The Classic NLP Pipeline

Before the deep learning era, NLP systems were typically built as a multi-stage pipeline. Each stage performed a discrete task, and its output was fed as input to the next stage.

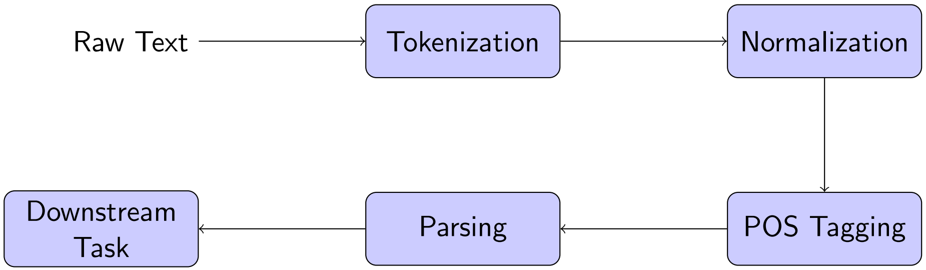

Figure: A simplified view of the classic, sequential NLP pipeline.

A critical weakness of this architecture is **error propagation**. An error made in an early stage (e.g., incorrect tokenization) will cascade and negatively impact all subsequent stages, leading to a brittle system. This

Text Representation: From Words to Vectors

Machines operate on numbers, not words. A core task in NLP is to convert text into a numerical format. Early methods relied on creating high-dimensional, sparse vectors.

- **Bag-of-Words (BoW):**
  - Represents a document as a vector of word counts.
  - Simple and efficient, but it completely ignores grammar, word order, and context.
  - Example: "The cat sat on the mat" and "The mat sat on the cat" have identical BoW representations.
- **Term Frequency-Inverse Document Frequency (TF-IDF):**
  - An improvement over BoW that weights words based on their importance.
  - A word's weight is high if it appears frequently in a document (*Term Frequency*) but rarely in the overall corpus (*Inverse Document Frequency*).
  - This helps to filter out common stop words (like "the", "a") and highlight important terms.

These methods are foundational but are limited by their inability to capture semantic meaning [1].

Sequence Labeling: Part-of-Speech (POS) Tagging

- **Definition:** The process of assigning a grammatical category (part of speech) to each word in a text.
  - Examples: Noun (NN), Verb (VB), Adjective (JJ), Adverb (RB), etc.
- **Example:**
  - Input: "The quick brown fox jumps over the lazy dog."
  - Output: "The/DT quick/JJ brown/JJ fox/NN jumps/VBZ over/IN the/DT lazy/JJ dog/NN."
- **Importance:** POS tagging is a crucial preprocessing step for many downstream tasks.
  - It helps in syntactic parsing by providing information about the grammatical structure of a sentence.
  - It aids in named entity recognition and information extraction by helping to identify nouns and noun phrases.
- Modern tools like SpaCy can achieve very high accuracy on this task using machine learning approaches.[1]

Sequence Labeling: Named Entity Recognition (NER)

- **Definition:** A subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories.
  - Common categories: Person (PER), Organization (ORG), Location (LOC), Date/Time, etc.
- **Example:**
  - Input: "Apple is looking at buying a U.K. startup for $1 billion."
  - Output: "[**Apple**]$_{ORG}$ is looking at buying a [**U.K.**]$_{LOC}$ startup for [**$1 billion**]$_{MONEY}$."
- **Applications:**
  - **Information Retrieval:** Enhancing search queries (e.g., searching for "Jobs" as a person, not a generic noun).
  - **Knowledge Base Construction:** Automatically populating databases with structured information extracted from unstructured text.
  - **Content Classification:** Summarizing the key entities mentioned in a document.

