Natural Language

In neuropsychology, linguistics, and the philosophy of language, a natural language or ordinary language is any language that has evolved naturally in humans through use and repetition without conscious planning or premeditation.

In contrast to artificial languages like Python, C, Java, etc., natural languages like English, Portuguese, French, etc., have evolved over time and use and it's difficult to express them in strict formal rules.

Definition of NLP

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) and computational linguistics that focuses on the interaction between computers and humans through natural language. The goal of NLP is to enable machines to read, understand, and derive meaning from human language in a way that is both syntactically and semantically correct.

> Formal Definition: "Natural Language Processing is a computational technique for the automatic analysis and representation of human language to perform useful tasks such as translation, summarization, sentiment analysis, and question answering."

NLP combines linguistics (syntax, semantics, pragmatics), machine learning, and deep learning to enable various tasks involving text and speech.

Understanding natural language is informally considered as AI-hard/AI-complete by several researchers.

Why is NLP hard?

NLP is hard because natural languages evolved without a standard rule or logic. They were developed in response to the evolution of the human brain — in its ability to understand signs, voice, and memory. With NLP, we are now trying to "discover rules" for something (language) that evolved without rules.

Let us now try to understand why NLP is considered hard using a few examples:

- **"There was not a single man at the party"**
  — Does it mean that there were no men at the party?
  — Or that there was no one at the party?
  — Does "man" refer to gender or mankind?
- **"The chicken is ready to eat"**
  — Is the bird ready to eat food?
  — Or is the chicken cooked and ready to be eaten?
- **"Google is a great company." vs. "Google this word and find its meaning."**
  — "Google" is a noun in the first and a verb in the second.
- **"The man saw a girl with a telescope."**
  — Did the man use a telescope to see the girl?
  — Or did the girl have the telescope?

Natural language is full of ambiguities. Ambiguity refers to the potential for multiple interpretations. This, along with extracting knowledge from unstructured data, makes NLP a hard problem.

Applications of NLP

NLP is a foundational component in many modern-day applications across industries. Below are key areas where NLP plays a central role:

- **Information Retrieval (IR)**
  • Search engines like Google use NLP techniques to understand queries and rank results.
  • Query expansion, spell correction, and relevance prediction rely on language models.
- **Machine Translation**
  • Converting text from one language to another using systems like Google Translate or DeepL.
  • Advanced systems use sequence-to-sequence models and transformers.
- **Chatbots and Virtual Assistants**
  • NLP powers voice-based assistants like Siri, Alexa, and Google Assistant.
  • Handles natural language understanding (NLU) and generation (NLG).
- **Text Classification**
  • Assigns labels to text (e.g., spam vs. non-spam, positive vs. negative sentiment).
  • Used in email filtering, product reviews, and social media monitoring.
- **Named Entity Recognition (NER)**
  • Identifies entities such as names, dates, organizations in text.
  • Used in financial analytics, legal tech, and medical record analysis.
- **Speech Technologies**
  • Speech-to-Text (STT): Converts spoken language into text.
  • Text-to-Speech (TTS): Synthesizes speech from written text.
- **Text Summarization**
  • Reduces long documents into concise summaries.
  • Used in news apps, legal document summarization, and academic research tools.

- **Question Answering**
  • Systems like ChatGPT, BERT-based models can answer open-domain or specific questions.
  • Applied in search, virtual tutors, and helpdesk automation.

Challenges in NLP

While NLP has seen significant progress, it still faces numerous challenges due to the complexities of human language:

a. Ambiguity

1. Lexical Ambiguity: Occurs when a word has multiple meanings depending on the context.
Examples:

- “Bat” → Could refer to a flying mammal or a cricket bat.
- “Bank” → Can imply a financial institution or the side of a river.

2. Syntactic Ambiguity: Arises when a sentence's structure allows for multiple grammatical interpretations.
Example:

- “He saw the man with the telescope.”
  — He used a telescope to see the man.
  — He saw a man who was holding a telescope.

