# Natural Language Processing (NLP) and Large Language Models (LLMs)

An Introduction – Autumn Semester 2024, Dr. Ahmed Abouzeid

This lecture is based on concepts from the book "***Introduction to Artificial Intelligence: A Modern Approach*** **(4th Edition)"** by Stuart Russell and Peter Norvig.
We will be focusing on concepts from:

- **Chapter 24: NLP and Language Models, Parsing, Tasks in NLP**
- **Chapter 25: Word Embeddings, State of the Art Models**

Norwegian University of Science and Technology

# Lecture Outline

- Introduction to NLP
  - Definition and scope
  - History and evolution of NLP
  - Key applications
- Fundamentals of NLP
  - Text preprocessing
  - Basic Language Models
  - Part-of-Speech (POS) tagging
  - Neural Network Approach (Word Embeddings) vs Statistical Approach
- Introduction to LLMs
  - A Quick Overview on the Transformer Architecture
  - Conducting Live Demo of a Transformer

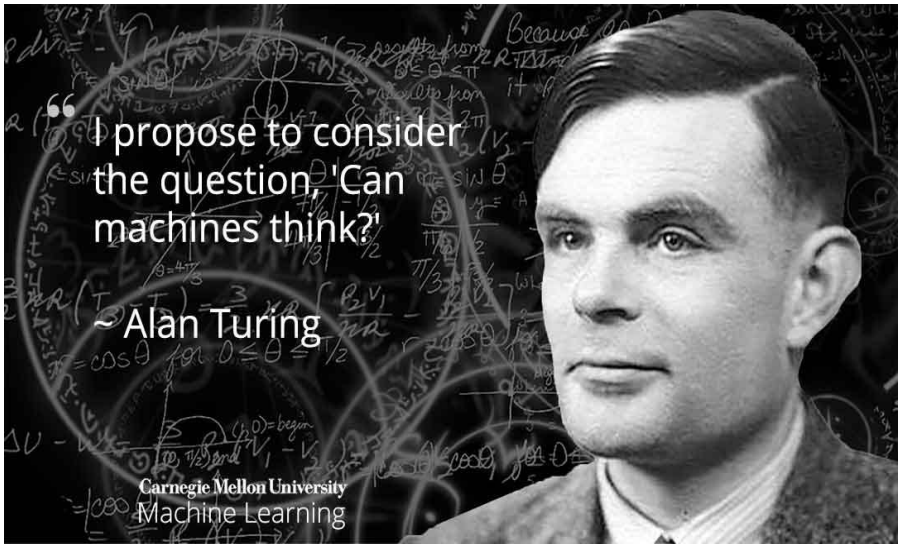
# NLP Definition

- NLP is a subfield in Artificial Intelligence (AI) and Computational Linguistics, focuses on:
  - The interaction between computers and humans' (natural) languages
  - Developing algorithms to interpret, and generate responses to/in human languages, which should be meaningful and useful


**Figure:** Natural Language Processing - Author: Seobility - License: CC BY-SA 4.0

# History and Evolution of NLP

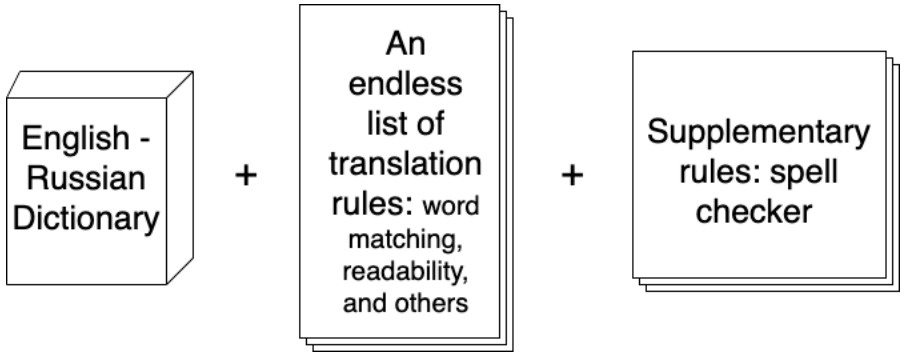
- The early beginings (1950s – 1960s)
  - **1950:** Alan Turing proposes the Turing Test in "Computing Machinery and Intelligence."
  - **1954:** The Georgetown-IBM experiment demonstrates the first machine translation system
  - **1957:** Noam Chomsky publishes "Syntactic Structures," revolutionizing linguistic theory




