edureka!
a Veranda Enterprise | ILLINOIS TECH

POST GRADUATE PROGRAM IN GENERATIVE AI AND ML

Natural Language Processing

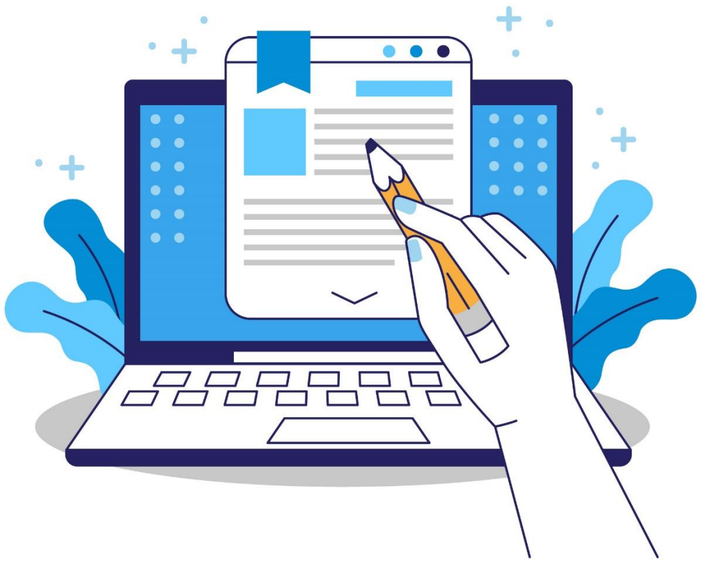
edureka!
a Veranda Enterprise

ILLINOIS TECH

Named Entity Recognition (NER) & Parsing

Topics

- Named Entity Recognition Techniques and Applications
- Pretrained and Transformer-Based NER Models
- Handling Ambiguity and Overlapping Entities in NER
- Parsing Algorithms and Dependency Parsing in NLP
- Syntax Tree Construction and Applications
- Relation and Coreference Extraction Techniques
- Text Summarization Methods and Evaluation Metrics



Learning Objectives

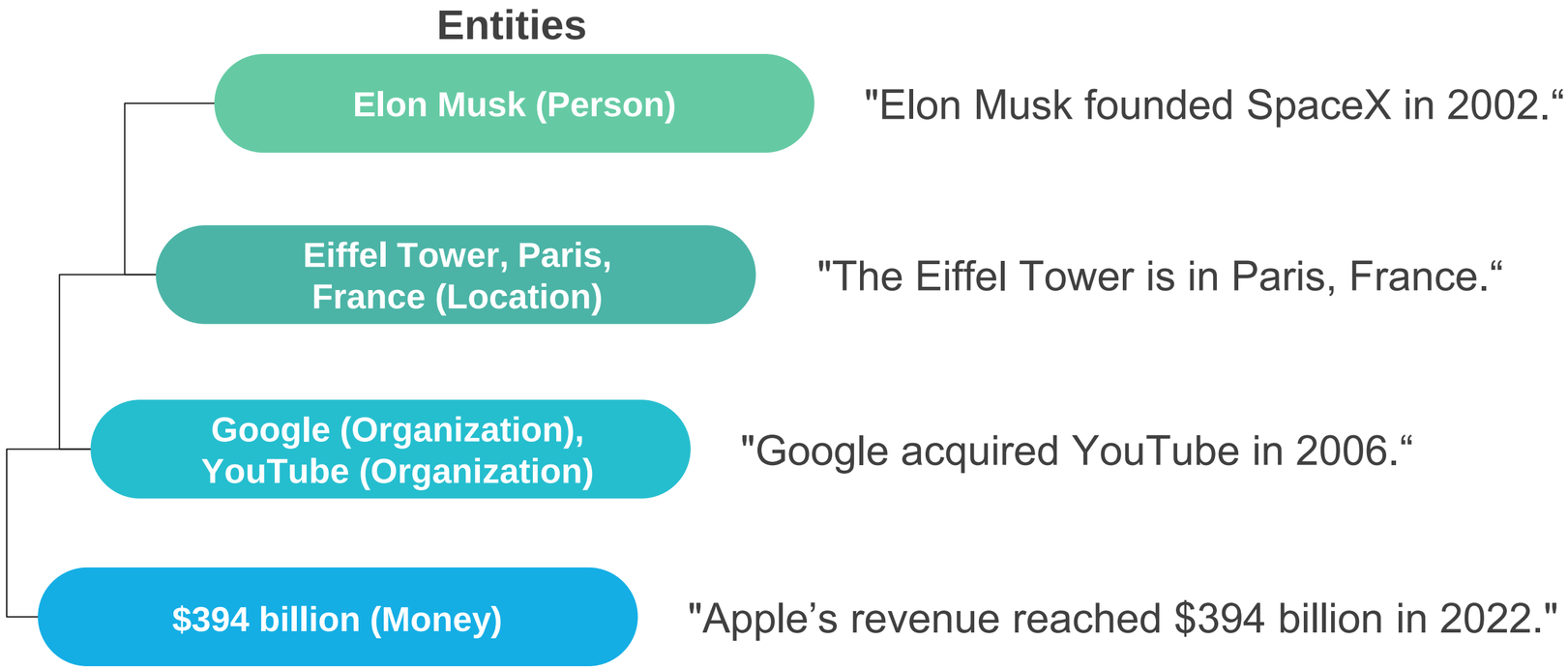
By the end of this lesson, you will be able to:

- Describe core techniques and applications of Named Entity Recognition in NLP.
- Compare pretrained and transformer-based NER models for improved entity extraction.
- Analyze strategies for resolving ambiguity and overlapping entities in NER.
- Demonstrate the use of parsing algorithms, syntax trees, and dependency parsing.
- Evaluate relation extraction, coreference resolution, and summarization methods in NLP tasks.

Introduction to NER

What is NER?

Named Entity Recognition (NER) focuses on detecting **"named entities"** which are significant elements such as names of people, locations, organizations, events, and products.



Rule-Based NER

A **rule-based approach** relies on predefined linguistic rules, patterns, and dictionaries to extract entities. It often uses **regular expressions, lexicons, and heuristic rules** to detect entities.

How It Works:

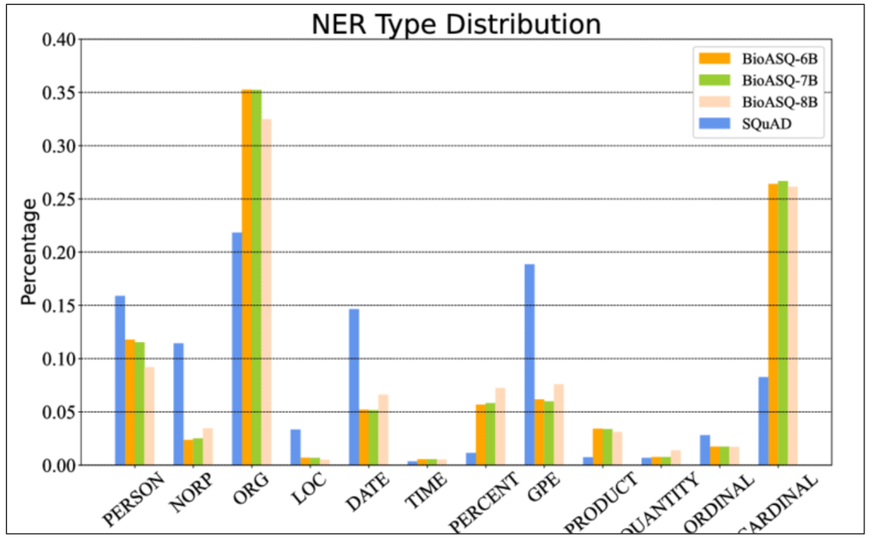
- Uses dictionaries of names, places, or domain-specific terms (e.g., a list of disease names in medical text).
- Applies pattern-matching rules (e.g., capitalized words followed by "Inc." or "Ltd." might be company names).
- Uses part-of-speech (POS) tagging to recognize names (proper nouns) or numerical entities (dates, prices).

Statistical NER

A statistical approach uses machine learning models trained on labeled data to predict entities based on contextual patterns.

How It Works:

- Learns from annotated text data (e.g., sentences labeled with "PERSON," "LOCATION," etc.).
- Uses contextual features, such as surrounding words, POS tags, and character sequences, to detect entities.
- Can generalize to unseen data, unlike rule-based systems.

