Natural Language Processing (TYAIML)

Unit-2

Normalization in NLP

Normalization in Natural Language Processing (NLP) is a crucial pre-processing step that transforms text into a consistent, canonical form to improve the performance of downstream NLP tasks. Here's a detailed breakdown:

1. What is Normalization?

Normalization refers to the process of converting text into a standardized format by:

- Handling inconsistencies (e.g., case, punctuation, spelling)
- Reducing noise (e.g., stopwords, special characters)
- Resolving morphological variations (e.g., "running" → "run")

Goal: Ensure that different representations of the same concept are treated identically by NLP models.

2. Why is Normalization Important?

- **Improves Model Accuracy:** Helps algorithms recognize that "USA", "usa", and "U.S.A" refer to the same entity.
- **Reduces Vocabulary Size:** Minimizes unique tokens (e.g., "Cat" and "cat" → treated as one word).
- **Enhances Efficiency:** Faster processing with cleaner, standardized text.
- **Better Generalization:** Models perform well on real-world noisy text (social media, emails, etc.).

3. Key Normalization Techniques

(A) Case Normalization

- Converts all text to lowercase (or uppercase) to avoid case sensitivity issues.

  e.g. text = "Hello World!“, normalized_text = "hello world!"

(B) Punctuation & Special Character Removal

- Removes non-alphanumeric characters unless they carry meaning (e.g., hashtags, emojis in sentiment analysis).

  e.g. text = "Hello, world! How are you?” normalized_text = "Hello world How are you"

(C) Stopword Removal

- Filters out common words (e.g., "the", "is") that add little semantic value.

  e.g. text = "this is an example sentence", normalized_text = "example sentence"

Tokenization in NLP

Tokenization is the process of breaking down text into smaller units called tokens**,** which can be words, subwords, or characters. It's the first step in NLP pipelines and critical for tasks like text classification, machine translation, and sentiment analysis.

1. Why Tokenization?

- Converts raw text into a format understandable by machines.
- Enables feature extraction (e.g., word frequencies, embeddings).
- Handles language-specific nuances (e.g., compound words in German).

2. Types of Tokenization

(A) Word Tokenization

Splits text into **words** based on spaces/punctuation.

text = "I love NLP!" Output: ['I', 'love', 'NLP', '!']

(B) Sentence Tokenization

Splits text into **sentences**.

text = "Hello! How are you? I'm fine." # Output: ['Hello!', 'How are you?', "I'm fine."]

(C) Subword Tokenization

Splits words into **smaller meaningful units** (subwords/morphemes).

Popular Algorithms:

1. **Byte Pair Encoding (BPE)**
   - Used in GPT, RoBERTa. Merges frequent character pairs iteratively.
   - Example: "unhappiness" → ["un", "happiness"].
2. **WordPiece**
   - Used in BERT. Similar to BPE but optimizes for likelihood.
   - Example: "jumping" → ["jump", "##ing"].
3. **SentencePiece**
   - Language-agnostic (works with raw text, no pre-tokenization).
   - Example: "Hello world!" → ["▁Hello", "▁world", "!"].

(D) Character Tokenization

Splits text into **individual characters**.

Example: text = "NLP" # Output: ['N', 'L', 'P']

(E) Whitespace Tokenization

Splits text based on **whitespace only**.

- Example: text = "Tokenization is easy" # Output: ['Tokenization', 'is', 'easy']

Stemming in NLP

Stemming is a **text normalization** technique in Natural Language Processing (NLP) that reduces words to their base or root form by removing suffixes and prefixes. The goal is to convert different forms of a word into a common representation.

How Stemming Works

Stemming applies heuristic rules to chop off word endings, often without considering context or meaning.