Check the full details of Alan’s proposal here

# History and Evolution of NLP

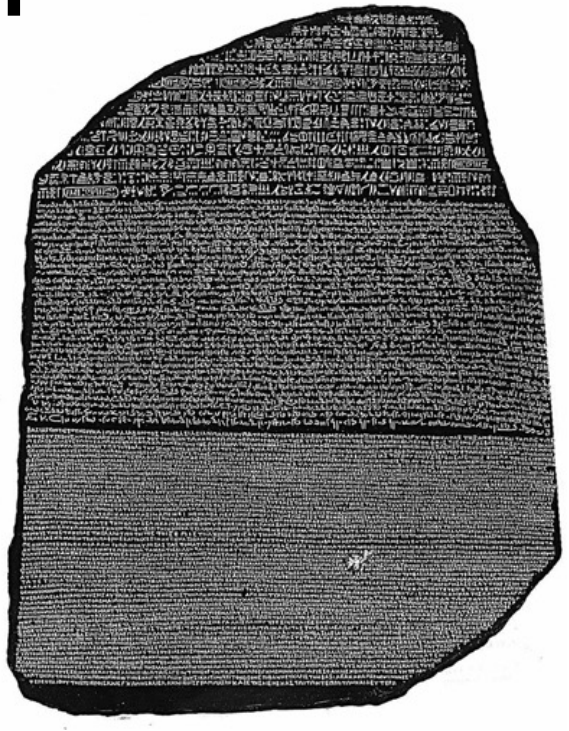
- Rule-Based Systems (1960s-1980s)
  - **1960s-1970s:** Development of rule-based systems for machine translation and text processing
  - **1966:** The ALPAC report leads to reduced funding for MT research in the US
  - **1980:** Introduction of the PROLOG programming language, influencing NLP development




Machine-based translation - Rule-based system approach example

# History and Evolution of NLP

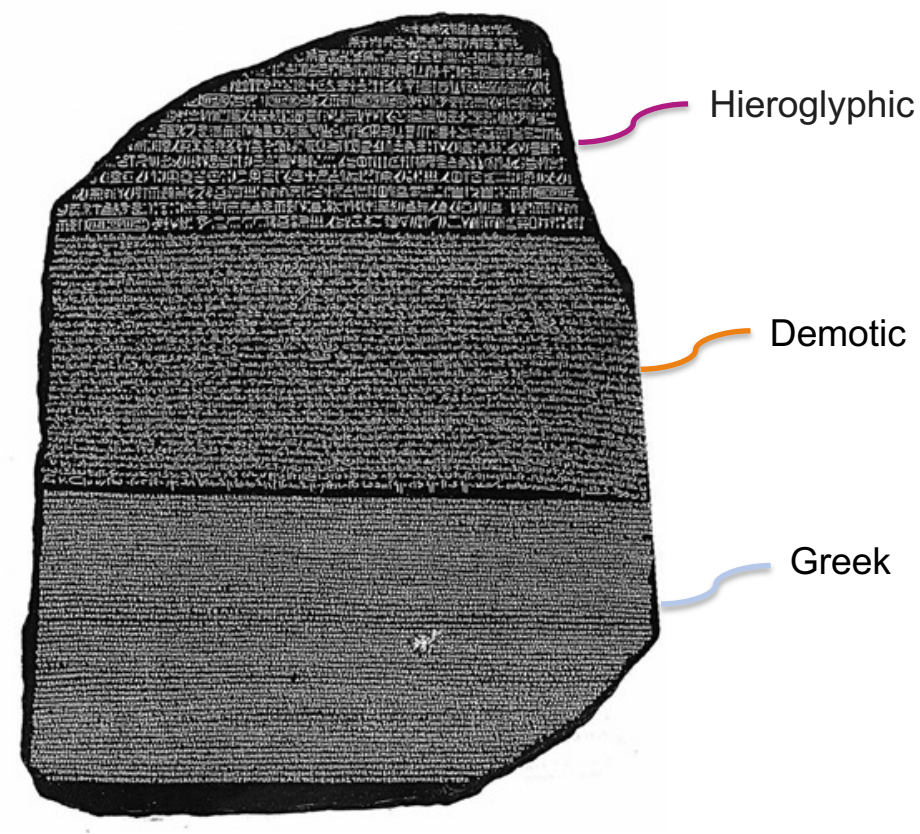
- Rise of Statistical Methods (1980s-1990s)
  - **1988:** Introduction of the first statistical machine translation models
  - **Early 1990s:** Hidden Markov Models (HMMs) become popular for speech recognition
  - **1996:** IBM's statistical translation model (Candide) gains prominence



The Rosetta Stone: An Inspiration for Statistical Machine Translation

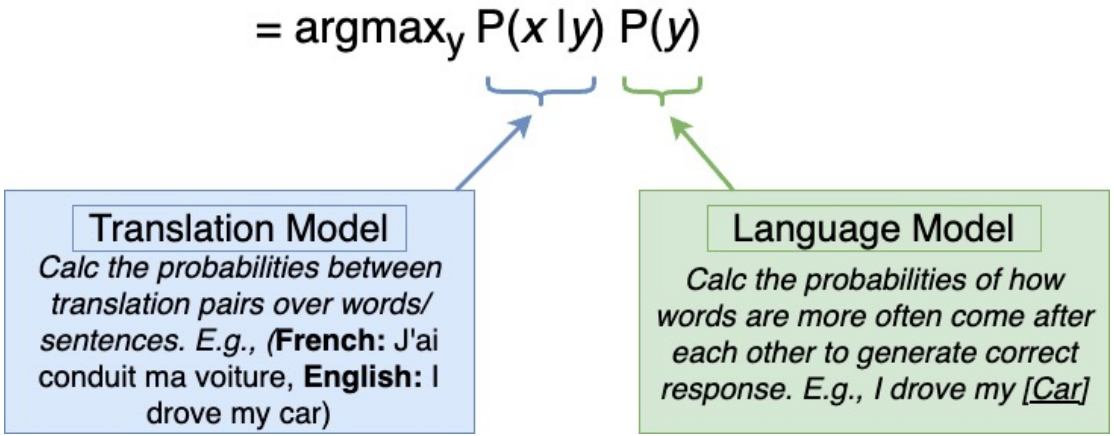
# Rise of NLP Statistical Methods (1980s-1990s)

