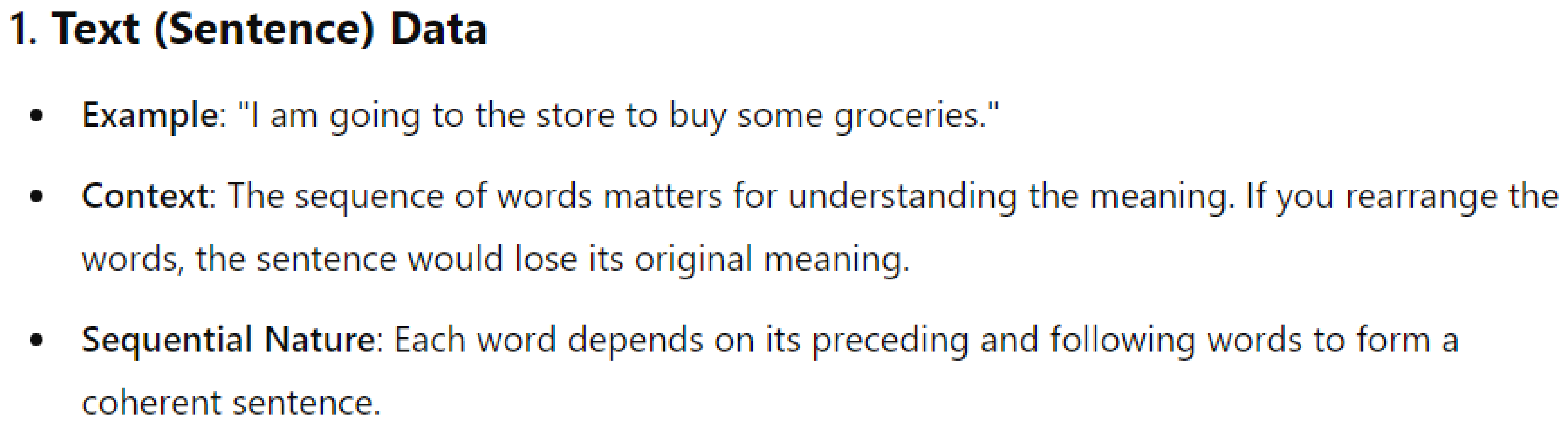
CAP 378
ARTIFICIAL INTELIGENCE

Tanzeela Javid Kaloo (32638)
Assistant Professor
System And Architecture
Lovely Professional University

UNIT – VI
Foundation of Machine Learning

Tanzeela Javid Kaloo (32638)
Assistant Professor
System And Architecture
Lovely Professional University

Sequential Data Types



1. Text (Sentence) Data

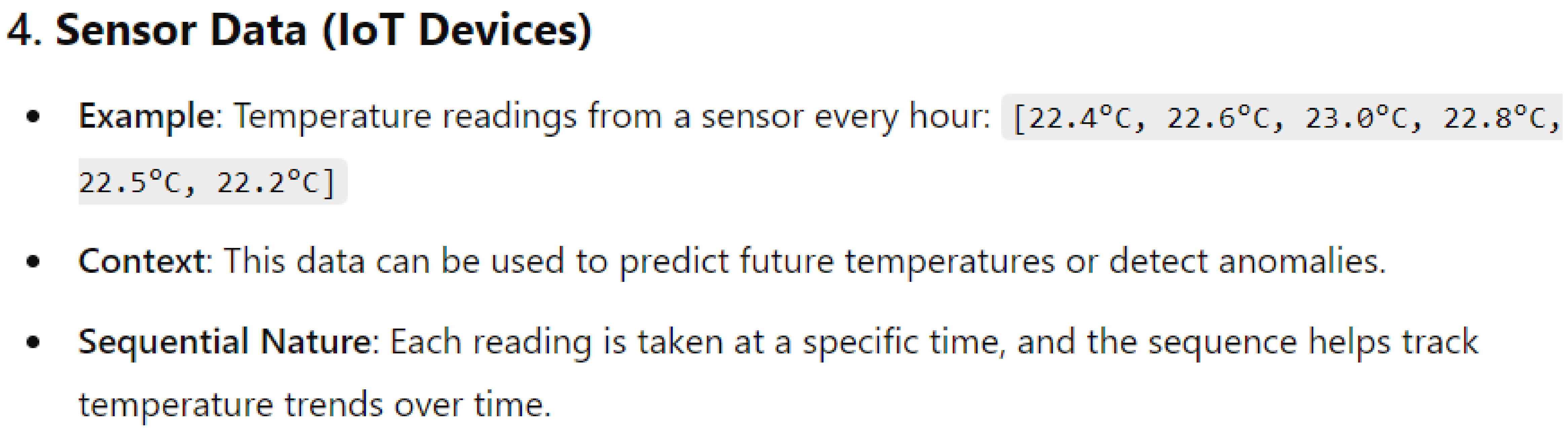
- **Example**: "I am going to the store to buy some groceries."
- **Context**: The sequence of words matters for understanding the meaning. If you rearrange the words, the sentence would lose its original meaning.
- **Sequential Nature**: Each word depends on its preceding and following words to form a coherent sentence.

