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

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

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

  $\mathbb{P}(\text{sentence}) = \mathbb{P}(w_1)\mathbb{P}(w_2 \mid w_1)\mathbb{P}(w_3 \mid w_1, w_2)\ldots$
  $\mathbb{P}(w_t \mid w_{t-1})$ (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.

Statement (Taylor–Young Theorem).

Let $f\colon \mathbb{R} \to \mathbb{R}$ be of class $\mathcal{C}^n$ in a neighborhood of a point $a \in \mathbb{R}^d$.

$$f(x) \underset{x \to a}{=} f(a) + \sum_{k=1}^{n} \frac{f^{(k)}(a)}{k!}\,(x-a)^k + o\big((x-a)^n\big).$$

Statement (Order 2 Taylor–Young formula)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be of class $\mathcal{C}^2$ in a neighborhood of $a \in \mathbb{R}^d$. For x close to a, set $h = x - a$.

$$f(a+h) = f(a) + \nabla f(a) \cdot h + \frac{1}{2}\, h^\top H_f(a)\, h + o(\|h\|^2) \quad (h \to 0).$$

Notations:

- $\nabla f(a)$: gradient vector of first partial derivatives, $\nabla f(a) = \big(f_{x_1}(a), \ldots, f_{x_d}(a)\big)$.
- $H_f(a)$: Hessian matrix, $H_f(a) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(a)\right)_{1 \le i,j \le d}$.

Interpretation: The linear term $\nabla f(a) \cdot h$ gives the tangent plane, and the quadratic term $\frac{1}{2} h^\top H_f(a) h$ describes the local curvature of f near a.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be of class $\mathcal{C}^1$, and denote

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right).$$

Orthogonality to Level Curves

Fix $k \in \mathbb{R}$ and define the level set

$$L_k = \{(x, y) : f(x, y) = k\}.$$

If $p \in L_k$ and $\nabla f(p) \neq 0$, then $\nabla f(p)$ is **orthogonal to the tangent vector** to L_k at p (i.e. $\nabla f(p)$ is normal to the curve L_k).

Idea of the proof. Let $\gamma : I \to \mathbb{R}^2$ be a parametrization of L_k with $\gamma(t_0) = p$. Since $f(\gamma(t)) \equiv k$,

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}(f \circ \gamma)(t_0) = \nabla f(\gamma(t_0)) \cdot \gamma'(t_0) = \nabla f(p) \cdot \gamma'(t_0).$$

Thus, $\nabla f(p)$ is orthogonal to all tangent vectors of L_k at p.

The Gradient Points Toward Increasing Values

For any unit vector $u \in \mathbb{R}^2$, the directional derivative of f at p is

$$D_u f(p) = \nabla f(p) \cdot u.$$

Hence,

$$\max_{\|u\|=1} D_u f(p) = \|\nabla f(p)\|, \quad \text{attained for } u = \frac{\nabla f(p)}{\|\nabla f(p)\|}.$$

The steepest decrease occurs for $u = -\nabla f(p)/\|\nabla f(p)\|$.

Key ideas:

1. $D_u f(p) = \nabla f(p) \cdot u$ (definition of directional derivative).
2. By Cauchy–Schwarz: $|\nabla f(p) \cdot u| \leq \|\nabla f(p)\|$.
3. First-order approximation:

$$f(p + tu) = f(p) + t\, D_u f(p) + o(t).$$

If $u = \frac{\nabla f(p)}{\|\nabla f(p)\|}$ and $t > 0$, then $f(p + tu) > f(p)$: the gradient points toward the **increase** of f.