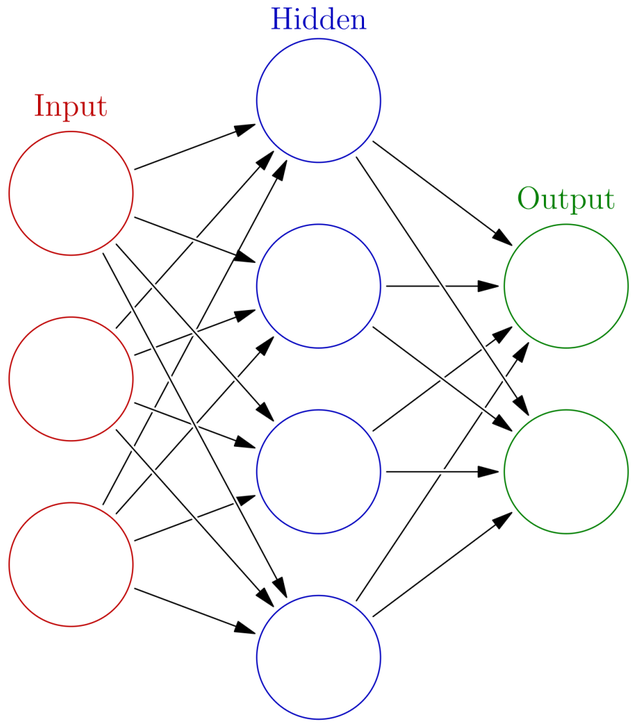


Neural NER

A neural approach uses deep learning models like Long Short-Term Memory (LSTM), Bidirectional LSTMs (BiLSTMs), and Transformer-based models to recognize entities without predefined rules or handcrafted features.

How It Works:

- Uses word embeddings (e.g., Word2Vec, GloVe) to capture semantic relationships between words.
- Uses contextualized models (e.g., BERT) to understand words based on the entire sentence rather than predefined patterns.
- Can adapt to different domains with minimal manual intervention.



Why use NER?

Efficient Information Extraction:

Improves Search and Retrieval

Enhances Context Understanding

Supports Domain-Specific Tasks

Facilitates Applications

Supports Relationship Extraction

Limitations of NER

Ambiguity in Entity Names

Out-of-Domain Challenges

Complex Sentence Structures

Dynamic Vocabulary

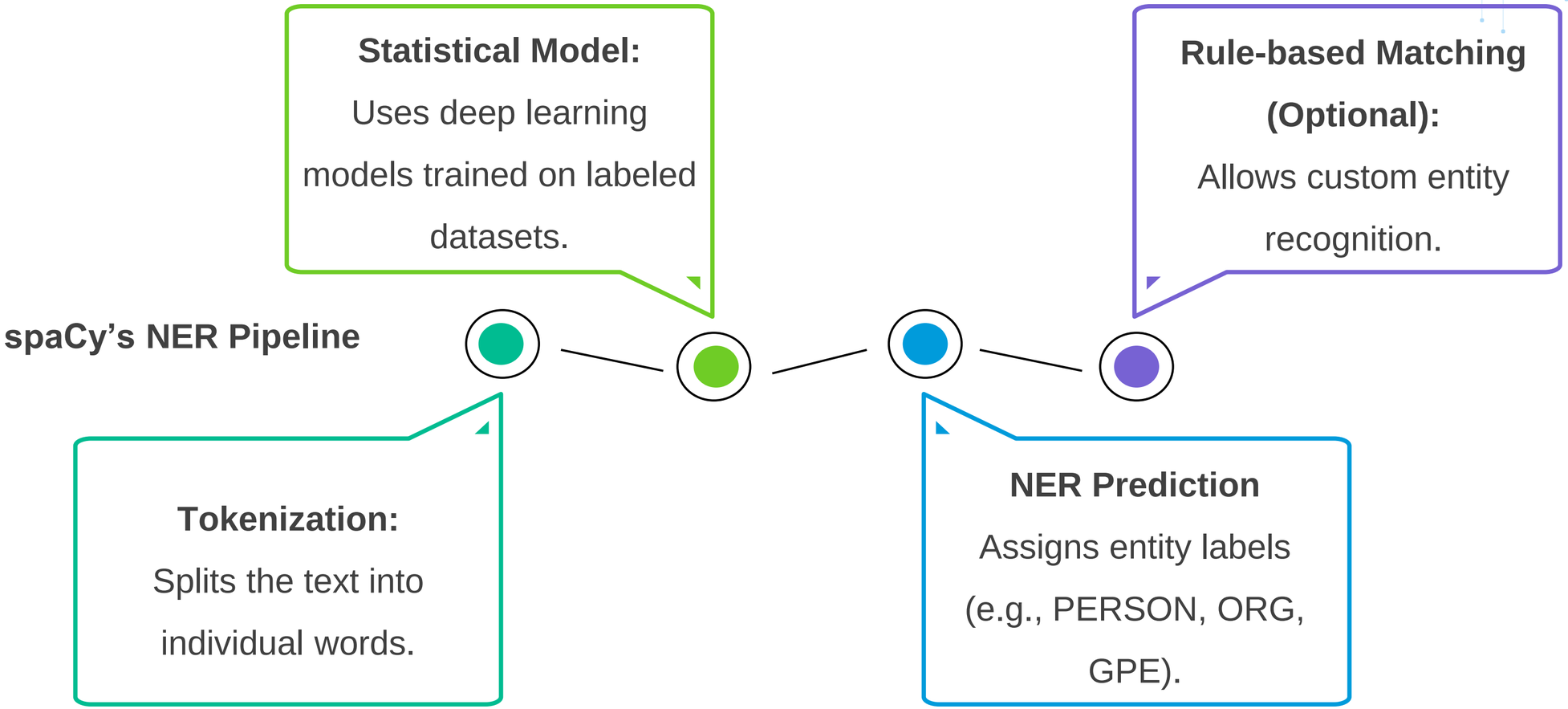
Pretrained NER Models: SpaCy, StanfordNLP

NER with spaCy and Stanford NLP

Named Entity Recognition (NER) is an NLP technique used to identify and classify entities like persons, organizations, locations, dates, etc., in text. Both **spaCy** and **Stanford NLP** provide robust NER capabilities, but they differ in approach, architecture, and performance.

spaCy

spaCy Pipeline in NER



Example Code in spaCy

```
import spacy

# Load the pre-trained NER model
nlp = spacy.load("en_core_web_sm")

# Input text
text = "Elon Musk founded SpaceX in 2002 and Tesla is
based in California."

# Process the text
doc = nlp(text)

# Extract entities
for ent in doc.ents:
    print(f"Entity: {ent.text}, Label: {ent.label_}")
```

Code Output:

Entity: Elon Musk, Label: PERSON

Entity: SpaceX, Label: ORG

Entity: 2002, Label: DATE

Entity: Tesla, Label: ORG

Entity: California, Label: GPE

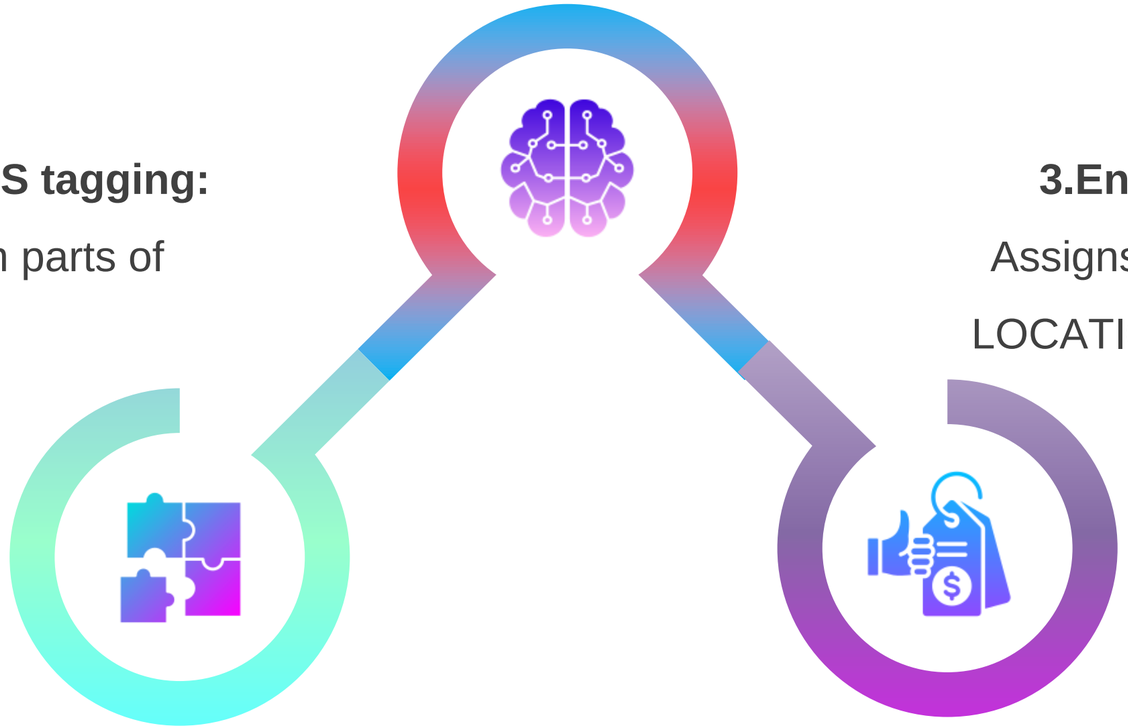
Stanford NLP (Stanza) Pipeline in NER

2.BiLSTM-CRF Model:

Recognizes entities using a deep learning model.