- Humans need parallel text to understand a new language
- The Rosetta Stone allowed humans to understand the ancient Egyptian language
- Computers need the same type of information
- To build a statistical machine translation model, we need a parallel text dataset. E.g., English-French millions of examples for translated sentences (pairs)
- The model's main task is to calculate the statistical patterns between pair of words/ sentences by calculating the probabilities of occurrences



# Rise of NLP Statistical Methods (1980s-1990s)

- Example of parallel corpus: Source language X (**French**), Target language Y (**English**)
- The task is to translate from X (**French**) to Y (**English**)
- Calculating the probabilities of occurrences:



# Challenges in NLP Statistical Methods

- Many language pairs or specific phrases might not have enough examples, leading to poor translation quality
- Statistical models require extensive feature engineering, where human experts must identify and encode linguistic rules and patterns, which is time-consuming and prone to errors
- Statistical models are more complex, they require significant computational resources and become difficult to scale
- They are more prone to producing translations that are grammatically incorrect or awkward

# History and Evolution of NLP

- Progress towards Deep Learning (2000s-2010s)
  - **2006:** Hinton developed Restricted Boltzmann Machines (RBMs)
  - **2010s**: Deep Learning Revolution

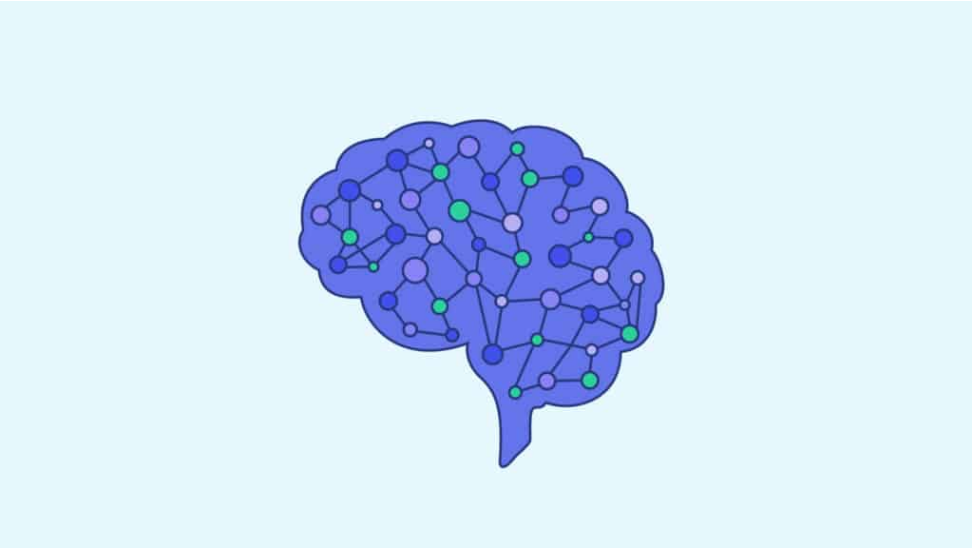
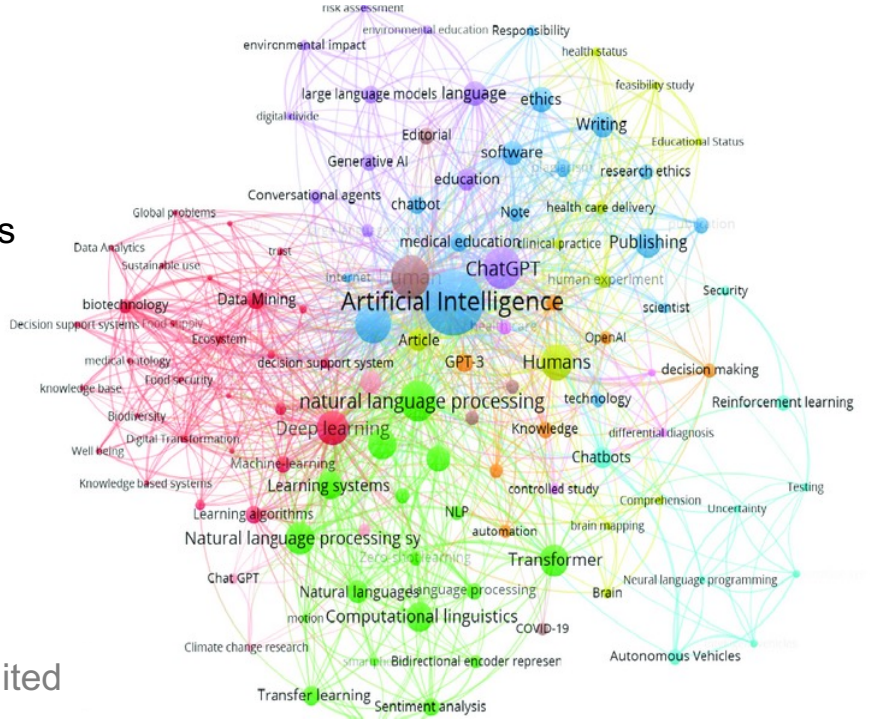


**Figure:** Neural Network imitating the function of human brain - Author: datascientest.com - License: CC BY-SA 4.0

# History and Evolution of NLP

- Modern Advances (2020s)
  - **2020:** GPT-3, developed by OpenAI, showcases the power of large-scale language models
  - **2022:** Emergence of multimodal models combining text and image processing
  - **2024:** Ongoing research in ethical AI and bias mitigation in NLP.