Setting. Let $f : \mathbb{R}^2 \to \mathbb{R}$ be of class $\mathcal{C}^1$ and let

$$u, v : \mathbb{R} \to \mathbb{R} \text{ differentiable.}$$

We define the composition

$$g : \ t \longmapsto f\big(u(t),\ v(t)\big).$$

Formula (chain rule, version 1D $\to$ 2D):

$$\frac{\mathrm{d}g}{\mathrm{d}t}(t) = \frac{\partial f}{\partial x}\big(u(t), v(t)\big)\ u'(t) + \frac{\partial f}{\partial y}\big(u(t), v(t)\big)\ v'(t)$$

Vector form

If $w(t) = (u(t), v(t))$ and $\nabla f = (f_x, f_y)$,

$$g'(t) = w'(t) \cdot \nabla f\big(w(t)\big).$$

Setting. Let $u, v : \mathbb{R}^2 \to \mathbb{R}$ be $\mathcal{C}^1$, and define

$$h(x, y) = f\big(u(x, y),\ v(x, y)\big).$$

Partial derivatives:

$$\frac{\partial h}{\partial x} = f_x(u, v)\, u_x + f_y(u, v)\, v_x,$$

$$\frac{\partial h}{\partial y} = f_x(u, v)\, u_y + f_y(u, v)\, v_y.$$

Denoting

$$W(x, y) = (u, v), \quad J_W = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}, \quad \nabla f = \begin{pmatrix} f_x \\ f_y \end{pmatrix}.$$

Matrix (Jacobian) Form

$$\nabla h(x, y) = J_W(x, y)^\top\, \nabla f\big(W(x, y)\big).$$

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 $Y_1, \dots, Y_n \sim f(y;\theta)$:

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

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:

$$\nabla_\theta \ell(\hat{\theta}) = 0,$$

$H_\ell(\hat{\theta})$ is negative definite (all its eigenvalues are strictly negative).

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

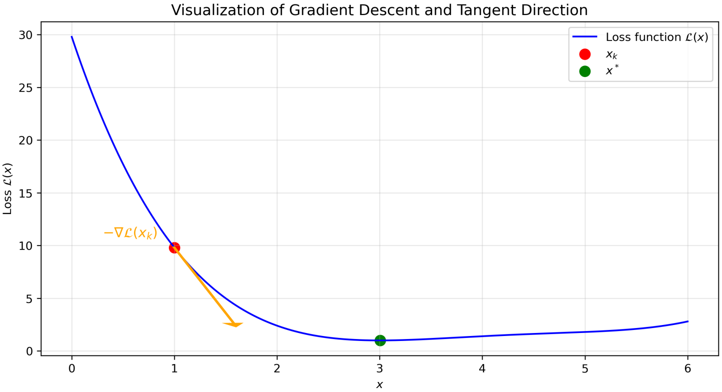
Let $\theta \in \mathbb{R}^d$ be the vector of parameters and $\mathcal{L}(\theta)$ a differentiable loss. Gradient Descent iteratively updates:

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

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 $-\nabla\mathcal{L}(x_k)$ points from the current point x_k toward the minimum x^*.

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 $z = (z_1, \dots, z_K)$, 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 $x = (x_1, \dots, x_d)$:

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

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.

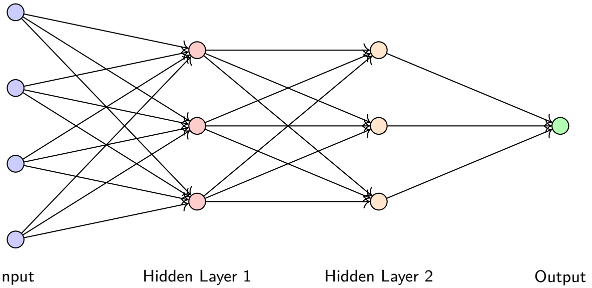
Definition

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.

Core Idea

Word embeddings are **not an end goal**, but a way to represent language numerically. They map discrete symbols (words) into a continuous vector space.

- **From Discrete to Continuous:** Transform one-hot word identifiers into dense vectors $v_w \in \mathbb{R}^p$.

$$\text{similar words} \Rightarrow \text{nearby vectors.}$$

- **Geometric Structure:** Linear relations reflect semantic regularities: $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}$.
- **A Foundational Representation:**
  - → Embeddings form the **input layer** for deeper models (RNNs, Transformers, classifiers).
  - → Capture statistical and semantic patterns from text.
  - → Bridge between **symbolic language** and **mathematical modeling**.