2. Time Series Data (Stock Prices)

- **Example**: A sequence of daily closing stock prices: `[120.5, 121.3, 119.8, 118.6, 120.2, 121.0]`
- **Context**: In finance, predicting future stock prices requires understanding trends and patterns in past prices.
- **Sequential Nature**: Each price depends on the previous day's price, and the entire sequence is used to analyze trends or make predictions.

3. Speech Data

- **Example**: Audio waveform data for the spoken sentence "Hello, how are you?"
  - Waveform: `[0.12, 0.35, -0.45, 0.22, -0.30, 0.15, ...]`
- **Context**: When processing audio, the order of the sound samples is crucial for reconstructing the spoken words.
- **Sequential Nature**: Each audio sample represents a point in time, and the temporal order must be preserved for accurate speech recognition or synthesis.

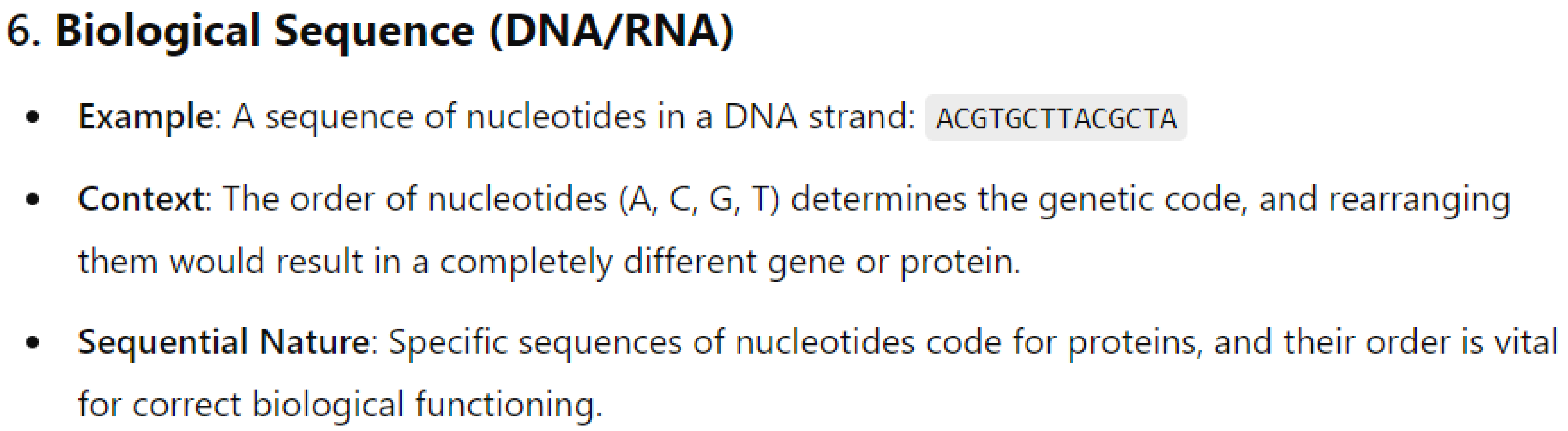


4. Sensor Data (IoT Devices)

- **Example**: Temperature readings from a sensor every hour: `[22.4°C, 22.6°C, 23.0°C, 22.8°C, 22.5°C, 22.2°C]`
- **Context**: This data can be used to predict future temperatures or detect anomalies.
- **Sequential Nature**: Each reading is taken at a specific time, and the sequence helps track temperature trends over time.

5. Financial Transactions (Credit Card Data)

- **Example**: A series of credit card transactions:
  - `[Date: Oct 1, Amount: $50, Location: Grocery Store]`
  - `[Date: Oct 2, Amount: $200, Location: Electronics Store]`
  - `[Date: Oct 3, Amount: $15, Location: Coffee Shop]`
- **Context**: This sequence could be analyzed for fraud detection or understanding purchasing behavior over time.
- **Sequential Nature**: The order of transactions is important to identify patterns, like frequent purchases in a short time that may indicate fraud.



6. **Biological Sequence (DNA/RNA)**

- **Example**: A sequence of nucleotides in a DNA strand: `ACGTGCTTACGCTA`
- **Context**: The order of nucleotides (A, C, G, T) determines the genetic code, and rearranging them would result in a completely different gene or protein.
- **Sequential Nature**: Specific sequences of nucleotides code for proteins, and their order is vital for correct biological functioning.