For example:

- "running" → "run"
- "happiness" → "happi" (imperfect, but reduces variation)

Common Stemming Algorithms-

Here are the most commonly used stemming algorithms:

1. Porter Stemmer (Porter, 1980)

- Most widely used stemming algorithm for English.
- Uses a series of heuristic steps (5 phases) to strip suffixes.
- Fast but aggressive—may over-stem (cut too much) or under-stem (leave too much).
- Example:
  - "running" → "run"
  - "happiness" → "happi" (not a real word)

2. Snowball Stemmer (Porter2, Improved Porter)

- An improved version of the Porter Stemmer.
- Supports multiple languages (English, French, Spanish, etc.).
- More consistent than Porter but still heuristic-based.
- Example:
  - "jumping" → "jump"
  - "easily" → "easili"

3. Lancaster Stemmer (Paice/Husk, 1990)

- More aggressive than Porter.
- Uses iterative rules and tends to over-stem words.
- Example:
  - "running" → "run"
  - "maximum" → "maxim" (unlike Porter's "maximum")

4. Lovins Stemmer (1968)

- One of the earliest stemming algorithms.
- Uses a large set of rules (294 endings).
- Faster than Porter but less used today.
- Example:
  - "happiness" → "happy"

5. Regex-Based Stemmer (Custom Rules)

- Uses regular expressions to define stemming rules.
- Flexible but requires manual tuning.
- Example:
  - Remove "-ing": "playing" → "play"

Comparison-

Stemmer	Language Support	Aggressiveness	Accuracy	Speed
Porter	English	Moderate	Medium	Fast
Snowball	Multi-language	Moderate	Medium	Fast
Lancaster	English	High (Over-stems)	Low	Fast
Lovins	English	High	Low	Fast
Regex	Custom	Custom	Varies	Fast

Over Stemming and Under Stemming-

Over-Stemming-

Definition: Over-stemming occurs when two different root words are reduced to the same stem, even though they have different meanings.

Consequence: It leads to loss of meaning or semantic confusion, as unrelated words are incorrectly grouped together.

Example from Image:

- Words: University and Universal
- Over-stemmed result: Both may be reduced to “univers”
- Problem: These are different in meaning (one refers to an educational institution, the other to something global), but are treated as the same.

Under-Stemming-

Definition: Under-stemming happens when related words are not reduced to the same stem, meaning the stemming process is too conservative.

Consequence: This leads to missed connections between words that should be grouped, reducing effectiveness in tasks like information retrieval or search.

Example from Image:

- Words: Data and Database
- Under-stemmed result: Data and Database remain as separate stems
- Problem: These are semantically related, but they are not treated as such.

Lemmatization in NLP

Lemmatization is a text normalization technique in Natural Language Processing (NLP) that reduces words to their base or dictionary form (lemma) by considering vocabulary and morphological analysis. Unlike stemming, which simply chops off word endings, lemmatization ensures that the output is a valid word present in the language.

Working-

1. **Uses a Vocabulary & Morphological Analysis** – Considers word meanings and parts of speech (POS).
2. **More Accurate than Stemming** – Returns meaningful base words.
3. **Slower than Stemming** – Requires more computational power due to linguistic analysis.

When to Use?

- Sentiment Analysis (e.g., "better" → "good")
- Chatbots & Virtual Assistants (needs meaningful words)
- Machine Translation (preserves word meaning)
- Search Engines & Text Mining (better than stemming for precision)

Stemming vs Lemmatization-

Feature	Stemming	Lemmatization
Approach	Rule-based (cuts suffixes)	Dictionary-based (linguistic analysis)
Output	May not be a real word (e.g., "happi")	Always a valid word (e.g., "happy")
Context	Ignores word meaning	Considers Part-of-Speech (POS)
Speed	Faster	Slower (due to POS analysis)
Accuracy	Lower	Higher

WordNet in NLP

WordNet is a large lexical database of English developed at Princeton University. It organizes words into synsets (synonym sets) and defines semantic relationships between them. Unlike a traditional dictionary, WordNet connects words based on their meanings rather than just alphabetical order.

Key Features:

- Contains nouns, verbs, adjectives, and adverbs.

- Groups words into synsets (sets of synonyms).
- Defines semantic relations (hypernyms, hyponyms, meronyms, etc.).
- Used for lemmatization, word sense disambiguation (WSD), and semantic analysis.