Edited

**Figure:** The thematic landscape of LLMs- Author: Reinaldo Luis Barbosa-Ramos Raidel - License: CC BY- 4.0

# Key Applications of NLP

- Machine Translation: Statistical-based or Neural-based
- Sentiment Analysis: e.g., customer feedback analysis
- Chatbots and AI assistants: e.g., ChatGPT
- Text Summarization and Classification: e.g., identification of a document main topic
- Paraphrasing and Grammar correction: e.g., Grammarly, ChatGPT
- Auto-completion: e.g., the auto-complete in Google Search or mobile devices
- Named Entity Recognition (NER): e.g., identification of locations, organization or personal names
- Part-of-Speech tagging: e.g., recognizing the structure of phrases such as nouns, pronouns, verbs, etc..
- Document Clustering: e.g., grouping documents based on similarity

# Fundamentals of NLP – Text Preprocessing

- Text preprocessing is a crucial step in NLP that involves transforming raw text into a clean and structured format suitable for analysis and modelling. The main goals are:
  - **Noise Reduction:** *Removing irrelevant or redundant information*
  - **Normalization:** *Standardizing text data to a consistent format, which helps in reducing the complexity*
  - **Tokenization:** *Splitting the text into smaller units (tokens) such as words or subwords, which serve as the basic units for analysis*

# Fundamentals of NLP – Text Preprocessing

- An Example:
  - **Sentence 1:** *"Text preprocessing is essential for NLP!".* Tokens: ["Text", "preprocessing", "is", "essential", "for", "NLP"]
  - **Sentence 2:** *"Text preprocessing helps improve NLP models."* Tokens: ["Text", "preprocessing", "helps", "improve", "NLP", "models"]
  - **Combined Tokens from both sentences:** ["Text", "preprocessing", "is", "essential", "for", "NLP", "Text", "preprocessing", "helps", "improve", "NLP", "models"]

# Fundamentals of NLP – Text Preprocessing

- An Example (continued):
  - Tokens after lowercasing (Normalization): ["text", "preprocessing", "is", "essential", "for", "nlp", "text", "preprocessing", "helps", "improve", "nlp", "models"]

    **(Further normalization also include stemming and lemmatization)**

  - Removing Stop Words (Noise Reduction): Not applied for some tasks such as auto-completion
    - Stop Words list: ["is", "for"]
    - Tokens after removing Stop Words: ["text", "preprocessing", "essential", "nlp", "text", "preprocessing", "helps", "improve", "nlp", "models"]

# Fundamentals of NLP – Text Preprocessing

- An Example (continued):

**Frequency Distribution Table**

| Token | Frequency |
|---|---|
| text | 2 |
| preprocessing | 2 |
| essential | 1 |
| nlp | 2 |
| helps | 1 |
| improve | 1 |
| models | 1 |

Statistical approach to NLP

# The N-grams Language Model

- A language model can be defined as a probability distribution describing the likelihood of any string
  - The model should say that *“Do I dare disturb the universe?”* has a reasonable probability as a string of English, but *“Universe dare the I disturb do?”* is extremely unlikely
  - With a language model, we can predict what words are likely to come next in a text, and thereby suggest completions for an email or text message based on the calculated statistics of how often words come after each other from some data. E.g., Wikipedia, or other textual datasets
  - **N**-grams: the model calculates the statistical representation of contiguous sequences of **N** items (Tokens) from a given text or speech. The items can be words, characters, or other units of text

# The N-grams Language Model

- In the sentence *”Text preprocessing is essential for NLP!”*
  - 1-grams (Unigrams) representations*:* [”text", "preprocessing", "is", "essential", "for", ”nlp"]
  - 2-grams (Bigrams) representations: [”text preprocessing", "preprocessing is", "is essential", "essential for", "for nlp"]
  - 3-grams (Trigrams) representations: [”text preprocessing is", "preprocessing is essential", "is essential for", "essential for nlp"]

# The Bigrams Language Model

### Combined Bigrams Frequency Count:

| Bigram | Frequency |
|---|---|
| Text preprocessing | 2 |
| preprocessing is | 1 |
| is essential | 1 |
| essential for | 1 |
| for NLP | 1 |
| preprocessing helps | 1 |
| helps improve | 1 |
| improve NLP | 1 |
| NLP models | 1 |