7. User Activity Log (Web or App Usage)

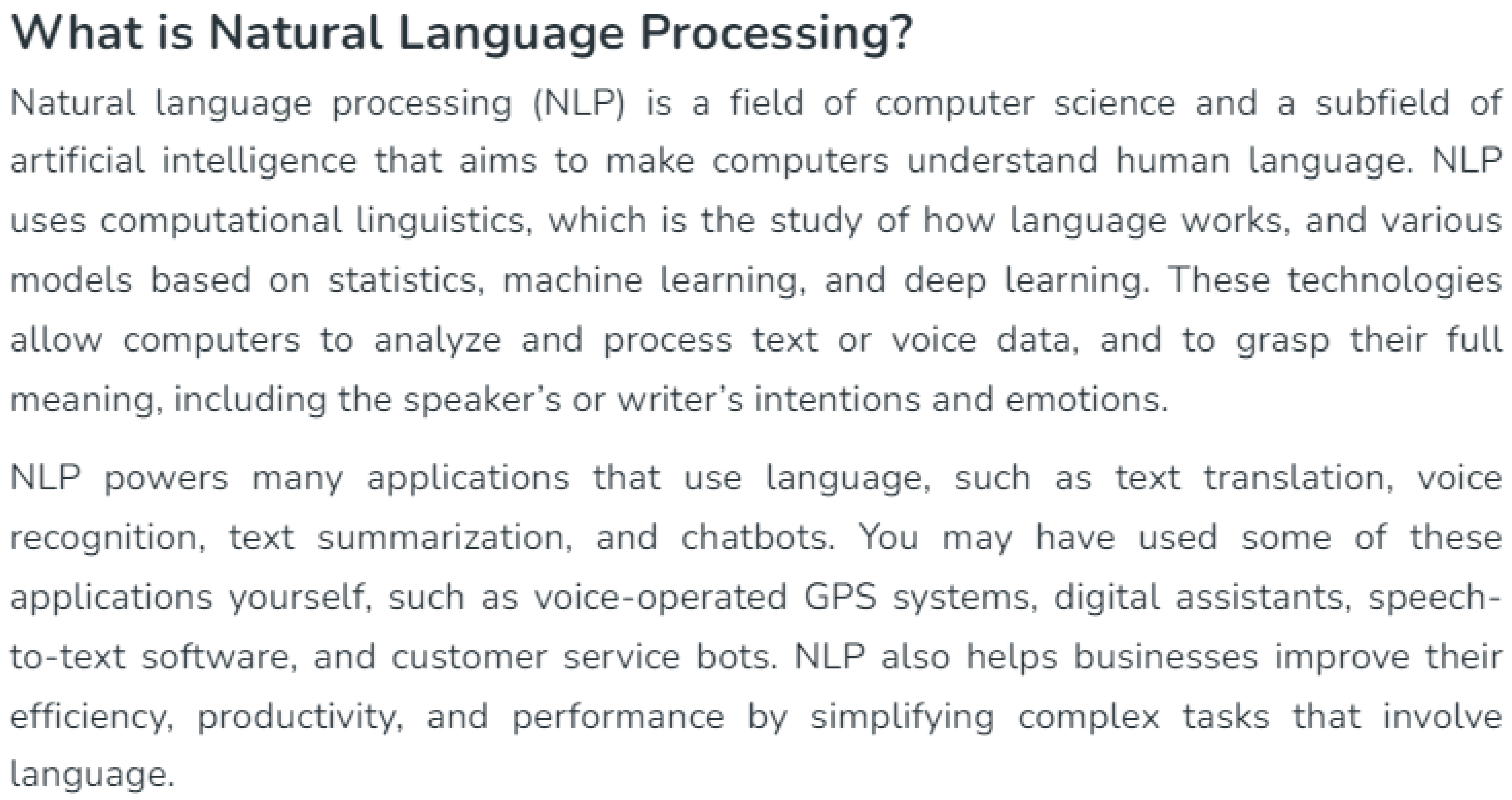
- **Example**: A sequence of actions taken by a user on a website:
    - `[Logged in, Viewed Homepage, Searched for 'Laptops', Added 'Laptop' to Cart, Checked Out]`
- **Context**: This type of data can be used for personalized recommendations or understanding user behavior.
- **Sequential Nature**: The order of actions provides insight into the user's intent, and analyzing the sequence can reveal patterns such as typical user journeys.

8. Machine Data (Production Line Sensor Data)

- **Example**: A sequence of vibration readings from a machine on a production line:
  - `[0.15, 0.20, 0.22, 0.30, 0.50]`
- **Context**: Monitoring this data helps detect when a machine is malfunctioning or needs maintenance.
- **Sequential Nature**: The gradual increase in vibration readings over time may indicate wear or the need for maintenance.