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:

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

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:

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

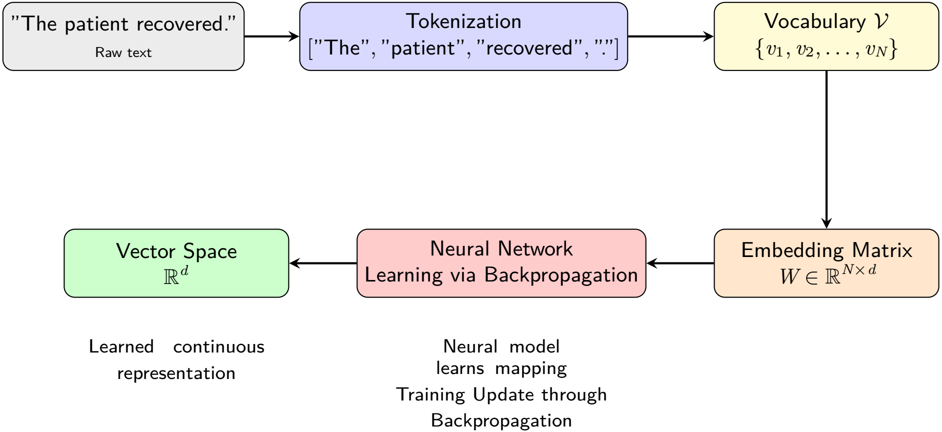
Result: Embeddings in a Vector Space

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.

From Symbols to Geometry



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 ≈ 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:

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

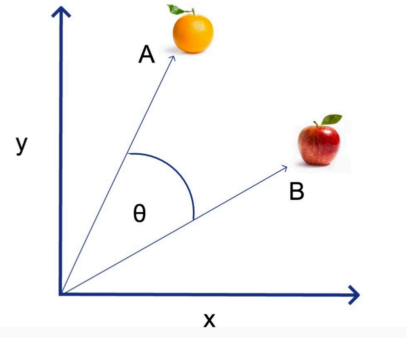
Sensitive to vector norms – focuses on spatial proximity.

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

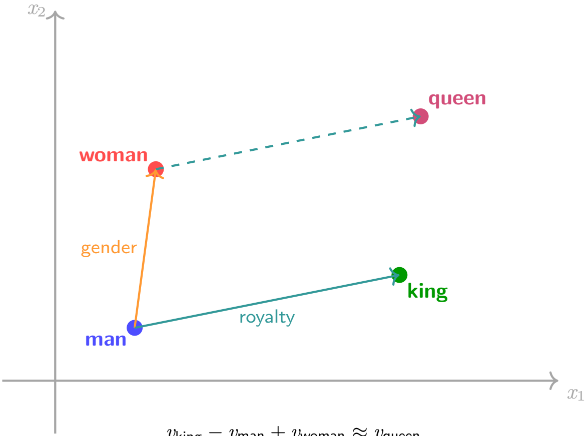
If U_{royalty} is the "royalty" subspace:

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

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.



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

Word2Vec

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:

doctor $\leftrightarrow$ hospital (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:

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

Main Idea

The **Continuous Bag-of-Words (CBOW)** model predicts the **target word** given its surrounding **context words**. It treats the context as a "bag of words", ignoring the order of appearance, and averages their embeddings to infer the central word.

Illustration

Given a sentence:

$$(\text{the, cat, sat, on, the, mat})$$

and a window size $C = 2$, the context for the central word "sat" is:

$$\{\text{the, cat, on, the}\}.$$

CBOW uses these context words to predict "sat".