3. Semantic Ambiguity: Pertains to multiple interpretations of sentence meaning even when grammar is clear.
Example:

- “Visiting relatives can be boring.”
  — Relatives who visit are boring?
  — The act of visiting relatives is boring?

Solutions: Disambiguation techniques using contextual embeddings (e.g., BERT), sense disambiguation models, and semantic parsing aim to resolve such ambiguities.

b. Context Understanding

Human language is profoundly context-sensitive. Successful language understanding requires:

- World Knowledge: Understanding common-sense facts and cultural references.
- Co-reference Resolution: Linking pronouns to entities, e.g., in "John gave Mark his book", whose book is it?
- Sarcasm and Irony Detection: Recognizing when the literal meaning is opposite to the intended meaning.

- Discourse Analysis: Understanding relations between sentences to capture narrative flow or argument structure.

Solutions: Incorporation of knowledge graphs (e.g., ConceptNet), discourse-aware transformers, and pragmatic analysis models.

c. Language Diversity

The global linguistic landscape is highly diverse:

- Syntax and Grammar Variance: Languages differ in sentence construction, such as SVO (English) vs. SOV (Japanese).
- Writing Systems: Varieties include alphabetic (English), logographic (Chinese), and abjad (Arabic) scripts.
- Morphologically Rich Languages: Languages like Finnish, Turkish, and Hungarian exhibit complex word inflections.

Solutions: Multilingual models (XLM-R, mBERT), language-specific tokenizers.

d. Informal and Code-Mixed Text

Modern communication, especially on social media, is:

- Informal: Slang, abbreviations, typos, emojis.
- Code-Mixed: Mixing multiple languages, e.g., “I’ll call you after puja at the mandir.”

Normalization Techniques: Slang & emoji preprocessing, transliteration, data augmentation, transfer learning.

e. Low-Resource Languages

Many languages lack annotated corpora and linguistic tools.

Solutions: Transfer learning, zero-shot/few-shot learning, initiatives like Meta’s NLLB.

f. Bias and Fairness

NLP models can inherit gender, racial, or cultural biases from training data.

- Gender Bias: "doctor - he", "nurse - she".
- Racial/Cultural Bias: Stereotypical or offensive outputs.

Ethical Imperatives: Debiasing techniques, audits, fairness metrics, ethical AI frameworks.

g. Computational Requirements

Deep NLP models require:

- High computational power (GPUs/TPUs)
- Energy-intensive training
- Low-latency inference needs

Solutions: Model compression, knowledge distillation (e.g., DistilBERT), efficient transformers (Longformer, TinyBERT).

Challenge Summary Table

Challenge	Impact	Research Directions
Ambiguity	Multiple meanings impede clarity	Contextual embeddings, WSD models
Context Understanding	Requires deep comprehension	Discourse models, Co-reference tools
Language Diversity	Complexity in syntax & morphology	Multilingual models, tokenization
Informal Text	Non-standard language processing	Normalization, code-mixing handling
Low-Resource Languages	Lack of tools for many languages	Transfer & Few-shot learning
Bias & Fairness	Ethical risks in outputs	Debiasing, fairness constraints
Computational Costs	Resource-heavy models	Efficient modeling, distillation

Summary

Component	Description
Definition	NLP is the study of enabling machines to process and understand human language.
Applications	Encompasses MT, chatbots, sentiment analysis, IR, summarization, etc.
Challenges	Includes ambiguity, context, multilingualism, data scarcity, bias, and compute costs

NLP Pipeline

Before we start developing the applications let us review the sequence of steps that we typically go through while developing NLP applications.

The NLP Pipeline refers to a systematic sequence of processes through which raw, unstructured textual data is transformed into actionable insights, predictions, or automated responses using computational models. Each stage of the pipeline plays a crucial role in ensuring that the final model or system is both accurate and efficient.

