NLP Pipeline

NLP Pipeline is a set of steps followed to build an end-to-end NLP Software. NLP Software consists of the following steps:

1. Data Acquisition: It refers to the process of collecting, sourcing and preparing textual data that will be used to train, validate and test NLP models.

2. Text Preparation: It is the process of cleaning, normalizing and structuring raw text data so it can be effectively used by Machine or Deep Learning models.

    - Text Cleanup
    - Basic Preprocessing
    - Advance Preprocessing

3. Feature Engineering: It is the process of transforming raw data into meaningful input features that improves the performance of machine learning models. Examples of Feature Engineering are –
    - Handling categorical data (one hot vector)
    - Creating New features: Extract date, month, day from given date
    - Bag of Words: Text- "I love NLP", features- {I:1, love:1, NLP:1}
    - TFIDF: highlights important words by down-weighting common ones.
    - N-grams: ex(bigram), "Machine learning if fun" → ("machine learning", "learning is", "is fun").
    - Word Embeddings: Turns the word into dense vectors, eg: *"king – man + woman ≈ queen"* (Word2Vec)
    - Linguistic feature: parts of speech tags: *Dogs bark loudly"* → [NOUN, VERB, ADV]

4. Modeling: In machine learning, modeling is the process of selecting, training, and evaluating a mathematical model that learns patterns from data and can make predictions or decisions.

5. Deployment: deploy the NLP model

    - Deployment(cloud)
    - Monitoring
    - Model Update

TEXT PREPARATION

1) CLEANING

Before preprocessing, raw data must be cleaned

- HTML Tag Cleaning: Remove HTML tags like <p>hello</p>,
, etc. since they add no meaning to the text.
- Emoji handling: Emojis can either removed or mapped to words (😊 → "smile") depending upon the use case.
- Spelling Check: Correct misspelled words so the model doesn't treat "good" and "good" as different tokens.

2) BASIC PREPROCESSING

After Cleaning, text undergoes basic preprocessing

- Tokenization: Splitting text into smaller units
  - Sentence Tokenization: splitting into sentences
    Example: "I love NLP. It is great." → ["I love NLP.", "It is great."]
  - Word Tokenization: splitting sentences into words
    Example: "I love NLP" → ["I", "love", "NLP"]

3) OPTIMAL PREPROCESSING

Additional steps to optimize text for models:

- Stop word removal: Removing common words like "is", "the", "and", which add little meaning.
- Stemming: Reducing words to their root form (may not always be a valid word).
  Example: "playing" → "play", "happily" → "happi".
- Removing Digits/Punctuation: Eliminate numbers and symbols unless important.
- Lowercasing: Convert all words to lowercase to maintain uniformity. "NLP" and "nlp" should be treated the same.
- Language detection: Identifying the language of text if dataset.

4) <u>ADVANCED PREPROCESSING</u>: More sophisticated techniques:
   - **POS Tagging(Parts of Speech Tagging):** Label each word with its grammatical role (noun, verb, adjective, etc.).
     Example: "Dogs bark" → ("Dogs", Noun), ("bark", Verb)
   - **Parsing:** Understanding sentence structure (syntax tree).
     Example: Subject, Verb, Object relations.
   - **Coreference Resolution:** Resolving references of pronouns/nouns.
     Example: "Ravi went home. He was tired." → "He" = "Ravi".

<u>FEATURE ENGINEERING</u>

Feature engineering is the process of transforming raw data into meaningful inputs (features) that a machine learning model can understand and learn from.

In NLP, raw text is not directly usable by ML models, so we convert it into numerical features. Examples: 1. Bag of Words, 2. Tf-idf, 3. Word Embedding (Word2Vec, GloVe)

1. ML Pipeline

Steps:

Raw Data → Preprocessing → Feature Extraction → Algorithm

- You take raw text data.
- Perform preprocessing (cleaning, tokenization, stop word removal, etc.).
- Then you do Feature Engineering:
- Finally, you feed these features into a machine learning algorithm (like Logistic Regression, SVM, Naïve Bayes).

- Advantage:
- You have control over features (you decide what's important).
- Works well with small datasets.
- Disadvantage:
- Requires manual effort (choosing features, tuning them).
- Features may lose semantic meaning (e.g., Bag of Words ignores context).

---

2. Deep Learning Pipeline:

Steps:

1. Raw Data → Preprocessing → Neural Network
    - Here, you don't explicitly extract handcrafted features.
    - Instead, after minimal preprocessing, raw data (like tokenized text converted into embeddings) is directly fed to a deep learning model (like RNN, LSTM, Transformer, BERT).
    - The model itself learns the best features during training.

- <u>Advantage:</u>
- No need for heavy manual feature engineering.
- Learns contextual and semantic features automatically.
- Works very well with large datasets and complex patterns.
- <u>Disadvantage:</u>
- Requires a lot of data to perform well.
- Computationally expensive (needs GPUs).

MODELLING

When you move past preprocessing and feature engineering, you build models. Different approaches exist:

1) Modeling Approaches:

- Heuristic Models
  - Rule-based methods (e.g., “if a sentence has ‘free’, mark it spam”)
  - Simple, interpretable, but not scalable for complex NLP tasks.

- ML Algorithms
  - Classical ML methods using handcrafted features.
  - Examples: Naïve Bayes, Logistic Regression, SVM, Random Forest.
  - Works fine for small datasets, but performance saturates.

- DL Models
  - Neural networks that learn features automatically
  - Examples: RNN, LSTM, GRU, CNN, Transformer.
  - Require more data, but give much better performance.

- Cloud APLs
  - Pre-trained models offered by companies (Google NLP API, AWS Comprehend, Azure Text Analytics).
  - You don’t train them; just send text and get results.

2). TRANSFER LEARNING: Transfer Learning is a technique where a model trained on large dataset and general task is reused(transferred) for a different but related tasks. Instead of training from scratch, we start with a pre-trained model and fine-tune it on our specific dataset.

Example: A model like BERT is pre-trained on massive text corpora(wkipedia, books).

- ❖ Instead of training from scratch, use a pre-trained model like BERT, GPT, RoBERTa and fine-tune it for particular task.
- ❖ Transfer Learning saves time and require less data

3). **Factors Influencing Model Choice:** Two main factors guide what model you use:

- Amount of Data
    - Small data → classical ML models.
    - Large data → DL models / Transformers.
- Nature of Problem
    - Simple tasks (spam detection, sentiment classification) → ML or even heuristics.
    - Complex tasks (translation, summarization, question answering) → DL/Transformers.

4). Evaluation:

After building models, we evaluate them. Two types of metrics are used:

Intrinsic Evaluation

- Evaluates how well the model performs on the NLP task itself.
- Done using standard metrics.
- Examples:
    - Classification tasks → Accuracy, Precision, Recall, F1-score.
    - Sequence labeling (NER, POS tagging) → Token-level accuracy, F1-score.

    - Language modeling → Perplexity.
    - Machine Translation → BLEU, METEOR scores.
    - Summarization → ROUGE score.

Extrinsic Evaluation

- Evaluates how well the NLP model helps in a downstream/real-world task.
- Example:
    - If you build a Named Entity Recognizer → extrinsic evaluation checks if it improves performance in Information Retrieval or Question Answering.
    - A sentiment model → extrinsic evaluation could be its usefulness in predicting stock movement or customer churn.

In short:

- Intrinsic = internal performance on the NLP task.
- Extrinsic = usefulness in real-world applications.