9. Customer Support Chat Logs

- **Example**: A conversation between a customer and a support agent:
  - **Customer**: "I’m having trouble logging in."
  - **Support**: "Can you try resetting your password?"
  - **Customer**: "I tried that, but it didn’t work."
- **Context**: The conversation's flow is essential for providing coherent responses and understanding customer needs.
- **Sequential Nature**: The dialogue must be processed in order to maintain the context of the conversation.



What is Natural Language Processing?

Natural language processing (NLP) is a field of computer science and a subfield of artificial intelligence that aims to make computers understand human language. NLP uses computational linguistics, which is the study of how language works, and various models based on statistics, machine learning, and deep learning. These technologies allow computers to analyze and process text or voice data, and to grasp their full meaning, including the speaker's or writer's intentions and emotions.

NLP powers many applications that use language, such as text translation, voice recognition, text summarization, and chatbots. You may have used some of these applications yourself, such as voice-operated GPS systems, digital assistants, speech-to-text software, and customer service bots. NLP also helps businesses improve their efficiency, productivity, and performance by simplifying complex tasks that involve language.

NLP Techniques

NLP encompasses a wide array of techniques that aimed at enabling computers to process and understand human language. These tasks can be categorized into several broad areas, each addressing different aspects of language processing. Here are some of the key NLP techniques:

1. **Text Processing and Preprocessing In NLP**
2. **Syntax and Parsing In NLP**
3. **Semantic Analysis**
4. **Information Extraction**
5. **Text Classification in NLP**