Architecture

Input: context words $(w_{t-C}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+C})$

Hidden layer: average (or sum) of their embeddings

Output: predicted target word w_t

Forward Pass

Each context word is represented by its one-hot vector $x_i \in \mathbb{R}^{|\mathcal{V}|}$. The embedding matrix $V \in \mathbb{R}^{p \times |\mathcal{V}|}$ maps words to dense embeddings:

$$v_{w_i} = V x_i.$$

The hidden representation (context vector) is the mean embedding:

$$h_t = \frac{1}{2C} \sum_{\substack{-C \le j \le C \\ j \neq 0}} v_{w_{t+j}}.$$

CBOW Forward Flow

$$x_{t-C}, \dots, x_{t-1}, x_{t+1}, \dots, x_{t+C} \xrightarrow[\text{avg}]{V} h_t \xrightarrow{U^\top} z$$

Predict the central word from its surrounding context.

Linear-Linear-Softmax Architecture

The output layer uses another matrix $U \in \mathbb{R}^{p \times |\mathcal{V}|}$ to produce unnormalized scores:

$$z = U^\top h_t, \quad z_i = u_i^\top h_t.$$

Applying the softmax gives a probability distribution over the vocabulary:

$$\mathbb{P}(w_t = i \mid \text{context}) = \frac{\exp(u_i^\top h_t)}{\sum_{k=1}^{|\mathcal{V}|} \exp(u_k^\top h_t)}.$$

Loss Function

The model is trained to maximize the log-likelihood of the correct target word. Equivalently, we minimize the negative log-probability:

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

Why It Works

Words that occur in similar contexts produce similar hidden representations h_t.

$\rightarrow$ The model therefore learns embeddings v_w that reflect **distributional similarity**.

Bag-of-Words Effect

The hidden layer $h_t = \sum v_{w_{t+j}}$ is a continuous analogue of a discrete word-count vector. Each word's embedding contributes proportionally to its local frequency in the context window.

$$h_t \propto \text{ weighted histogram of context words.}$$

Key Insight

No order information is used — CBOW relies purely on co-occurrence statistics. The averaging process acts as a **continuous bag-of-words representation**.

Summary

- Input: multiple context words
- Output: single target word
- Hidden layer: average of context embeddings
- Loss: cross-entropy (negative log-likelihood)

Optimization

$$\arg\min_{V,U} \left\{ -\sum_{t=1}^{T} \log \frac{\exp(u_{w_t}^\top h_t)}{\sum_{i=1}^{|\mathcal{V}|} \exp(u_i^\top h_t)} \right\}.$$

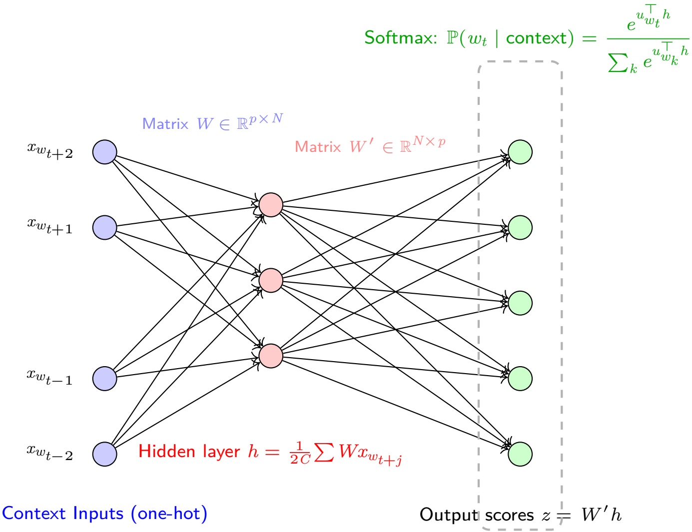
with

- $\mathcal{V}$: the vocabulary (set of all distinct words observed in the corpus). Its vocabulary size is $|\mathcal{V}|$.
- T : total number of tokens (word positions) in the training corpus.
- w_t : the **target (central) word** at position t in the corpus.
- $u_{w_t} \in \mathbb{R}^p$: the **output embedding vector** of the target word w_t, column of $U \in \mathbb{R}^{p\times|\mathcal{V}|}$.
- $v_{w_{t+j}} \in \mathbb{R}^p$: the **input embedding vector** of a context word w_{t+j}, obtained from the matrix $V \in \mathbb{R}^{p\times|\mathcal{V}|}$.
- C : context window size (number of words considered before and after the target word).
- h_t : the **context representation**, computed as the average (or sum) of the embeddings of all context words around w_t.