Statistical Models: Hidden Markov Models (HMMs)

- HMMs were a cornerstone of the statistical era of NLP, especially for sequence labeling tasks like POS tagging.[1]
- **Core Idea:** An HMM assumes that a sequence of observations is generated by a sequence of hidden states. We can't see the states, only the observations.
- **Analogy: Weather and Ice Cream**
  - **Hidden States:** The weather each day (e.g., Sunny, Rainy). We don't know it directly.
  - **Observable Emissions:** The number of ice creams sold each day. We can measure this.
  - **The Model's Job:** Given a sequence of ice cream sales, infer the most likely sequence of weather conditions that produced it.
- **Application to POS Tagging:**
  - **Hidden States:** The sequence of POS tags (NN, VB, JJ...).
  - **Observable Emissions:** The sequence of words in the sentence.
  - The HMM learns transition probabilities (e.g., P(NN — DT)) and emission probabilities (e.g., P("cat" — NN)) from data to find the most likely tag sequence for a given word sequence.[1]

Statistical Models: Conditional Random Fields (CRFs)

- CRFs are another type of statistical model that became a dominant method for sequence labeling, often outperforming HMMs.[1]
- **Key Advantage over HMMs:**
  - HMMs are *generative* models; they model the joint probability $P(\text{tags}, \text{words})$.
  - CRFs are *discriminative* models; they model the conditional probability $P(\text{tags}|\text{words})$ directly. This is often an easier problem to solve.
- **Practical Benefit:** CRFs can incorporate a rich set of overlapping features from the input sequence without violating independence assumptions.
  - For NER, a CRF can use features like: Is the word capitalized? What is the preceding word? Does the word contain numbers? Is it in a dictionary of company names?
- This flexibility made CRFs the state-of-the-art for tasks like NER before the advent of deep learning.[1]

Strengths and Weaknesses of Traditional Approaches

Strengths

- **Interpretability:** The features used in models like CRFs are often human-readable, making it easier to understand why the model made a particular decision.[1]
- **Data Efficiency:** They can often achieve reasonable performance with smaller datasets compared to data-hungry deep learning models.
- **Well-understood Theory:** Based on solid statistical foundations.

Weaknesses

- **Feature Engineering:** Their performance is highly dependent on the quality of hand-crafted features. This is a labor-intensive and brittle process requiring significant domain expertise.
- **Context Limitation:** They struggle to capture long-range dependencies and complex semantic relationships in text.
- **Error Propagation:** The pipeline architecture is inherently fragile, as discussed.

Day 2 Student Tasks: Introduction

Today's tasks focus on implementing the classic NLP pipeline and thinking about how to improve upon its components.

Application Task

Build a Text Analysis Tool

Implement a basic NLP pipeline to process and visualize text features.

Research Task

Improve a Traditional Model

Propose a novel way to enhance the feature set of a classical model like a CRF.

Day 2 Student Tasks: Application

Application Idea: Text Analysis Tool

- **Task:** Using a library like NLTK or SpaCy in Python, build a simple web application (e.g., using Flask or Streamlit) that takes a paragraph of text as input from a user.
- The application should process the text and visualize the following outputs:
  1. The list of tokens after tokenization.
  2. The Part-of-Speech (POS) tag for each token.
  3. Any identified Named Entities (NER) highlighted with their corresponding labels (e.g., PER, ORG, LOC).
- **Goal:** This will give you a practical understanding of the components of the classic NLP pipeline.

Day 2 Student Tasks: Research

Paper Idea: Improving CRFs for NER

- **Background:** Traditional CRFs for NER rely heavily on hand-crafted features (e.g., capitalization, word shape). Their performance suffers when encountering new or unseen entities.
- **Research Proposal:** Propose a novel set of features that could be incorporated into a CRF to improve its performance on identifying emerging entities, such as new company names, products, or slang terms.
- **Example Idea:** Could you derive features from a public knowledge base like Wikipedia or Wikidata? For instance, a feature could be "Is this n-gram a title of a new Wikipedia article created in the last month?". How would you engineer and integrate such features?