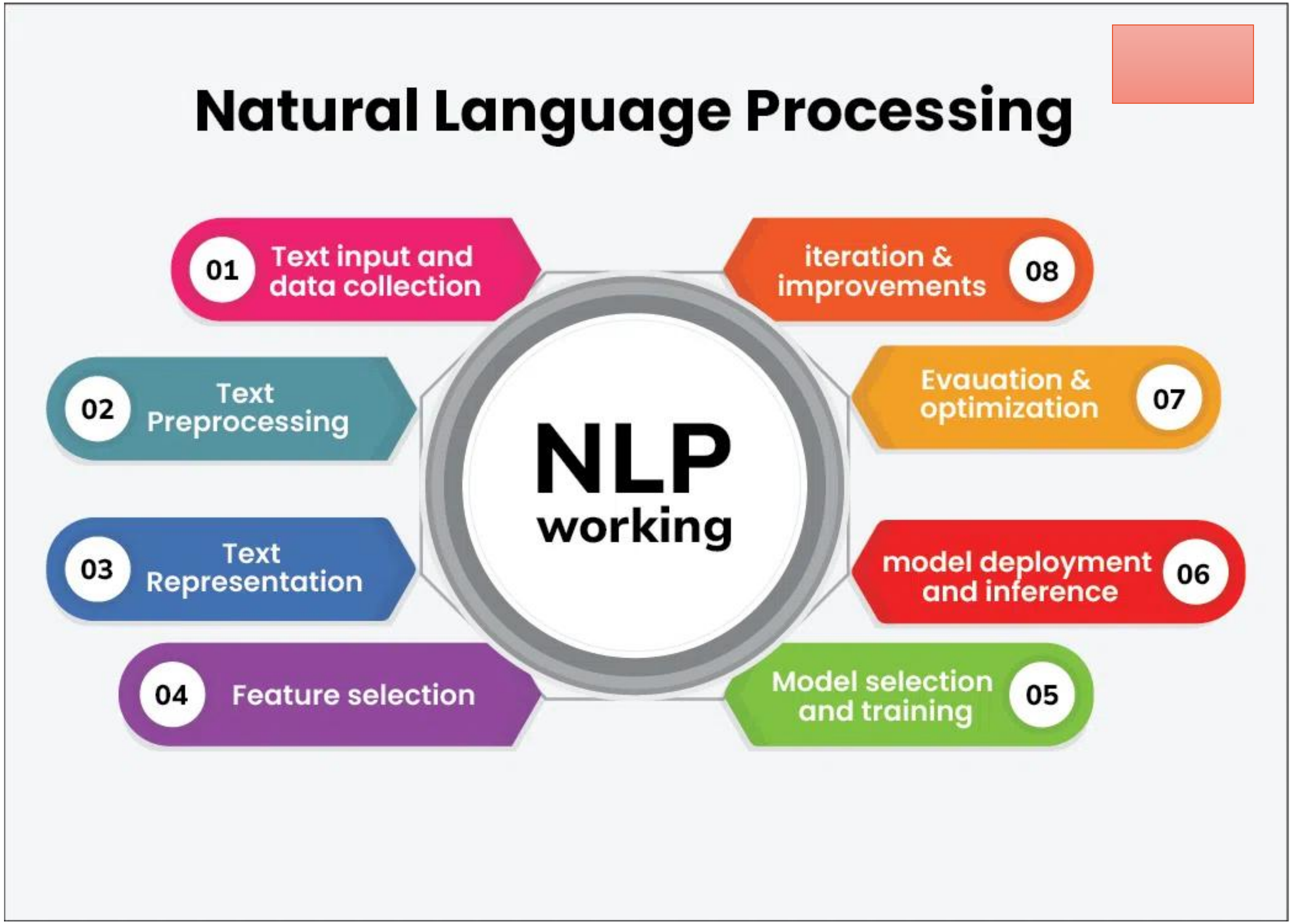
6. Language Generation

7. Speech Processing

8. Question Answering

9. Dialogue Systems

10. Sentiment and Emotion Analysis in NLP


Natural Language Processing
NLP working
01 Text input and data collection
02 Text Preprocessing
03 Text Representation
04 Feature selection
05 Model selection and training
06 model deployment and inference
07 Evauation & optimization
08 iteration & improvements

Working in natural language processing (NLP) typically involves using computational techniques to analyze and understand human language. This can include tasks such as language understanding, language generation, and language interaction.

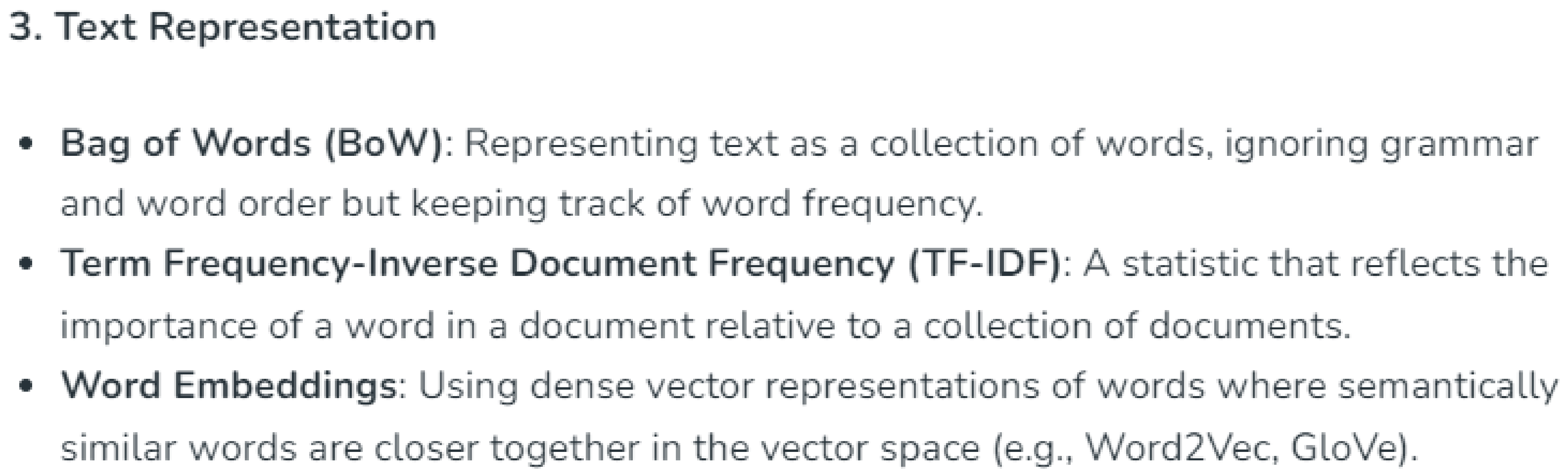
1. Text Input and Data Collection

- **Data Collection**: Gathering text data from various sources such as websites, books, social media, or proprietary databases.
- **Data Storage**: Storing the collected text data in a structured format, such as a database or a collection of documents.