1. Tokenization & POS tagging:

Prepares words with parts of speech.



3.Entity Classification:

Assigns tags (e.g., PERSON, LOCATION, ORGANIZATION).

Example Code in Stanza

```
import stanza

# Load the pre-trained NER pipeline
stanza.download('en')  # Only needed once
nlp = stanza.Pipeline(lang='en',
processors='tokenize,ner')

# Input text
text = "Elon Musk founded SpaceX in 2002 and Tesla is
based in California."

# Process the text
doc = nlp(text)

# Extract entities
for sentence in doc.sentences:
    for ent in sentence.ents:
        print(f"Entity: {ent.text}, Label: {ent.type}")
```

Code Output:

Entity: Elon Musk, Label: PERSON

Entity: SpaceX, Label: ORGANIZATION

Entity: 2002, Label: DATE

Entity: Tesla, Label: ORGANIZATION

Entity: California, Label: LOCATION

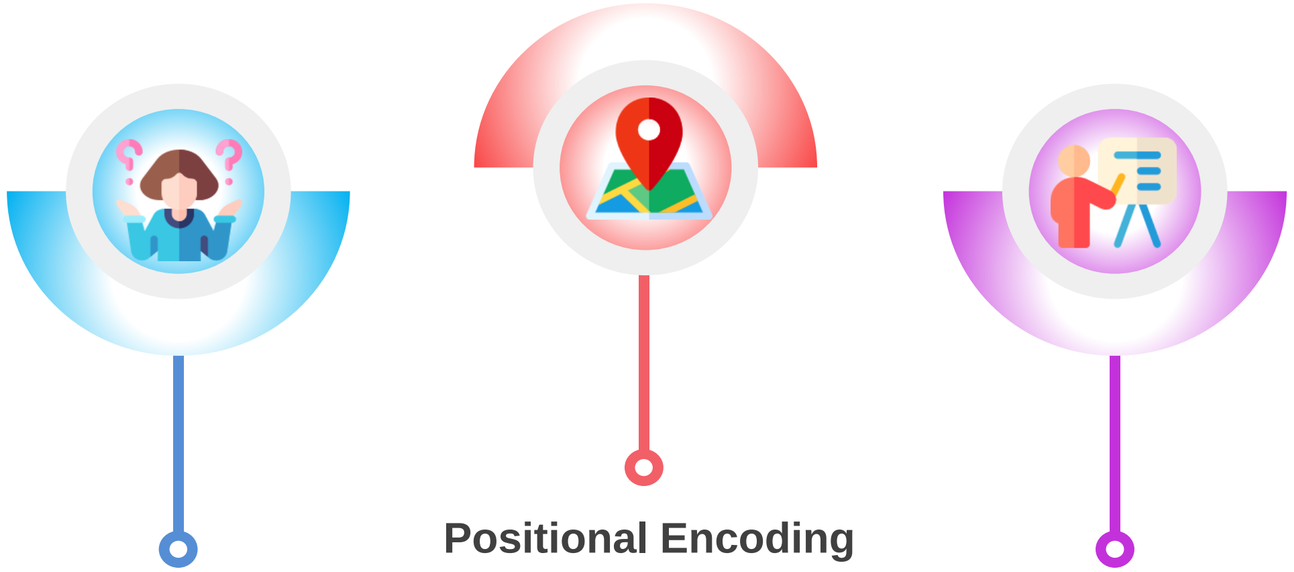
Comparison between spaCy and Stanza

Feature	spaCy	Stanford NLP (Stanza)
Speed	Fastest	Slower
Accuracy	Good, but may miss some entities	More accurate on complex texts
Ease of Use	Simple API	More setup required
Linguistic Depth	Focuses on ML-based models	Uses rich linguistic features
Multi-language Support	Supports multiple languages	Stronger multilingual capabilities
Customization	Requires retraining)	Supports rule-based modifications
Model Type	CNN-based	BiLSTM-CRF-based

Transformer-Based NER Models

Transformer Foundations

Transformers revolutionized NLP by replacing recurrence with **self-attention**, enabling models to process entire sequences simultaneously. Key concepts include:



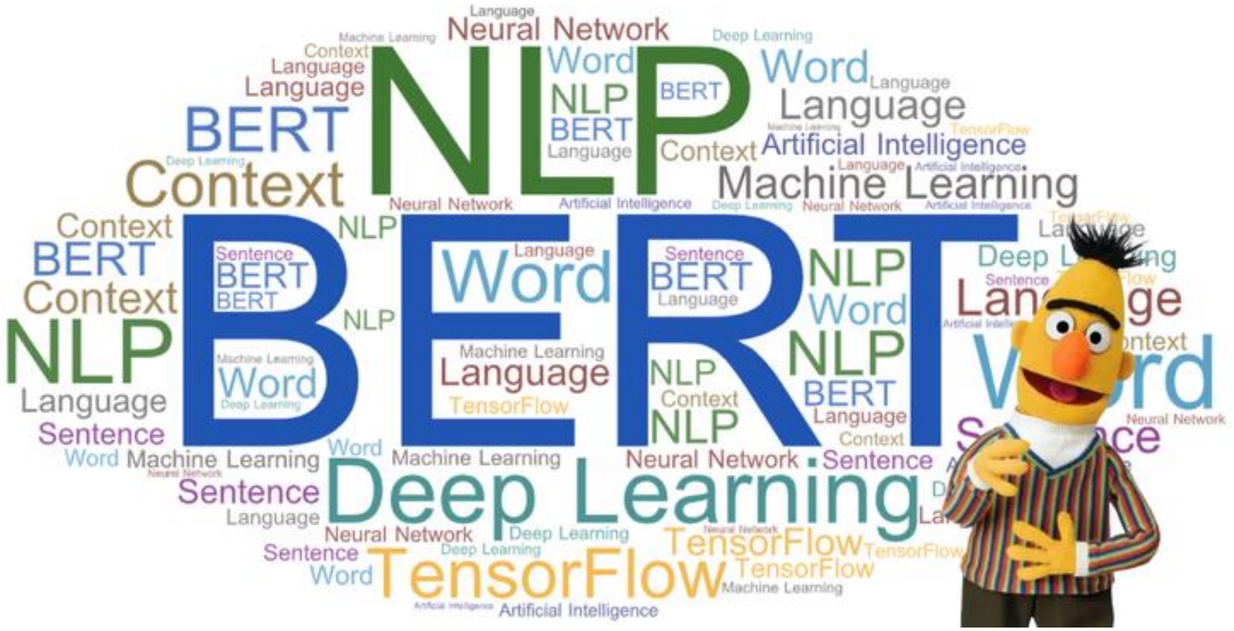
Self-Attention Mechanism

Positional Encoding

Pre-training and Fine-tuning Paradigm

BERT-NER

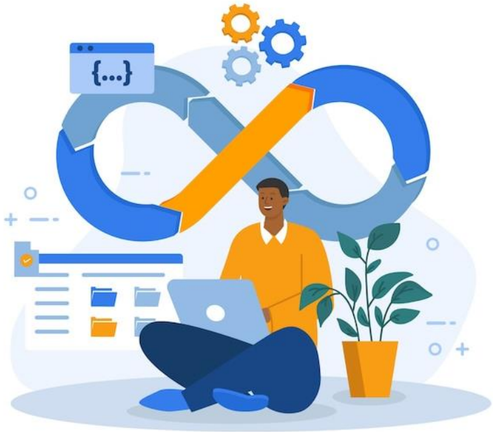
BERT-NER applies the BERT model to the task of NER. It leverages the rich contextual representations from BERT to classify each token into entity categories. In practice, a classification head (typically a simple feed-forward layer) is added on top of BERT to predict the entity label for each token.



Advantages of BERT-NER

Contextual Understanding: BERT's bidirectional context helps disambiguate entities in complex sentences.

Transfer Learning: Pre-trained on vast amounts of data, BERT provides robust features that adapt well to the NER task.



Flexibility: Fine-tuning allows the same model architecture to be applied across various languages and domains.