# The Bigrams Language Model

- For an auto-completion task of the input: *“text pre”*

**Probability of Next Token $w_{i+1}$ Given $w_i$:**

$$P(w_{i+1}|w_i) = \frac{\text{Count}(w_i, w_{i+1})}{\text{Count}(w_i)}$$

- **Calculate Probabilities:**
  - **For "preprocessing" after "text":**

$$P(\text{"preprocessing"}|\text{"text"}) = \frac{\text{Count}(\text{"text"}, \text{"preprocessing"})}{\text{Count}(\text{"text"})} = \frac{2}{2} = 1$$

  - **For "helps" after "preprocessing":**

$$P(\text{"helps"}|\text{"preprocessing"}) = \frac{\text{Count}(\text{"preprocessing"}, \text{"helps"})}{\text{Count}(\text{"preprocessing"})} = \frac{1}{2} = 0.5$$

# The Bigrams Language Model

- Given the previous slide’s calculations, the auto-complete suggests:
  - **For Input *"text pre":***
    - **Suggestion 1:** "text preprocessing" (high probability, based on bigram count)
    - **Suggestion 2:** "preprocessing helps" (lower probability but valid)

# The N-grams Language Model

- To generalize the calculations on any number **(N)** of contiguous sequences:

$$P(w_{1:N}) = \prod_{j=1}^{N} P(w_j \mid w_{1:j-1}).$$

# Fundamentals of NLP – Part-of-Speech (POS)

- A statistical language model such as N-grams suffers when it needs to generalize beyond the data it calculates frequencies from
  - The model can auto-complete a sentence only based on what was available in its source dataset
  - To help overcoming the generalization limitation, more knowledge should be modelled as well
  - Hence, a structured representation of the model source dataset is recommended. POS is an example for such structured information

# Fundamentals of NLP – Part-of-Speech (POS)

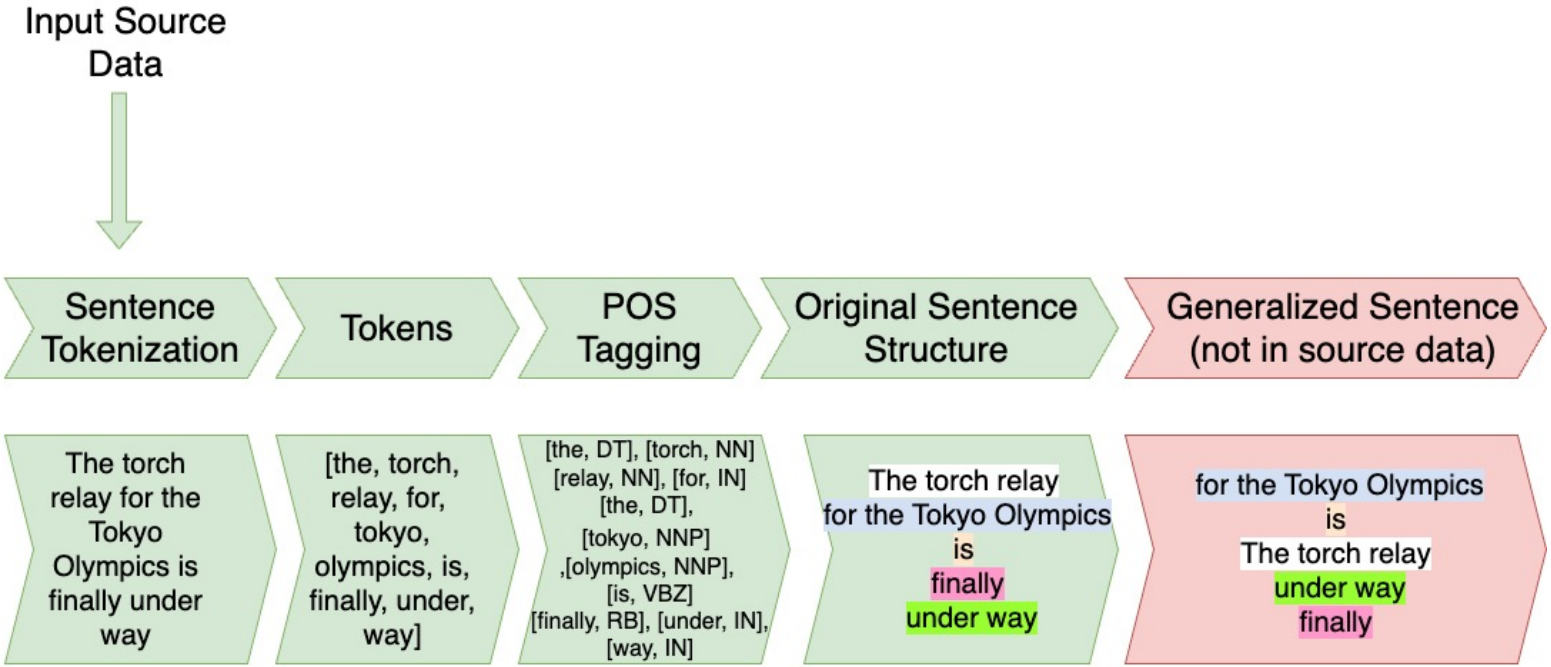



**Figure:** Example of Part-of-Speech (POS) process - Author: Brendon Albertson - License: CC BY- 4.0

Edited

# Fundamentals of NLP – The Neural Approach

- Neural Network-based approach improves a language model and helps reducing manual feature engineering in the source datasets and allow for more generalization through some techniques:

  - **Word Embeddings:** a representation of words that does not require manual feature engineering, but allows for generalization between related words:

    - "colorless" and "ideal" are both adjectives, a syntax that can be learned automatically
    - "cat" and "kitten" are both felines, a semantic that can be learned automatically
    - "awesome" has opposite sentiment to "cringeworthy", a sentiment that can be learned automatically

# Fundamentals of NLP – The Neural Approach

- Traditionally, a word can be represented by a vector that hopefully capture its contextual meaning and how to capture the latter from the words that often come within its context:
  - Following the linguist John R. Firth's (1957) maxim, "You shall know a word by the company it keeps"
  - We could represent each word with a vector of n-gram counts of all the phrases that the word appears in. However, with a 100,000-word vocabulary, there are $10^{25}$ 5-grams to keep track of (although vectors in this $10^{25}$-dimensional space would be quite sparse—most of the counts would be zero
  - We would get better generalization if we reduced this to a smaller-size vector, perhaps with just a few hundred dimensions. We call this smaller, dense vector: a **word embedding**

# Fundamentals of NLP – The Neural Approach

- Word Embeddings are learned automatically
  - They capture implicit relationships also, such as aunt to niece is same as uncle to nephew
  - They solve the computational overload in statistical NLP approaches

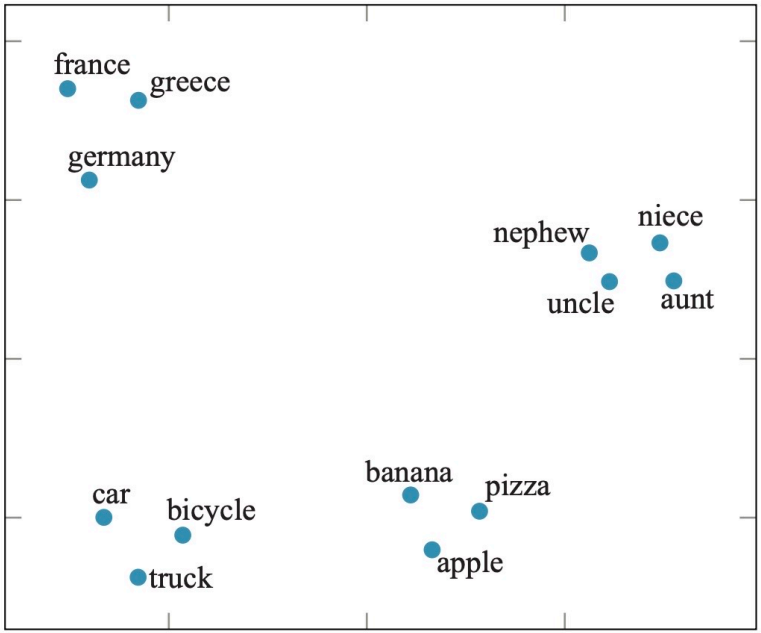



**Figure**: Word embedding vectors computed by the GloVe algorithm trained on 6 billion words of text. 100-dimensional word vectors are projected down onto two dimensions in this visualization. Similar words appear near each other

# Mathematical Foundation for Word Embeddings

- Using Word2Vec Method: Predict context words from a given target word

1. **Setup**:
   - **Vocabulary Size**: Assume a vocabulary of size $V$ (e.g., 10,000 words).
   - **Embedding Dimension**: Assume $d$ (e.g., 300 dimensions).
2. **Training Data**:
   - Example Sentence: "The quick brown fox jumps over the lazy dog."
   - For target word "fox" and context words "quick," "brown," "jumps," "over" within a window of size 2.
3. **One-Hot Encoding**:
   - Suppose the vocabulary indices are as follows:
     - "fox" → Index 3
     - "quick" → Index 7
     - "brown" → Index 12
     - "jumps" → Index 15
     - "over" → Index 18

- **One-Hot Vectors**:
  - "fox": $\mathbf{x}_{\text{fox}} = [0, 0, 1, 0, \ldots, 0]$ (1 at index 3)
  - "quick": $\mathbf{x}_{\text{quick}} = [0, 0, 0, 0, \ldots, 1, \ldots, 0]$ (1 at index 7)

# Mathematical Foundation for Word Embeddings

4. **Embedding Matrix $\mathbf{W}$:**
   - Initialize $\mathbf{W} \in \mathbb{R}^{V \times d}$, where each row $\mathbf{w}_i$ is the embedding vector for word $i$.
     - Suppose $\mathbf{W}$ is:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_V \end{bmatrix}$$

     - Where $\mathbf{w}_{\text{fox}}$ is a row vector in $\mathbb{R}^d$ (e.g., 300 dimensions).