Collecting Textual Data

- **Sources of Data:**
  - Web scraping (blogs, news articles, forums)
  - Social media platforms (tweets, posts)
  - Internal organizational data (emails, customer feedback, chat logs)
  - Public datasets (corpora, linguistic datasets like Wikipedia, SQuAD, etc.)
- **Importance:**
  - The quantity and quality of data directly affect the model's performance.
  - Data relevance must align with the domain-specific requirements.

Cleaning the Input Text Data

We use various techniques to clean the data, depending on the domain of the problem. Regular expressions are extensively used at this stage in identifying anomalies in the data and transforming them as required.

- Typos, HTML tags
- Special symbols, URLs
- Irregularities like non-UTF-8 characters

Techniques Used:

- Regular Expressions (regex)
- Noise Removal
- Lowercasing
- Handling missing data

Example:

```
Raw: "Congrats!!! You won $1000. Click http://spamlink.com now!! 🎉🎉"
Cleaned: "congrats you won 1000 click now"
```

Normalizing the Data

- Stopword Removal

- Stemming
- Lemmatization
- Case Normalization

Purpose: These steps reduce the dimensionality of data and help focus on meaningful terms that contribute to model learning.

Feature Engineering

Feature engineering textual data typically involves converting the cleaned and normalized data into vector representations.

- Bag of Words (BoW)
- TF-IDF
- Word Embeddings (Word2Vec, GloVe, FastText)
- Contextual Embeddings (BERT)

Apply Machine Learning Techniques to Build Models

Task	Suitable Models/Techniques
Text Classification	Naive Bayes, Decision Trees, SVM, Random Forest
Sequence Labeling	HMMs, CRFs, LSTMs, BiLSTM
Language Modeling	N-grams, RNNs, Transformers (GPT, BERT)
Sentiment Analysis	Logistic Regression, Neural Networks
Text Generation	LSTMs, GPT models

Validating the Model Built

- Train-Test Split
- Cross-Validation
- Metrics: Accuracy, Precision, Recall, F1-Score, ROC-AUC, Confusion Matrix

Deploying the Model and Making Predictions

- APIs for prediction
- Integrated systems (chatbots, recommendation engines)

Monitoring: Watch for performance degradation, bias, and fairness post-deployment.

Summary of NLP Pipeline

Step	Purpose
Collecting Data	Source relevant text data
Cleaning Text	Remove noise and irrelevant elements
Normalization	Standardize text (stemming, stopword removal)
Feature Engineering	Convert text to numerical format
ML Model Building	Train using appropriate algorithms
Model Validation	Evaluate model on unseen data
Deployment & Inference	Make predictions in real-time systems

Linguistic Essentials: Syntax, Semantics, Pragmatics

Understanding language from a linguistic perspective is essential for designing NLP systems that can effectively process, interpret, and generate human-like language. Linguistics offers structured frameworks to model language across three primary layers:

1. Syntax – The structure and arrangement of words in sentences
2. Semantics – The meaning conveyed by words, phrases, and sentences
3. Pragmatics – The influence of context, intent, and social norms on meaning

Syntax – Structure of Language

Syntax refers to the set of rules, principles, and processes that govern the structure of sentences in a given language. It determines how words combine to form grammatically correct sentences.

- Helps parse and validate sentence structure
- Enables syntax trees or dependency parsing
- Parsing: Analyzing the grammatical structure of sentences.
- Understanding sentence components: Such as subject, verb, object.
- Supporting tasks like: Machine translation, grammar checking, and information extraction.

Syntactic Representations:

- Constituency Parsing: Breaks sentences into nested sub-phrases.
- Dependency Parsing: Captures relationships between "head" words and their dependents.