RoBERTa-Based Approaches

RoBERTa (Robustly Optimized BERT Pretraining Approach) refines BERT's training process by:

- Using larger mini-batches
- Training on more data
- Removing the next sentence prediction task
- Tuning other hyperparameters

When applied to NER, RoBERTa-based models follow a similar architecture to BERT-NER with token-level classification but often deliver **better performance** due to more **robust pre-training.**

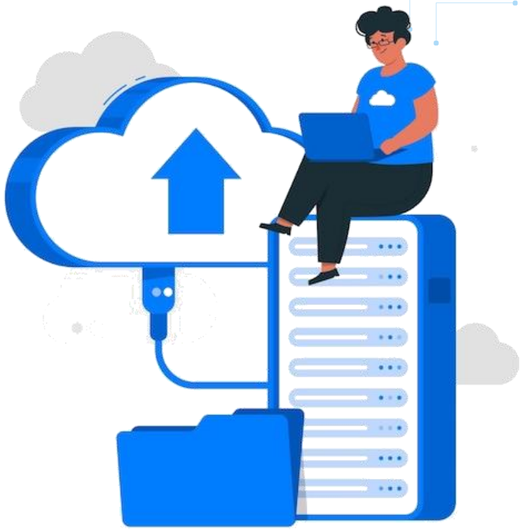
Advantages of RoBERTa

Improved Training Dynamics

Better Performance

Robustness

Comparative Analysis



Performance:

- Accuracy and F1 Scores:
- Generalization

Training and Fine-Tuning:

- BERT-NER
- RoBERTa-Based

Deployment:

- Hardware Requirements
- Resource Considerations

Fine-Tuning Transformers for NER

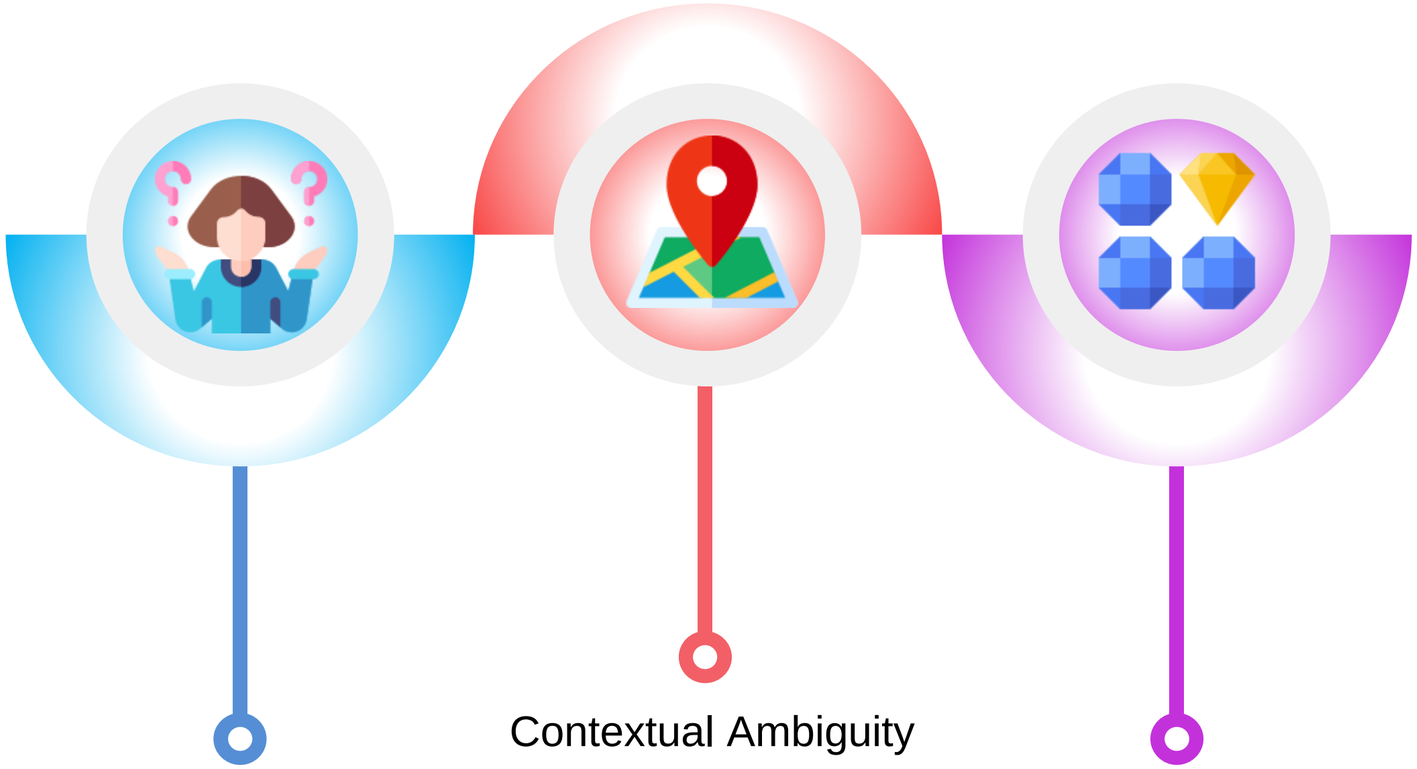
Ambiguity in NER

Ambiguity occurs when a word or phrase can belong to multiple entity types depending on context.

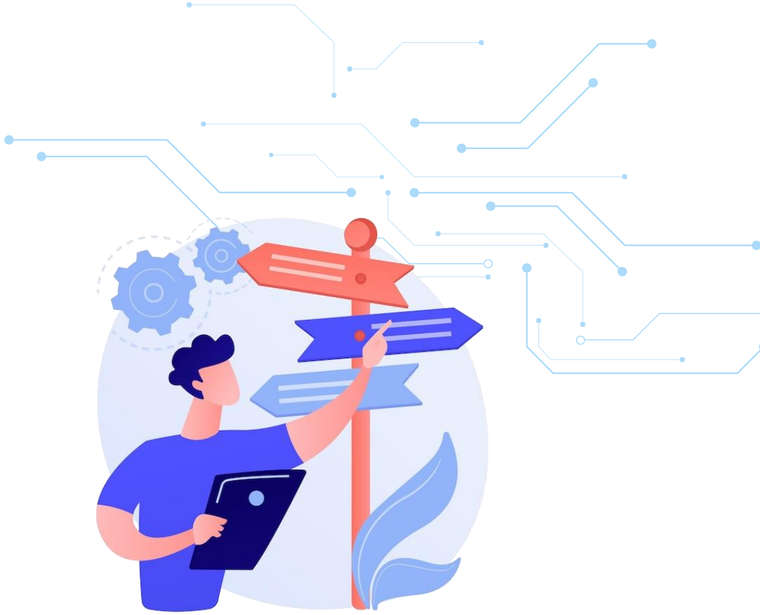
Organization

Fruit

Types of Ambiguity



How to Handle Ambiguity?



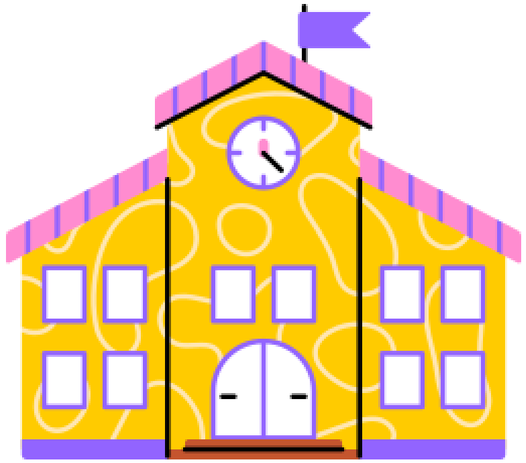
Context-aware models: Use transformers like BERT to understand words in context.

Entity linking: Cross-check entities with a knowledge base (e.g., Wikipedia).

Post-processing rules: Manually adjust outputs using domain-specific rules.

Overlapping Entities in NER

Overlapping entities occur when a single word or phrase belongs to **multiple entity categories at the same time**.