# Mathematical Foundation for Word Embeddings

5. **Objective Function**:
   - Maximize the probability of context words given the target word. For a target word $w_t$ and context words $w_c$:

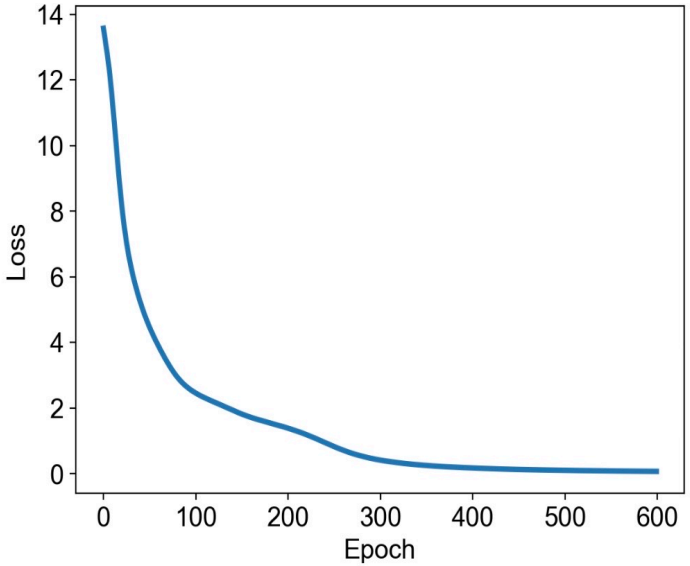
$$\text{Loss} = -\sum_{c \in \text{context}} \log P(w_c \mid w_t)$$

   - Using the softmax function for prediction:

$$P(w_c \mid w_t) = \frac{\exp(\mathbf{v}_{w_c}^T \mathbf{v}_{w_t})}{\sum_{w \in V} \exp(\mathbf{v}_w^T \mathbf{v}_{w_t})}$$

   - Here, $\mathbf{v}_{w_t}$ is the embedding vector of the target word $w_t$, and $\mathbf{v}_{w_c}$ is the embedding vector of the context word $w_c$.
6. **Gradient Update**:
   - Update $\mathbf{W}$ using gradient descent:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}} \text{Loss}$$



# Mathematical Foundation for Word Embeddings

- Using ELMo (Embeddings from Language Models) Method:
  - **Context Awareness**: Captures the meaning of words in the context of the entire sentence
  - **Dynamic Representations**: Produces different embeddings for the same word in different contexts
  - **Improved Performance**: Typically provides better performance on downstream tasks due to its ability to understand context

# Mathematical Foundation for Word Embeddings

- Using ELMo (Embeddings from Language Models) Method:

**1. Setup**:

- **Vocabulary Size**: Assume a vocabulary of size $V$ (e.g., 10,000 words).
- **Embedding Dimension**: Assume $d$ (e.g., 300 dimensions).
- **Context**: The entire sentence is used for context, not a fixed window.

**2. Training Data**:

- **Example Sentence**: "The quick brown fox jumps over the lazy dog."
- **Target Word**: Consider the word "fox".
- **Context**: Context is the entire sentence; thus, both preceding and following words are used.

**3. Initial Embeddings**:

- **Word Embeddings**: Before training, each word is initialized with embeddings, often using pre-trained embeddings like Word2Vec or GloVe.

# Mathematical Foundation for Word Embeddings

**4. Embedding Matrix $W$:**

- **Initialization**: Initialize $W \in \mathbb{R}^{V \times d}$, where each row $w_i$ is the embedding vector for word $i$.

  Suppose $W$ is:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_V \end{bmatrix}$$

  where $w_{\text{fox}}$ is a row vector in $\mathbb{R}^d$ (e.g., 300 dimensions).

# Mathematical Foundation for Word Embeddings

**5. Objective Function**:

- **Language Modeling Objective**: Train the model to predict the next word in the sequence (language modeling). ELMo uses a bidirectional LSTM to generate embeddings from both directions.
- **Cross-Entropy Loss**: For each position $t$ in the sentence, the loss is calculated based on predicting the next word in the sequence:

$$\text{Loss} = -\sum_{t=1}^{T} \log P(w_t \mid w_{<t}, w_{>t})$$

where $w_{<t}$ and $w_{>t}$ are the words before and after position $t$, respectively.

# Mathematical Foundation for Word Embeddings

**6. Bidirectional LSTM Processing**:

- **Forward LSTM**:
  - Processes the sentence from left to right, generating embeddings that incorporate the preceding context.
- **Backward LSTM**:
  - Processes the sentence from right to left, generating embeddings that incorporate the following context.
- **Combining Representations**:
  - The final representation for each word is a combination of the forward and backward LSTM outputs:

$$\mathbf{h}_t = \alpha \cdot \mathbf{h}_{\text{forward},t} + (1 - \alpha) \cdot \mathbf{h}_{\text{backward},t}$$

  where $\mathbf{h}_{\text{forward},t}$ and $\mathbf{h}_{\text{backward},t}$ are the embeddings from the forward and backward LSTMs, respectively, and $\alpha$ is a hyperparameter that balances the two representations.