Example: Sentence: “The cat sat on the mat.” → [NP The cat] [VP sat [PP on [NP the mat]]]

NLP Tasks Involving Syntax:

- Part-of-Speech (POS) Tagging
- Constituency Parsing
- Dependency Parsing
- Grammar Correction Tools (e.g., Grammarly)

Semantics – Meaning of Language

Semantics is the study of meaning in language—how individual words, phrases, and sentences convey meaning.

- Used to build systems that understand, infer, and represent meaning
- Helps resolve ambiguity and build word representations

Example: “The bank is closed today.” – Semantic disambiguation helps determine if “bank” is a financial institution or a riverbank.

Key NLP Tasks Involving Semantics:

- Word Sense Disambiguation (WSD)
- Named Entity Recognition (NER)
- Semantic Role Labeling (SRL)
- Word Embeddings (Word2Vec, GloVe, FastText)
- Textual Entailment
- Semantic Parsing

Pragmatics – Context of Language

Pragmatics deals with the contextual and situational meaning of language. It explains how the interpretation of sentences can change based on speaker intent, cultural norms, and context.

- Important in dialogue systems, chatbots, and co-reference resolution
- Helps in understanding indirect speech acts, sarcasm, and implicature

Example: “Can you pass the salt?” – Syntactically a question; pragmatically a request.

NLP Tasks Involving Pragmatics:

- Dialogue and Conversational Modeling
- Co-reference Resolution
- Sentiment and Sarcasm Detection
- Contextual Question Answering

Comparative Summary

Aspect	Syntax	Semantics	Pragmatics
Focus	Sentence structure	Meaning of words/sentences	Meaning in context
Example	Grammar rules (NP → Det + Noun)	“Bank” = financial institution?	“Can you pass the salt?” = Request
NLP Use	POS tagging, Parsing	NER, Word Embeddings, SRL	Dialogue systems, Sentiment, QA

Text Processing in NLP

Text is a collection of meaningful sentences. Each sentence in turn comprises many words.

Consider the text "India is a republic nation. We are proud Indians". This text contains 2 sentences and 9 words.

```
import nltk
from nltk import *
text = "India is a republic country. We are proud Indians"
print(nltk.word_tokenize(text))
# Output: ['India', 'is', 'a', 'republic', 'country', '.', 'We', 'are', 'proud', 'Indians']

# Vocabulary
tokens = nltk.word_tokenize(text)
vocab = sorted(set(tokens))
print(vocab)
# Output: ['.', 'India', 'Indians', 'We', 'a', 'are', 'country', 'is', 'proud', 'republic']

# Punctuation
from string import punctuation
vocab_wo_punct = [i for i in vocab if i not in punctuation]
print(vocab_wo_punct)
# Output: ['India', 'Indians', 'We', 'a', 'are', 'country', 'is', 'proud', 'republic']
```

1. Tokenization

Tokenization is the process of splitting text into smaller units, known as tokens. These tokens may be words, subwords, or sentences.

Example:
Input: "NLP is fun!"
Output Tokens: ["NLP", "is", "fun", "!"]

Use in NLP:

- Input for all downstream tasks
- Helps in vocabulary creation
- Used in both traditional models and deep learning pipelines

2. Lemmatization

Lemmatization reduces words to their base or dictionary form (lemma) using morphological analysis.

Example:
Words: “running”, “ran”, “runs” → Lemma: “run”

Use in NLP:

- Preserves word meaning
- Important for semantic analysis
- Used in search engines, question answering

```
from nltk.stem.wordnet import WordNetLemmatizer
lemmaObj = WordNetLemmatizer()
print(lemmaObj.lemmatize("went", pos='v'))
# Output: go
```

3. Stemming

Stemming cuts off affixes from words to get the root form. It’s a heuristic-based approach and often less accurate than lemmatization.