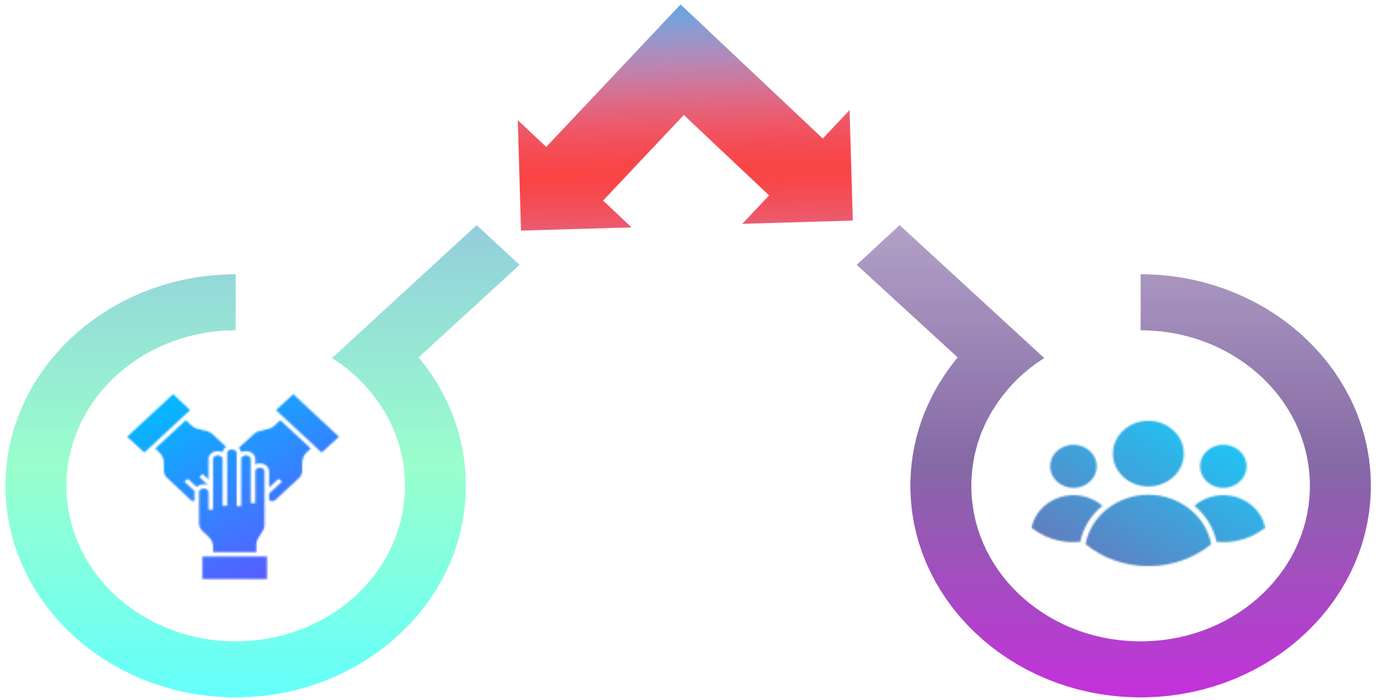


"Harvard University in Cambridge"

Harvard University- **Organization**

Cambridge- **Location**

Types of Overlapping Entities

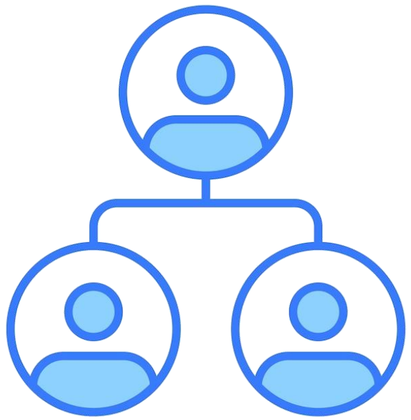


Nested Entities

Multi-Label Entities

How to Handle Overlapping Entities?

Span-based NER models: Instead of labeling single tokens, these models detect all possible spans.



Multi-label classification: Allows an entity to belong to multiple categories.

Hybrid approaches: Combining rule-based and deep learning models for better precision.

Parsing Algorithms: Earley, CYK

Parsing Algorithms

Parsing algorithms are essential for transforming sentences into structured representations that machines can work with. They are fundamental in building parsers that can handle the complexities and ambiguities of natural language.

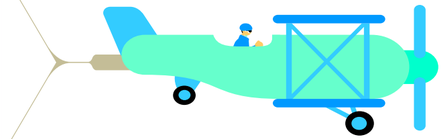
Earley Algorithm

The Earley algorithm is a dynamic programming method capable of parsing any context-free grammar.

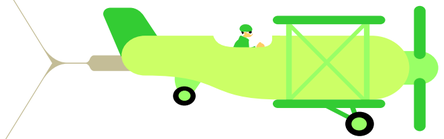
CYK Algorithm

The Cocke–Younger–Kasami (CYK) algorithm is a bottom-up parser that requires the input grammar to be in Chomsky Normal Form (CNF).

It uses a table-filling approach where each cell represents possible non-terminals for subsequences of the sentence.



The algorithm systematically builds up the parse by combining smaller parts into larger constituents.



Comparative Analysis

Flexibility vs. Structure

- Earley: Works with any context-free grammar.
- CYK: Requires grammar in Chomsky Normal Form (CNF).

Efficiency Considerations

- Performance depends on grammar and sentence complexity.

Use Case

Earley Algorithm:

- Handles diverse sentence structures.
- Manages ambiguous grammars effectively.
- Works with standard context-free grammars.

CYK Algorithm:

- Best with grammars in Chomsky Normal Form (CNF).
- Provides clear yes/no parsing results.
- Efficient due to its structured approach with CNF.

Dependency Parsing with spaCy & StanfordNLP

What is Dependency Parsing?

- The root represents the main verb or central word.
- Nodes represent words, and edges denote grammatical relationships (e.g., subject, object).

"The cat sits on the mat."

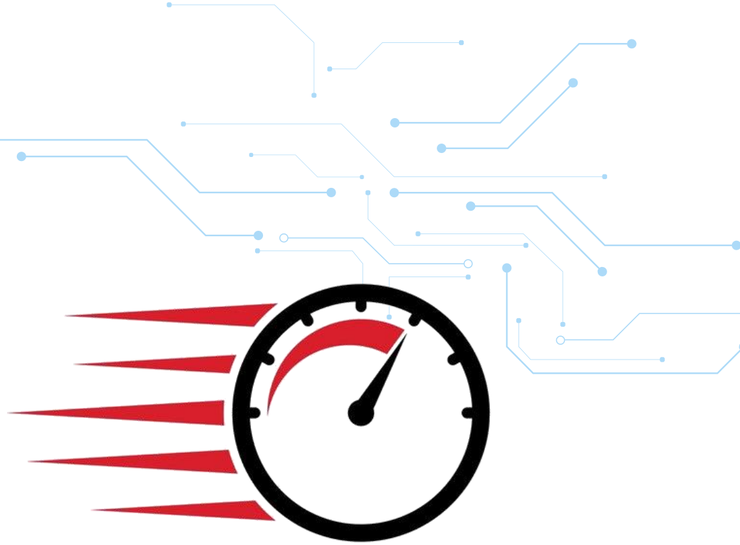
Dependency parsing identifies:

"cat" → "sits" (subject)

"sits" → "on" (preposition)

"on" → "mat" (object)

SpaCy for Dependency Parsing



SpaCy is highly optimized for **speed and is production-ready**, providing pre-trained models that support multiple languages.

It has an **intuitive API** that allows quick integration into NLP pipelines.

The process typically involves **tokenizing text, performing part-of-speech** (POS) tagging, and then extracting dependency relationships.

Stanford NLP for Dependency Parsing

Processes text by annotating it with rich **linguistic information** before constructing a dependency tree.

It supports a **wide range of languages** and is often favored in research contexts.

Renowned for its comprehensive **linguistic annotations** and detailed dependency graphs.

Building a Syntax Tree in Python

Syntax Trees

A syntax tree is a graphical representation of the grammatical structure of a sentence. It visually breaks down a sentence into its component parts, showing how individual words and phrases connect to form a coherent structure.

NLTK

SpaCy

Syntax Trees (Contd.)

Sentence: The cat quickly chased a mouse.

Word Categories:

- Nouns (N): cat, mouse
- Verbs (V): chased
- Adverb (Adv): quickly
- Articles (Det): The, a

Construction Process

01

Tokenization and POS Tagging

02

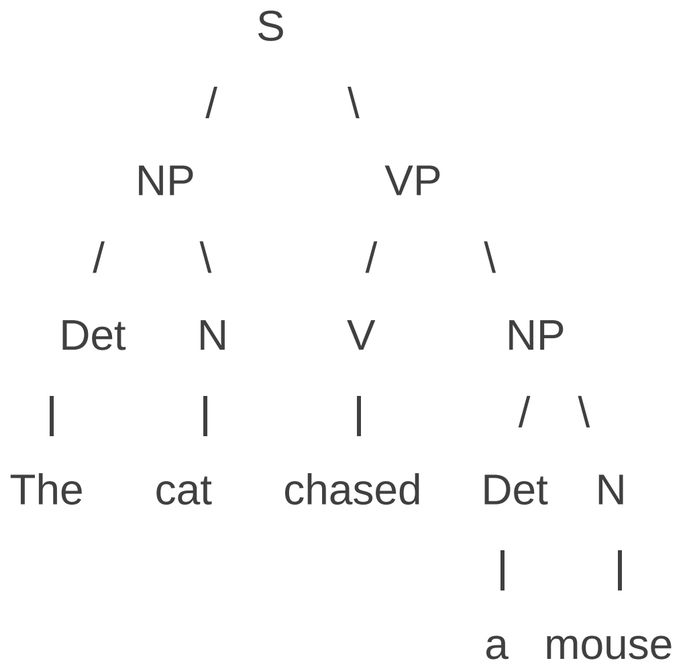
Parsing the Sentence

03

Visualization

Structure Breakdown and Tree

- **S** (Sentence)
- **NP** (Noun Phrase) → Det + N → The cat
- **VP** (Verb Phrase) → V + Adv + NP → chased quickly a mouse
- **NP** (Noun Phrase) → Det + N → a mouse



Applications

01

Used for detailed grammatical analysis, building language models, and educational demonstrations in linguistics.