Structure-

A. Synsets (Synonym Sets)

- A **synset** represents a **unique meaning** of a word.
- Example:
  - The word **"bank"** has multiple synsets:
    - bank.n.01 (financial institution)
    - bank.n.02 (sloping land beside a river)
    - bank.v.01 (tip laterally, as in "bank an airplane")

B. Lexical Relationships

WordNet defines relationships between words, such as:

Relation	Description	Example
Hypernym	More general term (is-a)	"Dog" → "Animal"
Hyponym	More specific term	"Animal" → "Dog"
Meronym	Part-of relation	"Wheel" → "Car"
Holonym	Whole-of relation	"Car" → "Wheel"
Antonym	Opposite meaning	"Hot" → "Cold"

Advantages-

- Structured Lexical Database – Organizes words by meaning, not just spelling.
- Supports Semantic Analysis – Useful for WSD, sentiment analysis, and chatbots.
- Open-Source & Widely Used – Available in NLTK, spaCy, and Gensim.

Limitations-

- English-Centric – Limited support for other languages.
- Manual Curation – Not automatically updated with new slang/terms.
- Computationally Heavy – Large databases require more processing.

Corpora in NLP

A **corpus** (plural: **corpora**) is a large, structured collection of **machine-readable texts** used for linguistic analysis and NLP tasks. Corpora serve as the foundational data for training and evaluating language models.

Key Characteristics:

- Machine-readable (digitally stored and processed)
- Structured (often annotated with metadata like POS tags, sentiment, etc.)
- Representative of a language, dialect, or domain
- Balanced (covers diverse text types when possible)

Types-

Type	Description	Example
Monolingual	Contains text in one language	British National Corpus
Multilingual	Text in multiple languages	Wikipedia dumps
Parallel	Same content in different languages	Europarl (EU Parliament speeches)
Annotated	Text with metadata (POS tags, syntax etc.)	Penn Treebank
Spoken	Transcriptions of spoken language	Switchboard Corpus
Specialized	Focused on a specific domain	Medical or Legal Corpus

Commonly Used Corpora in NLP-

Corpus	Content	Use Case
Brown Corpus	First million-word English corpus	POS tagging, lexical analysis
WordNet	Lexical database	Relationships, lemmatization
Penn Treebank	Syntactically annotated corpus	Parsing, grammar analysis
COCA	Corpus of Contemporary American English	Word usage, frequency
Reuters Corpus	News articles	Topic modelling, classification
OpenSubtitles	Movie/TV subtitles	Dialogue systems, translation

Applications of Corpora in NLP-

- Language modelling
- POS tagging
- Named Entity Recognition (NER)
- Sentiment analysis
- Machine translation
- Question answering
- Chatbot training

Challenges with Corpora-

- Size: Larger corpora require more storage and processing power.
- Bias: Corpora can reflect social or cultural biases.
- Language Evolution: Corpora can become outdated quickly.
- Annotation Errors: Manual or automatic annotations can be incorrect.

Vital Uses of Corpora-

The several important uses of corpora in Natural Language Processing (NLP) and linguistics are as follows:

1. Lexicography

Definition: The art and science of dictionary-making.

Use of corpora:

- Corpora help identify the actual usage of words in various contexts.
- Lexicographers analyze large collections of text to understand:
    - Word meanings
    - Word frequencies
    - Collocations (words that commonly appear together)

Example:

To define the word *"run"*, a corpus can show it used in:

- "He runs a business."
- "She runs every morning."

  This helps determine different senses of the word and common phrases like *"run a risk"* or *"run out of time"*.

2. Grammar and Syntax

Definition: Study of sentence structures and how words function within them.

Use of corpora:

- Reveal common syntactic patterns used in real language.
- Help linguists or computational systems detect grammatical structures like:
    - Subject–Verb–Object order
    - Passive voice patterns
    - Use of modal verbs and tenses

Example:

Analysing a corpus can show that "is being eaten" is more common than "is getting eaten," which informs grammar instruction and NLP models.

3. Stylistics

Definition: The study of writing style, often for literary or rhetorical analysis.

Use of corpora:

- Compare writing styles across genres or authors.
- Analyse:
    - Sentence length
    - Word choices

- Use of figurative language
- Formality level

Example:

Comparing a political speech corpus with a poetry corpus reveals how vocabulary, tone, and structure vary dramatically between genres.