2. Text Preprocessing

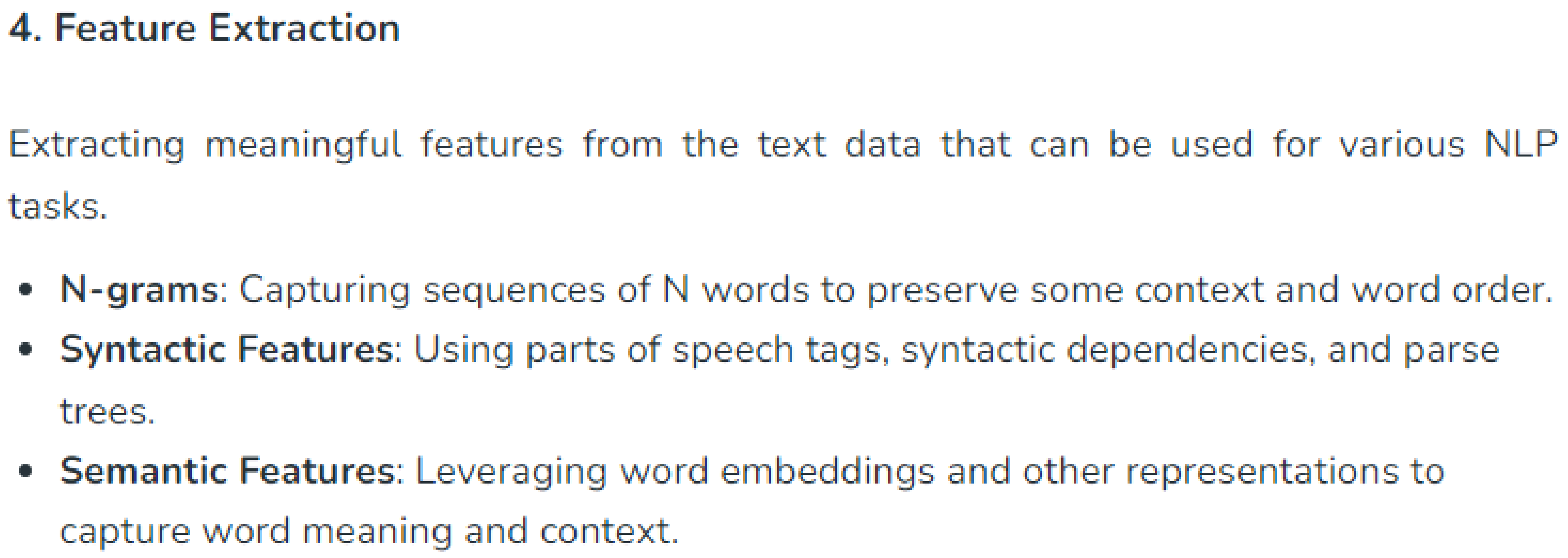
Preprocessing is crucial to clean and prepare the raw text data for analysis. Common preprocessing steps include:

- **Tokenization**: Splitting text into smaller units like words or sentences.
- **Lowercasing**: Converting all text to lowercase to ensure uniformity.
- **Stopword Removal**: Removing common words that do not contribute significant meaning, such as “and,” “the,” “is.”
- **Punctuation Removal**: Removing punctuation marks.
- **Stemming and Lemmatization**: Reducing words to their base or root forms. Stemming cuts off suffixes, while lemmatization considers the context and converts words to their meaningful base form.
- **Text Normalization**: Standardizing text format, including correcting spelling errors, expanding contractions, and handling special characters.



3. Text Representation

- **Bag of Words (BoW)**: Representing text as a collection of words, ignoring grammar and word order but keeping track of word frequency.
- **Term Frequency-Inverse Document Frequency (TF-IDF)**: A statistic that reflects the importance of a word in a document relative to a collection of documents.
- **Word Embeddings**: Using dense vector representations of words where semantically similar words are closer together in the vector space (e.g., Word2Vec, GloVe).



4. Feature Extraction

Extracting meaningful features from the text data that can be used for various NLP tasks.

- **N-grams**: Capturing sequences of N words to preserve some context and word order.
- **Syntactic Features**: Using parts of speech tags, syntactic dependencies, and parse trees.
- **Semantic Features**: Leveraging word embeddings and other representations to capture word meaning and context.

5. Model Selection and Training

Selecting and training a machine learning or deep learning model to perform specific NLP tasks.

- **Supervised Learning**: Using labeled data to train models like Support Vector Machines (SVM), Random Forests, or deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).
- **Unsupervised Learning**: Applying techniques like clustering or topic modeling (e.g., Latent Dirichlet Allocation) on unlabeled data.
- **Pre-trained Models**: Utilizing pre-trained language models such as BERT, GPT, or transformer-based models that have been trained on large corpora.

6. Model Deployment and Inference

Deploying the trained model and using it to make predictions or extract insights from new text data.

- **Text Classification**: Categorizing text into predefined classes (e.g., spam detection, sentiment analysis).
- **Named Entity Recognition (NER)**: Identifying and classifying entities in the text.
- **Machine Translation**: Translating text from one language to another.
- **Question Answering**: Providing answers to questions based on the context provided by text data.