Customize grammars to handle more complex sentences or different languages.

02

03

Integrate syntax trees with machine learning models for advanced NLP tasks.

Relation Extraction Techniques

What is Relation Extraction?

Relation Extraction identifies and classifies semantic relationships between entities in a given text.

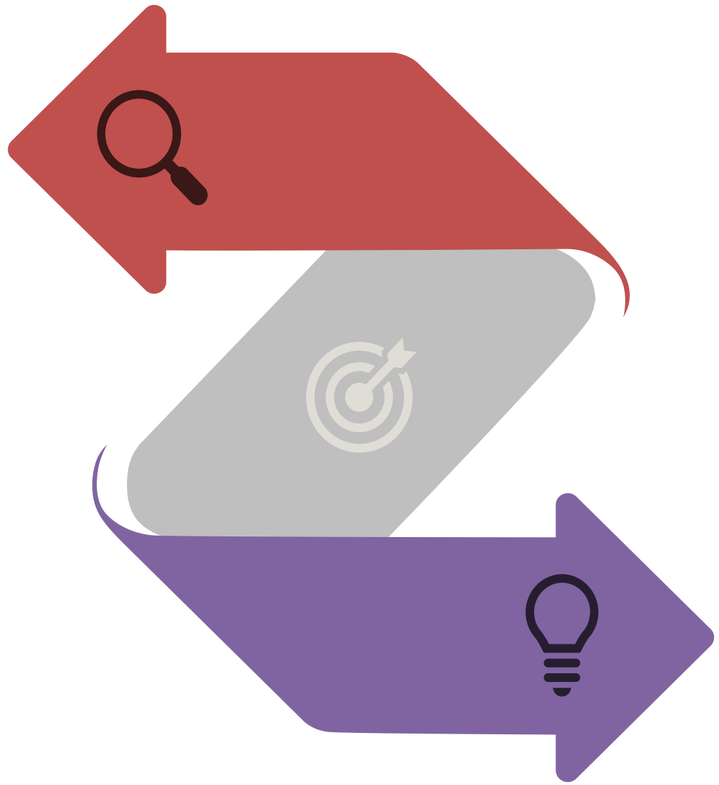
Text: *"Steve Jobs founded Apple."*

Relation: **Founder-Of**(Steve Jobs, Apple)

Types of Relations

Binary Relations

Relationships involving two entities



N-ary Relations

More complex relationships involving multiple entities

Relation Extraction Techniques

Relation Extraction (RE) in NLP is a key task in **Information Extraction (IE) and Text Mining**, aiming to identify and categorize **relationships between entities** in text. It plays a crucial role in converting unstructured text into structured knowledge.

Rule and Pattern Based Methods

Hand-crafted patterns

Based on syntactic or lexical patterns.

Example:"X founded Y" → Relation = Founder-Of

Dependency Path Patterns

Use parse trees to find patterns.

Example: if the dependency path between entities contains *nsubj* and *dobj* through the verb f*ounded*, infer *Founder-Of*.

Supervised Machine Learning Methods

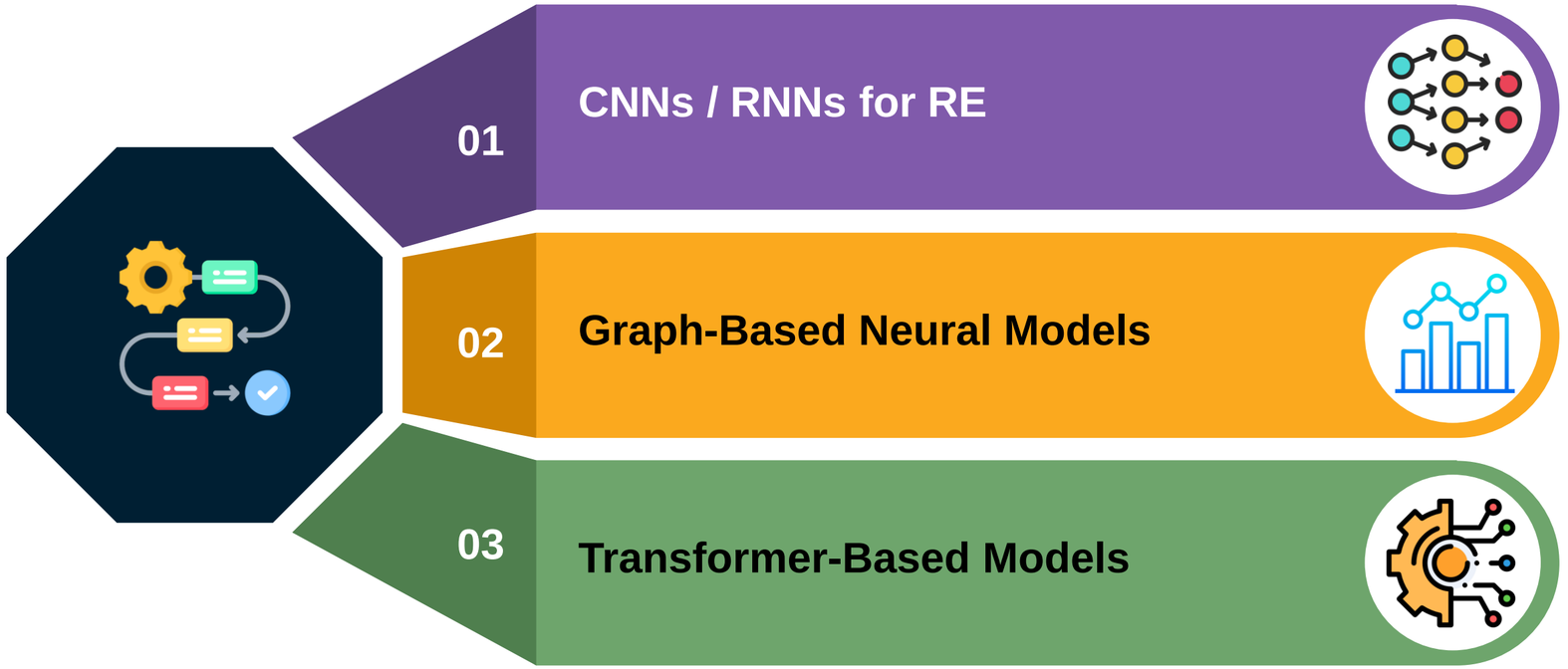
01 Named Entity Recognition (NER)

02 Candidate entity pair generation

03 Feature extraction (see previous answer)

04 Train classifier (SVM, Logistic Regression, etc.)

Neural Network Based Methods



Open Information Extraction (OpenIE)

Open Information Extraction (OIE) is a Natural Language Processing (NLP) task that aims to extract structured, machine-readable information, usually in the form of triples (subject, relation, object), from unstructured text, without relying on pre-defined relation schemas

- Extracts relations without predefined schemas using:
- Rule-based heuristics
- Deep learning approaches