4. Training and Evaluation (in NLP)

Definition: Using corpora to train and test machine learning and NLP systems.

Use of corpora:

- Training supervised models for tasks such as:
  - Part-of-speech (POS) tagging
  - Named Entity Recognition (NER)
  - Machine translation
- Evaluating model performance on test corpora using metrics like:
  - Accuracy
  - Precision/Recall
  - BLEU score (for translation)

Example:

- The Penn Treebank is a standard corpus used to train syntactic parsers.
- The CoNLL 2003 dataset is used to evaluate named entity recognition systems.

Basic Corpus Analysis

A corpus (plural: corpora) is a large and structured set of texts used in linguistic and NLP research. Corpus analysis refers to studying this data to uncover patterns, trends, and linguistic features.

One of the fundamental steps in corpus analysis is:

Frequency Distribution Building-

What is it?

Frequency distribution is the process of counting how often each word, phrase, or token appears in a corpus.

Why is it useful?

- Identifies the most commonly used words**.**
- Helps filter out stopwords (like “the”, “and”, “is”).
- Assists in topic modelling**,** keyword extraction, and other NLP tasks.
- Useful in text classification**,** sentiment analysis, etc.

Example:

If a corpus contains: "The cat sat on the mat. The cat is happy."

Frequency distribution: the-3, cat-2, sat-1, on-1, mat-1, is-1, happy-1.

Analysing a Corpus-

Once we build a frequency distribution, analysis may include:

1. Word Frequency Patterns
    - Common vs rare words.
    - Domain-specific vocabulary.
2. Collocation Analysis
    - Words that often appear together (e.g., *strong tea*, *climate change*).
3. Concordance Analysis
    - See words in context (KWIC – Keyword in Context).
    - Helps understand meaning and usage**.**
4. Part-of-Speech Tagging
    - Determine grammatical roles (noun, verb, etc.) of words in a corpus.
5. Named Entity Recognition (NER)
    - Identify and classify proper names (people, places, organizations).

Natural Language Toolkit (NLTK):

NLTK (Natural Language Toolkit) is a leading Python library for working with human language data. Developed at the University of Pennsylvania, it provides easy-to-use interfaces to over 50 corpora and lexical resources along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

Key Features-

Feature	Description
Tokenization	Splitting text into words, sentences, or smaller units.
Stopword Removal	Removing common words like “the”, “is”, etc.
Stemming & Lemmatization	Reducing words to their base/root form.
POS Tagging	Identifying parts of speech (noun, verb, etc.)
Named Entity Recognition	Identifying proper names (e.g., "India", "Apple Inc.").
Corpora Access	Access to pre-loaded linguistic corpora
Text Classification	Sentiment analysis, document classification, etc.
Parsing	Syntax tree generation for grammatical analysis.

Core Components-

A. Corpora and Lexical Resources

NLTK includes many important corpora:

- Gutenberg Corpus: Classic literary texts
- Brown Corpus: First million-word electronic corpus
- WordNet: Lexical database of English
- Stopwords: Common words to filter out
- CMU Pronouncing Dictionary: For speech processing

B. Text Processing Modules

1. Tokenization- Splitting text into words, sentences, or other units.
2. Stemming & Lemmatization- Reducing words to their base forms.
3. Part-of-Speech Tagging-Labelling words with their grammatical roles.
4. Named Entity Recognition-Identifying names, organizations, locations etc.

C. Statistical NLP

1. Frequency Distributions- Analysing word occurrences.
2. Collocations- Finding common word pairs.

D. Machine Learning Capabilities

1. Text Classification
2. Sentiment Analysis
3. Visualization Tools

   A. Dispersion Plots- Show word distribution across text.

   B. Parse Trees- Visualize sentence structure.

Advantages:

- Excellent for teaching and research
- Comprehensive linguistic tools
- Well-documented with many examples

Limitations:

- Not as fast as spaCy for production
- Somewhat outdated ML algorithms
- Requires separate downloads for corpora.

******