#What Is NLP (Natural Language Processing)?

- Field of AI focused on the interaction between computers and human language.
- Goal: Make machines understand, interpret, and generate human language in a meaningful way.

#Role of NLP in ML Models

#Basic ML Algorithms (Linear Regression, Logistic Regression, etc.):

- Typically used for structured numerical/tabular data → No direct NLP usage.
- However, in **Text Classification (Spam Detection, Sentiment Analysis)** →
  NLP preprocessing is required before applying these models:
  - Tokenization
  - Stop-word removal
  - Text vectorization (Bag of Words, TF-IDF, Word Embeddings)

#NLP-Specific Algorithms & Techniques:

1) Text Preprocessing Steps

- **Tokenization:** Splitting text into words or sub words.
- **Stop-word Removal:** Removing common words ("the", "is", etc.).
- **Stemming / Lemmatization:** Reducing words to root forms (e.g., "running" → "run").
- **Vectorization:**
  - Bag of Words (BoW): Sparse representation → Counts of word occurrences.
  - TF-IDF (Term Frequency-Inverse Document Frequency): Weighs important words.

2) NLP-Specific Models

1. **Naive Bayes Classifier** (Supervised, Classification)
   - Very popular in text classification (Spam vs. Not Spam).
   - Works well with BoW or TF-IDF vectors.
2. **Word Embeddings (Word2Vec, GloVe)**
   - Convert words into dense numerical vectors → Capture semantic meaning.

3) Advanced Gen AI + NLP Integration:

#Transformers (BERT, GPT, etc.):

- Entirely NLP-focused → Used for text generation, classification, question-answering.
- Key Concepts:
  - Self-Attention → Model learns relationships between words.
  - Pre-training + Fine-tuning →
    Pre-train on large corpus → Fine-tune on specific task (e.g., sentiment analysis).
- Example Use-Cases:
  - Text classification
  - Named Entity Recognition (NER)
  - Text summarization
  - Chatbots

#RAG (Retrieval-Augmented Generation):

- Combines document retrieval (NLP-based vector search) + Generative Model (LLM).
- Critical NLP components:
  - Text Embedding Models (OpenAI ada-002, BERT-based embeddings).
  - Chunking Text → Creating semantically meaningful parts for retrieval.

#Why NLP Is Important Here?

- Preprocessing text into structured vectors for ML models (Naive Bayes, Logistic Regression).
- Generative models (GPT, BERT) are NLP-first → All deep learning workflows rely on tokenization, embeddings.
- RAG systems are built on dense embeddings + textual query processing.

#Core NLP Pipeline:

1. Text Preprocessing: Essential for converting raw text into a structured format suitable for models.

#Tokenization:

- Process of splitting text into smaller units → Words, Subwords, or Characters.
- Example:
  Input: "Machine learning is fun."
  Tokens: ["Machine", "learning", "is", "fun", "."]

#Stop-word Removal:

- Common words ("the", "is", "and") removed to reduce noise.
- Example:
  Input: "The cat is on the mat."
  After stop-word removal → ["cat", "mat"]

#Stemming vs. Lemmatization:

- **Stemming:** Removes suffixes to reduce words to base form.
  Example: "running" → "run"
- **Lemmatization:** Converts word to dictionary form considering context.
  Example: "better" → "good"

2. Text Vectorization: Convert tokens into numerical representation.

#Bag of Words (BoW):

- Counts occurrences of words in document → Sparse vectors.
- Example:
  Text A: "I like cats" → [1, 1, 0, 0] (if vocabulary = [I, like, dogs, cats])

#TF-IDF (Term Frequency – Inverse Document Frequency):

- Weighs words by importance → Reduces weight for frequent common words.
- Example Formula:
- TF-IDF(word, document) = TF(word) × log(N / DF(word))

#Word Embeddings:

- Pre-trained dense vectors representing semantic meaning of words.
    - Word2Vec, GloVe, FastText
- Example:
  "King" vector - "Man" vector + "Woman" vector ≈ "Queen" vector

#NLP-Specific Tasks:

1. **Named Entity Recognition (NER):**
   - Extract entities (Person, Organization, Location) from text.
   - Example:
     Sentence: "Google was founded by Larry Page."
     Output: [("Google", ORG), ("Larry Page", PERSON)]

2. **Sentiment Analysis:**
   - Classify text as Positive, Negative, Neutral.
   - Example:
     Review: "Great movie!" → Positive

3. **Text Summarization:**
   - Generate short summary from long text (Extractive or Abstractive).
   - Example:
     Input: Full article → Output: Key summary sentence.

4. **Question-Answering (QA):**
   - Given context + question → Generate precise answer.
   - Example:
     Context: "The Eiffel Tower is in Paris."
     Question: "Where is the Eiffel Tower?"
     Answer: "Paris"

#Tokenization + Stop-word Removal + Lemmatization:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

text = "Natural Language Processing is amazing and very useful."

# Tokenization
tokens = word_tokenize(text)
```

```
# Stop-word Removal
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]

print(lemmatized_tokens)
```

#Vectorization (TF-IDF):

```
from sklearn.feature_extraction.text import TfidfVectorizer

documents = [
   "Machine learning is amazing.",
   "Natural language processing is a branch of AI.",
   "Deep learning models are very powerful." ]

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents)

print(tfidf_matrix.toarray())  #TF-IDF vectors
print(vectorizer.get_feature_names_out())  #Vocabulary
```

#Naive Bayes Text Classification Example:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample Data
texts = ["I love this movie", "This movie is terrible", "Fantastic film!", "I hate this film"]
labels = [1, 0, 1, 0]  # 1 = Positive, 0 = Negative

# Vectorization
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.25, random_state=42)

# Train Naive Bayes
model = MultinomialNB()
model.fit(X_train, y_train)

# Predict & Evaluate
y_pred = model.predict(X_test)
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
```

#Named Entity Recognition (NER) Example:

```
import spacy

# Load pre-trained model
nlp = spacy.load('en_core_web_sm')

text = "Barack Obama was the 44th President of the United States."

doc = nlp(text)
for ent in doc.ents:
   print(f'{ent.text} → {ent.label_}')
```

#Sentiment Analysis Example (Using TextBlob):

```
from textblob import TextBlob

text = "I really enjoyed the movie. It was fantastic and uplifting."

blob = TextBlob(text)
print(f'Polarity: {blob.sentiment.polarity}')  # Range: [-1, 1]
print(f'Subjectivity: {blob.sentiment.subjectivity}')  # Range: [0, 1]
```

#Simple Question-Answering Example (Hugging Face Pipeline):

```
from transformers import pipeline

qa_pipeline = pipeline('question-answering')

context = "The Eiffel Tower is one of the most famous landmarks in Paris."
question = "Where is the Eiffel Tower located?"

result = qa_pipeline(question=question, context=context)
print(f"Answer: {result['answer']}")
```