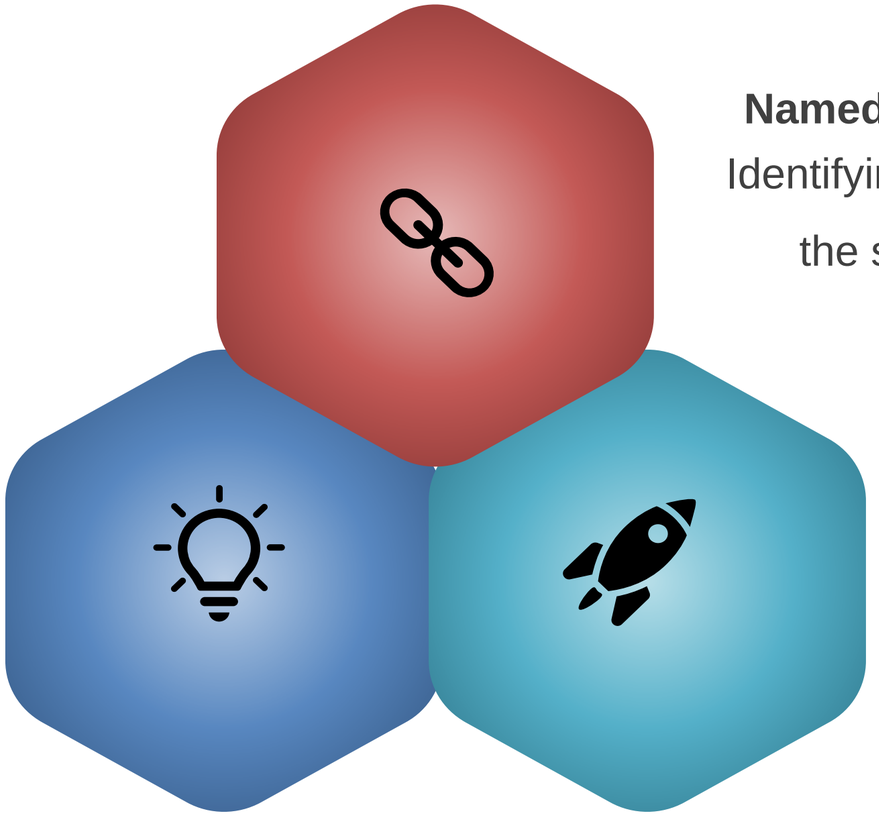
Coreference Resolution (Tracking Entities in Text)

Types of Coreference

1

Pronominal Coreference

Resolving pronouns



2

Named Entity Coreference

Identifying multiple mentions of the same named entity.

3

Bridging Coreference

Linking related entities

Rule-Based Coreference Resolution

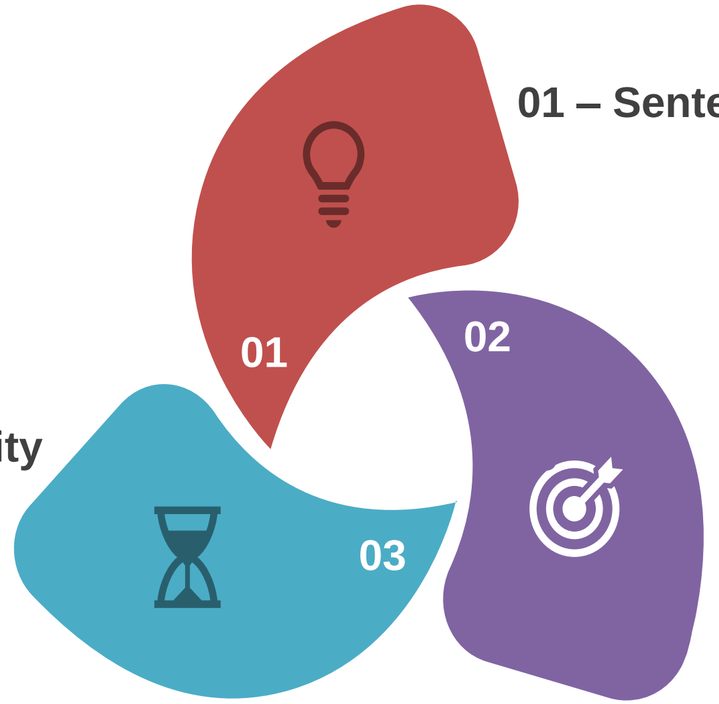
Heuristics-Based Methods

Use syntactic and proximity-based rules.

Syntactic Rules

Consider grammatical features such as gender and number agreement.

Machine Learning Approaches



01 – Sentence distance

02 – Dependency relations

03 - Lexical similarity

Challenges in Coreference Resolution

Ambiguity and Context

- Pronouns can refer to multiple entities, and resolving them often requires understanding the broader context.

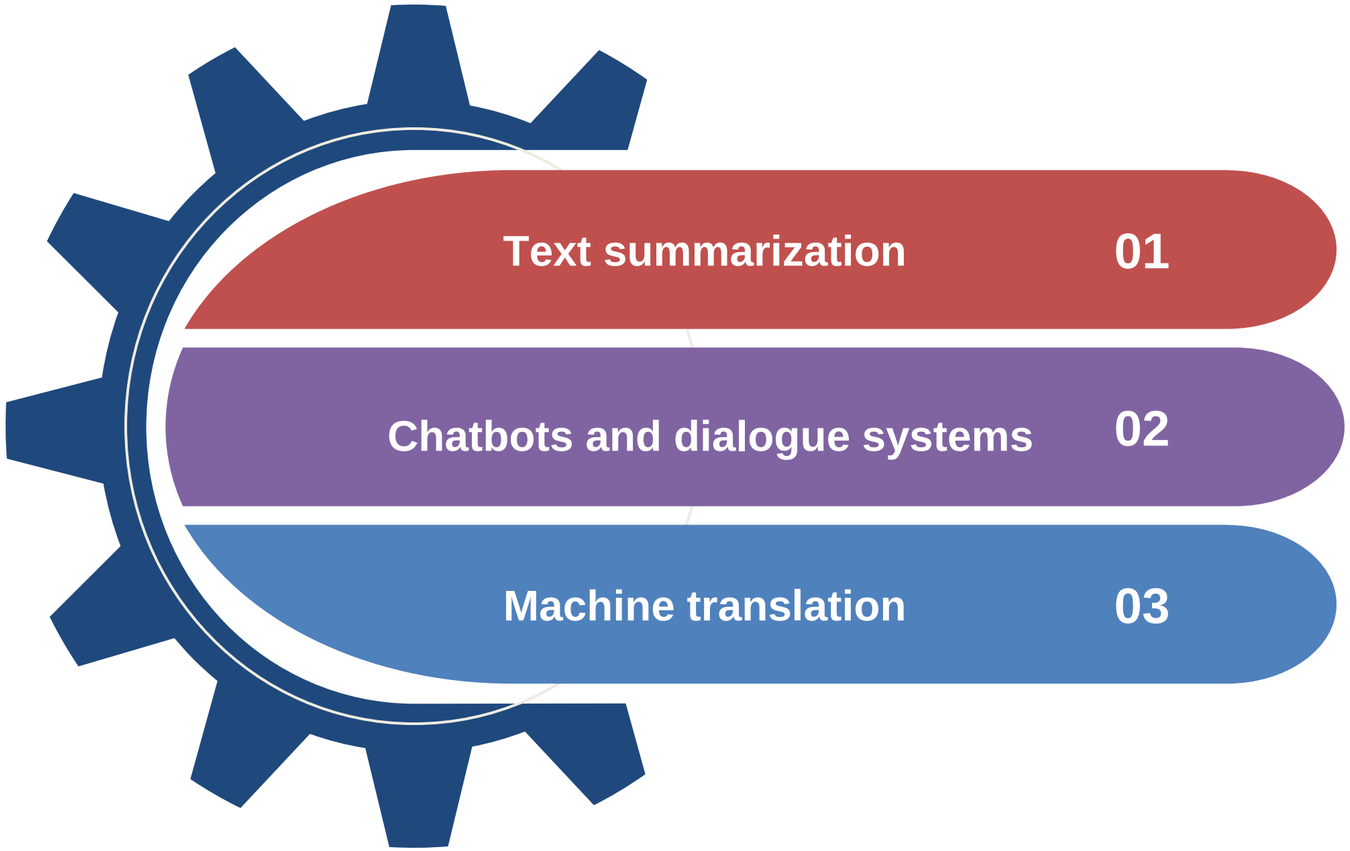
Linguistic Complexity

- Factors such as gender, number, and semantic context complicate the task.

Lack of annotated data

- Manual annotation is time-consuming and expensive, impacting the scalability of solutions.

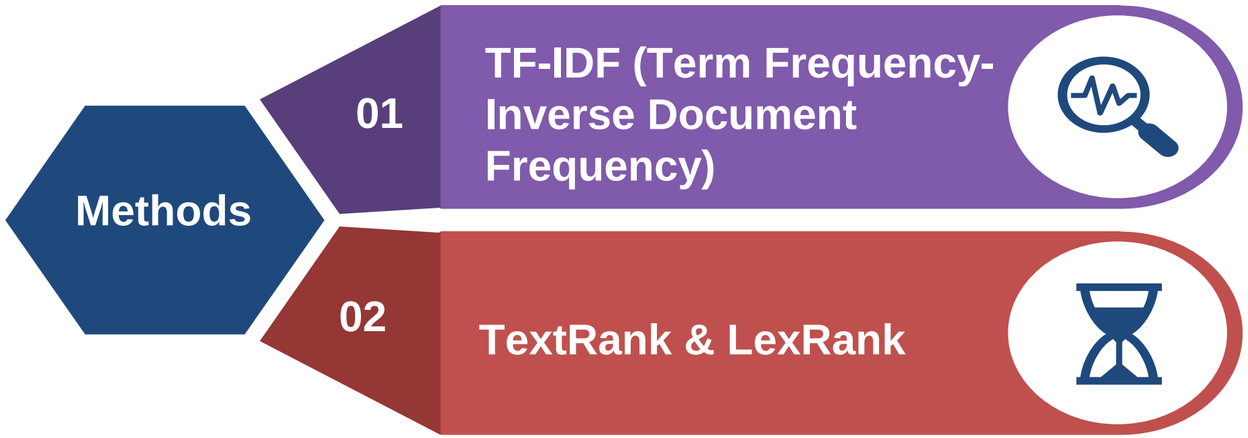
Applications of Coreference Resolution



Text Summarization: Extractive & Abstractive

Extractive Summarization Methods

Extractive summarization selects key sentences directly from the original text based on certain criteria. It does not generate new sentences but instead picks the most important ones.