Gradient updates move embeddings so that true target words receive higher probabilities given their surrounding contexts.

Practical Variants

- ☐ **Negative Sampling:** replaces full softmax by sampled logistic loss.
- ☐ **Sub-sampling:** reduces the impact of very frequent words.
- ☐ **Hierarchical Softmax:** speeds up training for large vocabularies.



Softmax: $\mathbb{P}(w_t \mid \text{context}) = \frac{e^{u_{w_t}^\top h}}{\sum_k e^{u_{w_k}^\top h}}$
Matrix $W \in \mathbb{R}^{p \times N}$
Matrix $W' \in \mathbb{R}^{N \times p}$
$x_{w_{t+2}}$
$x_{w_{t+1}}$
$x_{w_{t-1}}$
$x_{w_{t-2}}$
Hidden layer $h = \frac{1}{2C} \sum W x_{w_{t+j}}$
Context Inputs (one-hot)
Output scores $z = W' h$

Core Idea

The **Skip-Gram** model reverses the logic of CBOW. Instead of predicting a **target word from its context**, it predicts the **context words from the target**.

$$\text{CBOW: (context)} \rightarrow \text{target} \qquad \text{Skip-Gram: (target)} \rightarrow \text{context.}$$

Motivation

Predicting multiple context words from a single central word forces each word embedding to carry enough information to generate its linguistic neighborhood. This encourages more informative word vectors — especially useful for rare words.

Training Signal

Each word supervises its own context — training examples are built automatically from co-occurrences in the corpus. This is again a form of **self-supervised learning**.

Architecture

Input: one central (target) word w_t

Output: $2C$ surrounding context words

Goal: maximize $\mathbb{P}(w_{t+j} \mid w_t)$ for all $j \in [-C, C], j \neq 0$

Hidden Representation

Each input word w_t is represented by its one-hot vector $x_t \in \mathbb{R}^{|\mathcal{V}|}$, and embedded through the matrix $V \in \mathbb{R}^{p \times |\mathcal{V}|}$:

$$v_{w_t} = V x_t.$$

This vector acts as the hidden representation used to predict all context words.

Skip-Gram Forward Flow

$$x_t \xrightarrow{V} v_{w_t} \xrightarrow{U^\top} z \xrightarrow{\text{softmax}} \mathbb{P}(w_{t+j} \mid w_t)$$

Predict each context word separately.

Local Objective

For each central word w_t and each context word w_{t+j} within a window of size C:

$$\mathcal{L}_{t,j} = -\log \mathbb{P}(w_{t+j} \mid w_t) = -\log \frac{\exp(u_{w_{t+j}}^\top v_{w_t})}{\sum_{i=1}^{|\mathcal{V}|} \exp(u_i^\top v_{w_t})}.$$

Global Objective

The Skip-Gram model minimizes the total negative log-likelihood across the entire corpus:

$$\arg\min_{V,U} \left\{ -\sum_{t=1}^{T} \sum_{\substack{j=-C \\ j\neq 0}}^{C} \log \mathbb{P}(w_{t+j} \mid w_t) \right\}.$$

Each co-occurrence (w_t, w_{t+j}) provides a gradient update that moves their embeddings closer.

Interpretation

Frequent co-occurrences (e.g., “the”–“cat”) generate many updates, increasing their similarity. Rare co-occurrences have negligible effect.

Structural Differences

⇒ **CBOW:** predicts the central word from multiple context words. It uses the *average of context embeddings* as input.

⇒ **Skip-Gram:** predicts multiple context words from a single central word. It uses *one input embedding* to produce several predictions.