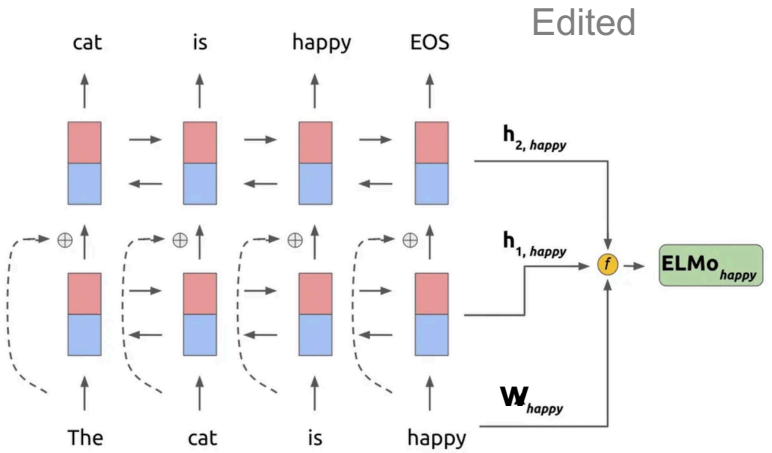



**Figure**: An example of combining the bidirectional hidden representations and word representation for "happy" to get an Elmo-specific representation – Author: Karan Purohit

# Mathematical Foundation for Word Embeddings

**7. Gradient Update:**

- **Backpropagation Through Time (BPTT):**
  - Compute gradients for the LSTM parameters using BPTT. This involves:
    - **Computing gradients for LSTM weights** based on the loss function, taking into account the gradients from both directions (forward and backward).
- **Update Parameters:**
  - Update the parameters of the LSTM and embedding layers using gradient descent:

$$\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} - \eta \cdot \nabla_{\mathbf{W}} \text{Loss}$$

where $\eta$ is the learning rate, and $\nabla_{\mathbf{W}} \text{Loss}$ are the gradients computed through BPTT.

# ELMo vs Word2Vec

- **Deep Representation**:
  - **ELMo**: Utilizes deep bidirectional LSTMs, capturing richer and more complex representations of words by processing text in both forward and backward directions
  - **Word2Vec**: Uses shallow neural networks, which do not account for deep contextual information
- **Handling Polysemy**:
  - **ELMo**: Handles polysemy more effectively by generating different embeddings for the same word depending on its context, allowing for more accurate semantic representation
  - **Word2Vec**: Struggles with polysemy because it produces a single embedding per word

# ELMo vs Recent Advances in NLP (Transformers)

- **Computational Efficiency**:
  - **ELMo**: Computationally expensive due to their sequential nature. Training and inference can be slower as each word is processed sequentially from both directions
  - **Transformers**: Allow for parallel processing of the entire sequence. This parallelization speeds up training and inference, making Transformers more efficient and scalable
- **Long-Range Dependencies**:
  - **ELMo**: While it captures context from both directions, they can still struggle with very long-range dependencies due to limitations in their capacity to retain information over long sequences
  - **Transformers**: Allow direct access to any part of the sequence, enabling better handling of long-range dependencies and relationships between distant words
- **Dynamic Embeddings**:
  - **ELMo**: Embeddings are generated from bidirectional LSTMs but still rely on static word embeddings as input. This can limit the dynamic nature of the representations.
  - **Transformers**: Generate embeddings dynamically based on the entire sequence context

# Transformers

- **Self-Attention**:
  - Captures relationships between all words in a sequence
  - Handles long-range dependencies
- **Multi-Head Attention**:
  - Learns different relationships with multiple attention heads
  - Enhances representation richness
- **Positional Encoding**:
  - Adds position information to handle word order
  - Maintains sequence structure

Parallel Corpus

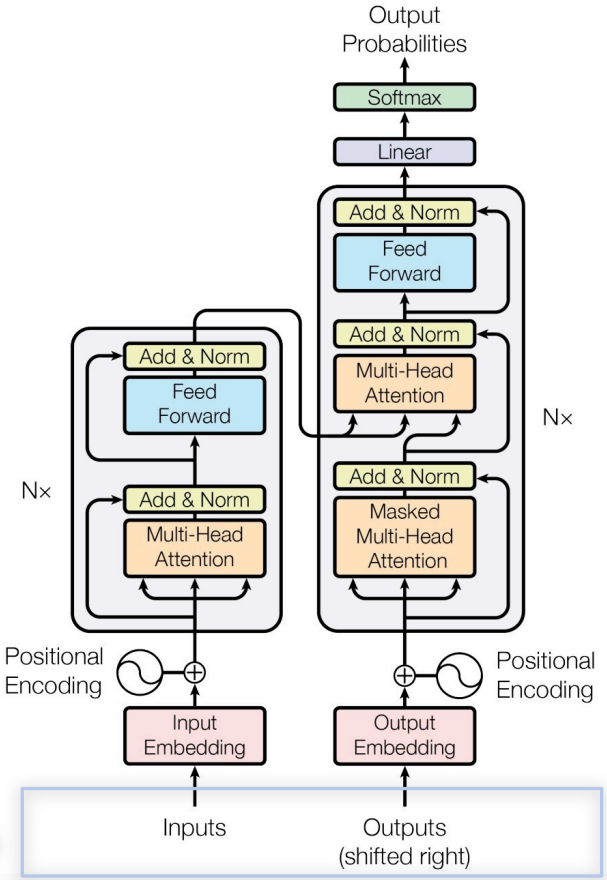



**Figure**: Transformer architecture – From the paper "Attention Is All You Need"

# Quick Live Demo to Fine-Tune a Pretrained Transformer Model

# Thank You!

# Questions?