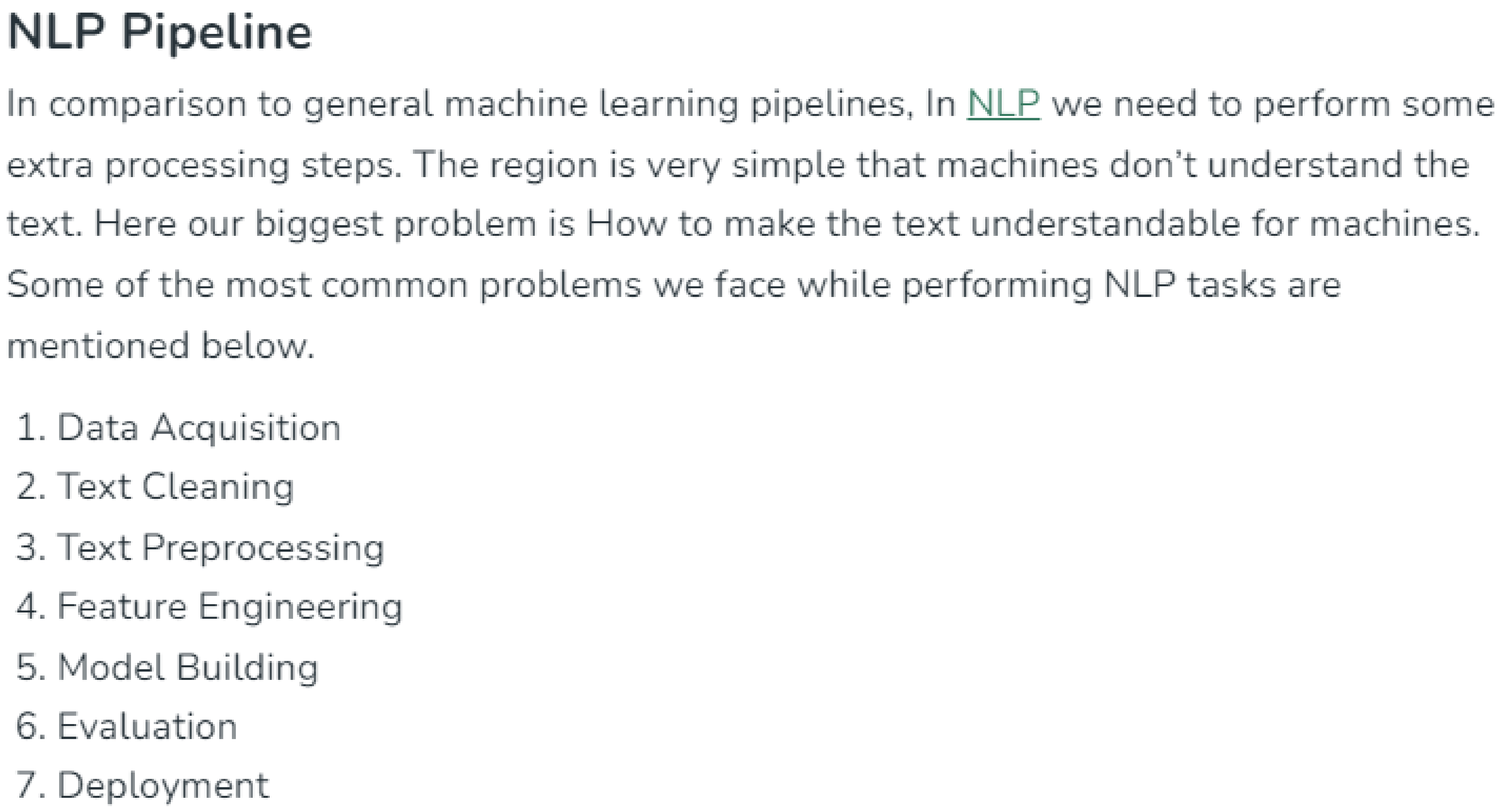
7. Evaluation and Optimization

Evaluating the performance of the NLP algorithm using metrics such as accuracy, precision, recall, F1-score, and others.

- **Hyperparameter Tuning**: Adjusting model parameters to improve performance.
- **Error Analysis**: Analyzing errors to understand model weaknesses and improve robustness.