	CBOW	**Skip-Gram**
Input	Context words	Central word
Output	Central word	Context words
Objective	$\mathbb{P}(w_t \mid \text{context})$	$\mathbb{P}(\text{context} \mid w_t)$
Best for	Frequent words	Rare words
Pros	Much faster	Better quality

GloVe

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:

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

Co-occurrence Matrix

GloVe builds a word–context matrix $X \in \mathbb{R}^{N \times N}$ where:

$$X_{ij} = \text{number of times word } j \text{ appears in the context of } i.$$

Each entry counts how often j occurs **within a fixed-size window around** i.

Example

Sentence: *"The cat sat on the mat."* (window size = 2) For target word *cat*, context = {The, sat}. Thus: $X_{\text{cat,The}}\uparrow$, $X_{\text{cat,sat}}\uparrow$.

Normalization and Probabilities

$$X_i = \sum_{k=1}^{N} X_{ik}, \qquad P_{ij} = \frac{X_{ij}}{X_i}.$$

$X_i =$ total co-occurrences involving word i. $P_{ij} =$ probability that word j appears near i.

Key Intuition

The ratio

$$\frac{P_{ik}}{P_{jk}}$$

encodes semantic relations between words i and j. GloVe learns embeddings that reproduce these global co-occurrence ratios.

Key Idea

Semantic meaning arises from **ratios of co-occurrence probabilities**, not raw counts.

For three words i, j, k:

$$\frac{P_{ik}}{P_{jk}} = \frac{X_{ik}/X_i}{X_{jk}/X_j}.$$

This ratio measures how much more likely k is to appear with i than with j.

Example

$$\frac{P_{\text{ice,solid}}}{P_{\text{steam,solid}}} \text{ large,} \quad \frac{P_{\text{ice,gas}}}{P_{\text{steam,gas}}} \text{ small.}$$

$\Rightarrow$ "solid" relates more to *ice*, while "gas" relates more to *steam*.

Takeaway

Word meaning = pattern of co-occurrences with all other words. GloVe encodes these statistical relationships into geometric distances between vectors.

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:

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

From Co-occurrence Ratios to Linear Geometry

If meaning is captured by co-occurrence ratios, we want:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

For this relationship to be expressed in vector space, F must depend on differences between word vectors, so that directions encode semantic relations:

$$F(w_i, w_j, \tilde{w}_k) = \exp\big((\tilde{w}_k)^\top (w_i - w_j)\big).$$

Taking logarithms gives a linear form:

$$(\tilde{w}_k)^\top (w_i - w_j) = \log P_{ik} - \log P_{jk} \iff w_i^\top \tilde{w}_j + b_i + \tilde{b}_j = \log X_{ij}$$

Hence, the dot product between two word vectors approximates the **logarithm of their co-occurrence frequency**, bridging statistical association and geometric similarity.

Weighting Function and Intuition

The weighting function $f(X_{ij})$ controls the influence of each co-occurrence pair:

$$f(x) = \begin{cases} (x/x_{\text{max}})^{\alpha} & \text{if } x < x_{\text{max}}, \\ 1 & \text{otherwise.} \end{cases}$$

Typical values: $\alpha = 3/4$, $x_{\text{max}} \approx 100$ (Pennington et al., 2014).

- → Frequent pairs (X_{ij} large) are informative but should not dominate.
- → Rare pairs (X_{ij} small) are noisy and thus down-weighted.
- → $f(x)$ smoothly balances the contribution of frequent and infrequent co-occurrences.

Interpretation

- ✓ GloVe performs a **log-linear factorization** of the co-occurrence matrix.
- ✓ It captures both **local context patterns** (like Word2Vec) and **global corpus statistics**.
- ✓ The weighting $f(x)$ ensures stable and efficient learning across frequency scales.

FastText

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.

ELMo

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:

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

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*.

BERT

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:

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

Pretraining Tasks:

- ☐ **Masked Language Modeling (MLM):** Predict masked words from context. $\mathcal{L}_{\text{MLM}} = -\sum_{t \in M} \log \mathbb{P}(w_t \mid w_{\setminus t})$
- ☐ **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.

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.