Example:
“connected”, “connecting”, “connection” → Stem: “connect”

Use in NLP:

- Faster preprocessing
- Useful in IR systems and early search algorithms
- May produce non-dictionary words (e.g., “studies” → “studi”)

```
from nltk.stem.snowball import SnowballStemmer
stemObj = SnowballStemmer("english")
stemObj.stem("Studying") #Prints 'studi'

stemmed_vocab=[]
stemObj = SnowballStemmer("english")
for i in vocab_wo_punct:
    stemmed_vocab.append(stemObj.stem(i))
print(stemmed_vocab) #Prints ['india', 'indian', 'we', 'a', 'are', 'countri', 'is', 'proud', 'republ']
```

4. Stop Word Removal

Stop words are commonly used words (e.g., “the”, “is”, “and”) that carry little semantic value and are often removed from text.

Example:
“The sun rises in the east.” → [“sun”, “rises”, “east”]

Use in NLP:

- Reduces dimensionality
- Improves performance in tasks like classification and clustering

NLTK provides a pre-defined set of stopwords for English, as shown

```
from nltk.corpus import stopwords
wo_stop_words = []
stop_words_set = set(stopwords.words("english"))
for i in vocab_wo_punct:
    if i not in stop_words_set:
        wo_stop_words.append(i)
print(wo_stop_words) #Prints ['India', 'Indians', 'We', 'country', 'proud', 'republic']
```

5. Normalization

Normalization standardizes text by converting it to a consistent format, often as part of noise removal.

Techniques:

- Lowercasing: “Apple” → “apple”
- Removing punctuation: “What’s up?” → “Whats up”
- Handling special characters and spelling corrections

Use in NLP:

- Improves model generalization
- Essential for social media and OCR-generated text

6. N-gram Generation

N-grams are contiguous sequences of n items (typically words) from a text. They capture context and word co-occurrence.

In many applications of text analysis, tokens are not treated individually but based on how they occur together. For example, systems that automatically predict the word that you are going to type next need to look at tokens that are commonly used with one another.

Examples:
Sentence: “I love NLP”

Bigrams (2-grams): (“I love”, “love NLP”)

Use in NLP:

- Language modeling
- Text classification and information retrieval
- Feature extraction for ML models

```
Sentence= "I love NLP"
Sentence=nltk.word_tokenize(Sentence)

#bigrams
bigrams = ngrams(Sentence,2)
# print(list(bigrams))
[('I', 'love'), ('love', 'NLP')]
```

7. Part-of-Speech (POS) Tagging

POS tagging assigns grammatical categories (e.g., noun, verb, adjective) to each word in a sentence.

Example:
Sentence: “She is reading a book.”
Tags: [“She/PRON”, “is/VERB”, “reading/VERB”, “a/DET”, “book/NOUN”]

Use in NLP:

- Syntax analysis
- Named Entity Recognition
- Dependency parsing and semantic role labelling

Part of speech (POS) refers to the category to which a word is assigned based on its function. You may recall that the English language has 8 parts of speech - noun, verb, adjective, adverb, pronoun, determiner, preposition, conjunction, and interjection.

The below code demonstrates POS tagging on text:

```
from string import punctuation
Sentence="She is reading a book."
#Tokenization using word tokenizer
tokenized_list = word_tokenize(Sentence)
# print(tokenized_list)
#Tokenization using word punct tokenizer
punct_tokenized_list = wordpunct_tokenize(Sentence)
print(punct_tokenized_list)
WL=[]
for w in punct_tokenized_list:
    if w not in punctuation:
        WL.append(w)
print(WL)
```

```
from nltk import pos_tag
pos_list = pos_tag(WL)
print(pos_list)
['She', 'is', 'reading', 'a', 'book', '.']
['She', 'is', 'reading', 'a', 'book']
[('She', 'PRP'), ('is', 'VBZ'), ('reading', 'VBG'), ('a', 'DT'), ('book', 'NN')]
```

8. Named Entity Recognition (NER)

NER identifies and classifies named entities such as persons, organizations, dates, locations, etc., from text.