Abstractive Summarization

Generates new sentences while **preserving meaning**.

Uses sequence-to-sequence models and transformers.

Challenges:

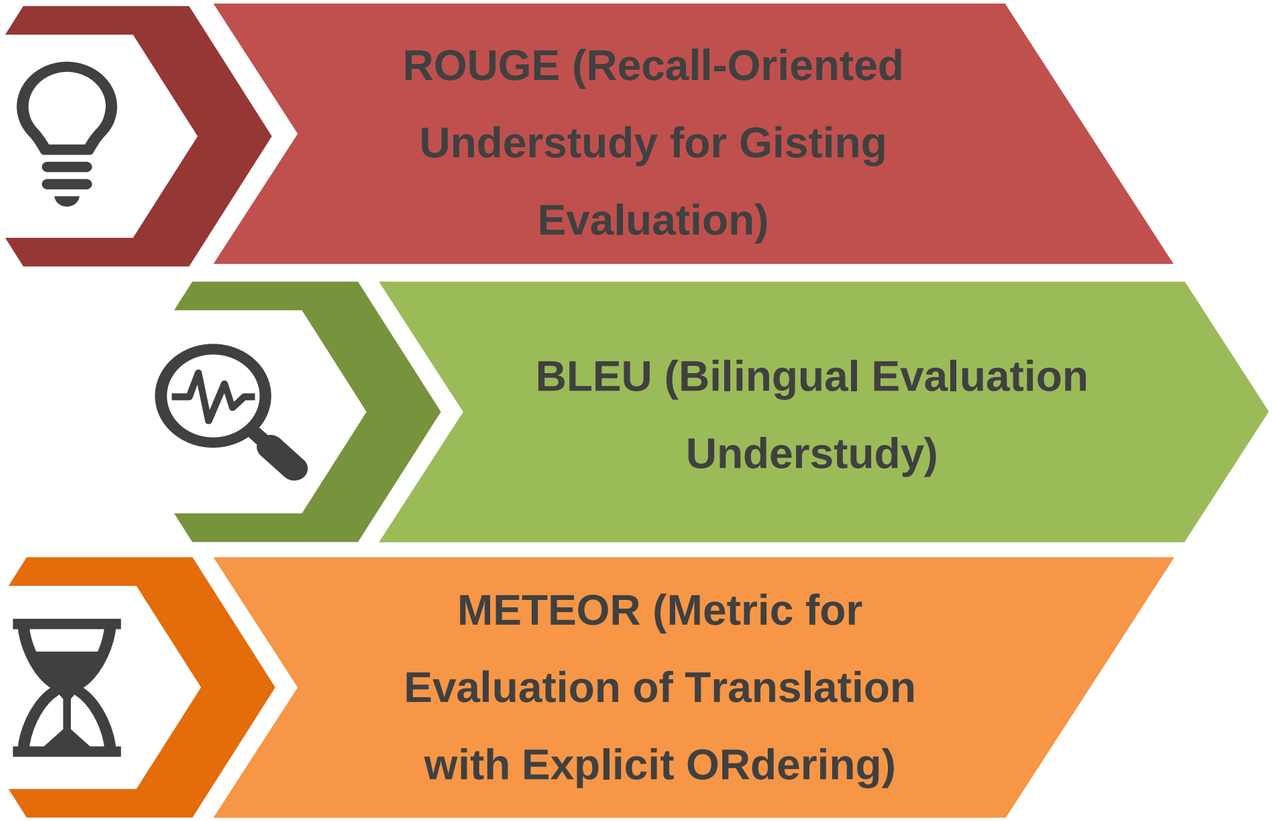
- Coherence
- Factual consistency
- Semantic understanding

Rule-Based Approaches for Summarization

Uses manually defined heuristics, such as:

- Selecting sentences based on their position (e.g., first and last sentences of paragraphs).
- Extracting sentences with high-frequency words or domain-specific terms.

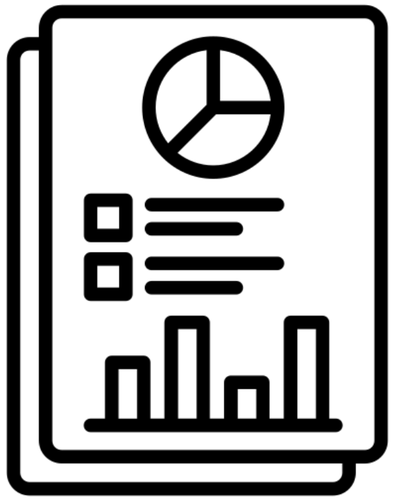
Evaluation Metrics for Summarization



Applications of Text Summarization

News Aggregation

Legal and Medical Document Summarization



Research Paper Summarization

Constituency and Dependency Parsing (Demonstration)

Note: Refer to Module 3: Demo 1 on LMS for detailed steps.

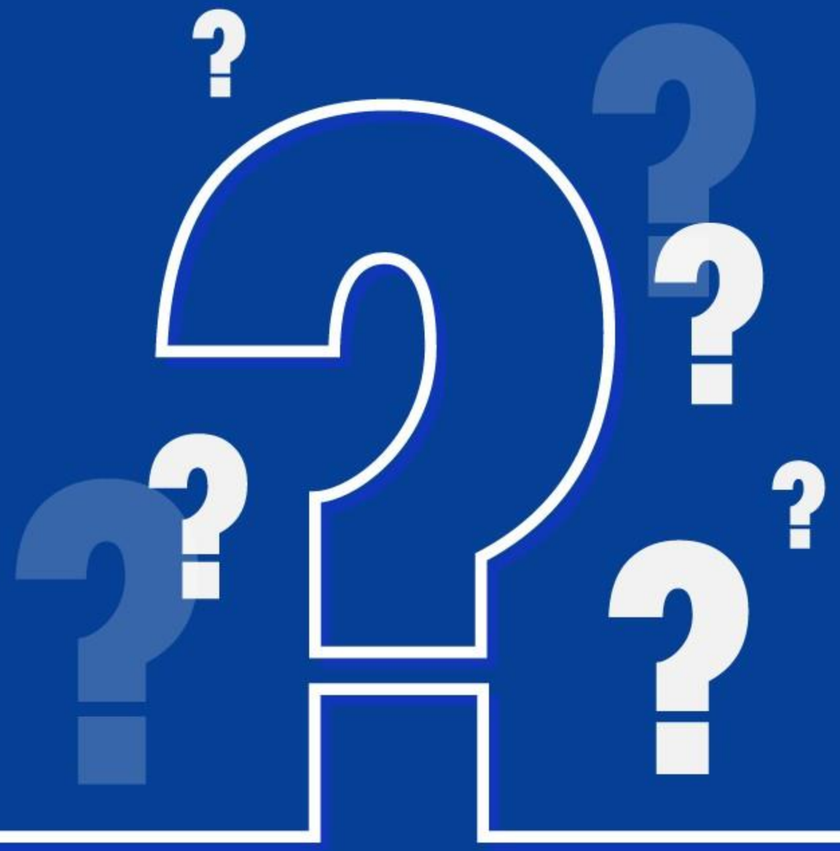
Summary

In this lesson, you have learned to:

- Explain the fundamental concepts and practical uses of Named Entity Recognition in NLP.
- Explore the effectiveness of pretrained and transformer-based models for entity recognition.
- Identify methods to address ambiguity and overlapping entities within NER tasks.
- Apply parsing techniques and construct syntax trees for sentence structure analysis.
- Describe the role of relation extraction, coreference resolution, and summarization in understanding text.

edureka! a Veranda ENTERPRISE

Questions



edureka! a Veranda ENTERPRISE

Feedback

edureka! a Veranda ENTERPRISE

Thank You

For information, Please Visit our Website
www.edureka.co