- → **Bidirectional attention** (unlike unidirectional LSTMs),
- → **Transformer encoders** (Vaswani et al., 2017).

It learns rich, general-purpose language representations through large-scale pretraining.

Architecture and Training Objectives

Architecture: Multi-layer Transformer encoder with self-attention:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

Pretraining Tasks:

- ☐ **Masked Language Modeling (MLM):** Predict masked words from context. $\mathcal{L}_{\mathrm{MLM}} = -\sum_{t \in M} \log \mathbb{P}(w_t \mid w_{\setminus t})$
- ☐ **Next Sentence Prediction (NSP):** Predict whether sentence B follows sentence A.

Impact and Legacy

- ✓ Unified and extended Word2Vec, ELMo, and attention-based ideas.
- ✓ Provided **universal pretrained representations** transferable to any downstream task.
- ✓ Inspired successors: RoBERTa, ALBERT, DistilBERT, GPT series.

BERT established the modern paradigm of **pretrain** $\rightarrow$ **fine-tune**.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bottou, L. (2012). Stochastic gradient descent tricks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis*, pages 1–32. Basil Blackwell, Oxford. Reprinted in Firth, J. R. (1957). *Papers in Linguistics 1934–1951*. Oxford University Press.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.

Hinton, G. (2012). Neural networks for machine learning, lecture 6a: Overview of mini-batch gradient descent. Coursera Lecture, University of Toronto.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. J. (2006). A tutorial on energy-based learning. In Bakker, J. and Schölkopf, B., editors, *Predicting Structured Data*. MIT Press.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1301.3781.

Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 21, pages 1081–1088.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of NeurIPS*, pages 5998–6008.

Winograd, T. (1972). *Understanding Natural Language*. Academic Press, New York.