Example:
Sentence: “Google was founded in California in 1998.”
Entities: [“Google” → ORG, “California” → LOC, “1998” → DATE]

Use in NLP:

- Information extraction
- Search engines and recommender systems
- Finance, legal, and healthcare document analysis

Summary Table

Technique	Purpose	Example
Tokenization	Breaks text into words/subwords	“NLP is fun” → [“NLP”, “is”, “fun”]
Lemmatization	Gets base form using grammar rules	“running” → “run”
Stemming	Truncates suffixes	“connection” → “connect”
Stop word removal	Eliminates common, non-informative words	“the”, “is”, “a”
Normalization	Standardizes text formatting	“I’m” → “I am”, “HELLO” → “hello”
N-grams	Captures co-occurrence patterns	Bigrams: “machine learning”
POS Tagging	Tags words with grammatical roles	“play” → Verb or Noun

NER	Extracts named entities	“India” → LOC, “2024” → DATE

NLP Libraries: NLTK and SpaCy Overview

Natural Language Processing tasks can be complex, involving tokenization, tagging, parsing, and more. To simplify these, developers and researchers use specialized NLP libraries that provide ready-to-use tools and models. Two of the most popular open-source libraries in Python are NLTK and SpaCy.

1. What are NLP Libraries?

NLP libraries are frameworks or toolkits that provide pre-built methods for various natural language processing tasks such as:

- Tokenization
- POS Tagging
- Lemmatization/Stemming
- Named Entity Recognition
- Text Classification
- Language Modeling

These libraries abstract the underlying complexity, enabling faster development and experimentation in NLP.

2. NLTK – Natural Language Toolkit

Overview:

- NLTK is one of the oldest and most comprehensive NLP libraries in Python.
- Designed for education and research.
- Offers access to corpora, lexical resources (like WordNet), and statistical NLP tools.

Features:

- Word and sentence tokenization
- POS tagging and chunking
- Named Entity Recognition (NER)
- Stemming and Lemmatization
- Parsing (CFGs, Treebanks)
- Built-in corpora (e.g., Brown, Gutenberg)
- Interface with WordNet for semantic processing

Example Usage:

```
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
text = "Natural Language Processing with NLTK."
tokens = word_tokenize(text)
print(tokens)
```

Strengths:

- Great for learning NLP fundamentals
- Includes educational datasets and lexical resources
- Good for rule-based NLP tasks and grammar exploration

Limitations:

- Slower and less scalable for production use
- Lacks built-in support for GPU-based models or modern deep learning

3. SpaCy – Industrial-strength NLP

Overview:

- SpaCy is a modern NLP library focused on performance, speed, and scalability.
- Built with Cython for speed.
- Designed for real-world applications and production deployment.

Features:

- Tokenization and lemmatization
- POS tagging and dependency parsing
- Named Entity Recognition (NER)
- Pre-trained models for multiple languages
- Easy integration with deep learning frameworks like TensorFlow and PyTorch
- Efficient pipeline management (custom components)

Example Usage:

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("SpaCy is great for fast NLP pipelines.")
for token in doc:
    print(token.text, token.pos_, token.lemma_)
```

Strengths:

- Fast and efficient; suitable for large-scale NLP tasks
- Easy to customize pipelines and add new models
- Compatible with modern deep learning architectures
- Good visualization tools (displacy for dependency parsing and NER)

Limitations:

- Less educational (less focus on theory or syntax trees)
- Smaller built-in corpora compared to NLTK

Comparative Summary: NLTK vs SpaCy

Feature	NLTK	SpaCy
Focus	Education, research	Production, speed, scalability
Tokenization	Yes	Yes (faster and more accurate)
POS Tagging	Yes	Yes
Named Entity Recognition	Basic	Advanced with pre-trained models
Lemmatization	Yes	Yes (via pre-trained models)
Deep Learning Support	Limited	Excellent (integrates with DL models)
Language Models	Few pre-trained	Multiple pre-trained models
Visualization	Minimal	Built-in displacy for syntax and NER