8. Iteration and Improvement

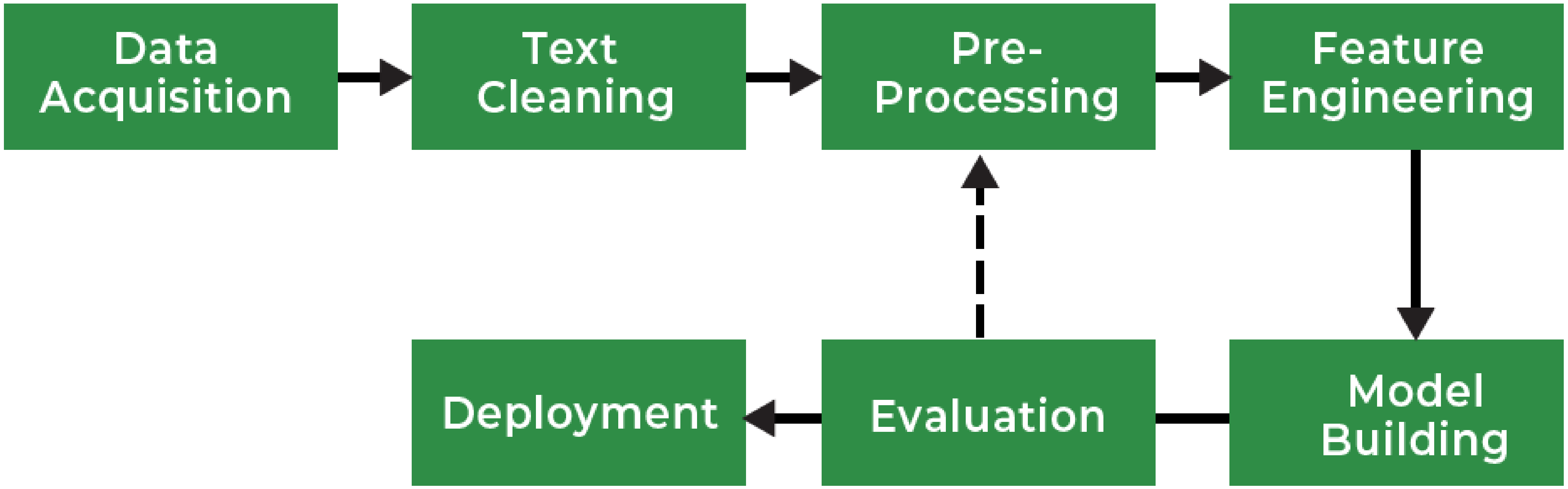
Continuously improving the algorithm by incorporating new data, refining preprocessing techniques, experimenting with different models, and optimizing features.



NLP Pipeline

In comparison to general machine learning pipelines, In NLP we need to perform some extra processing steps. The region is very simple that machines don't understand the text. Here our biggest problem is How to make the text understandable for machines. Some of the most common problems we face while performing NLP tasks are mentioned below.

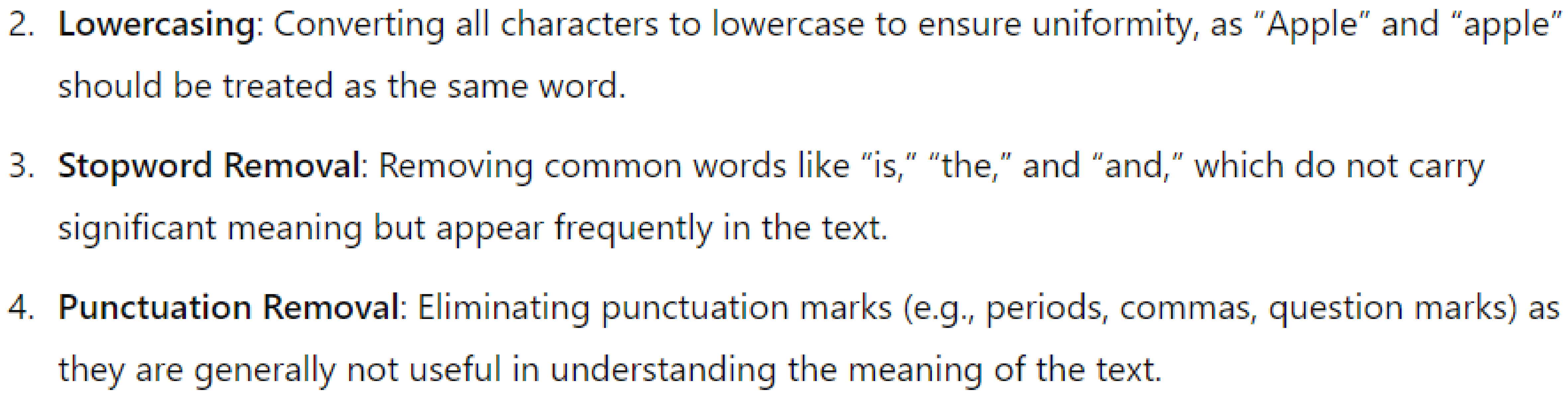
1. Data Acquisition
2. Text Cleaning
3. Text Preprocessing
4. Feature Engineering
5. Model Building
6. Evaluation
7. Deployment


Data Acquisition
Text Cleaning
Pre-Processing
Feature Engineering
Model Building
Evaluation
Deployment

Pre-processing Steps in NLP:

Before you can apply machine learning or deep learning models, you need to prepare and clean the text data. The following are the key pre-processing steps:

1. **Tokenization**: This is the process of breaking down text into smaller units, like words or sentences. For example, tokenizing a sentence splits it into individual words.
   - *Word Tokenization*: Splitting a text into individual words.
   - *Sentence Tokenization*: Breaking the text into sentences.



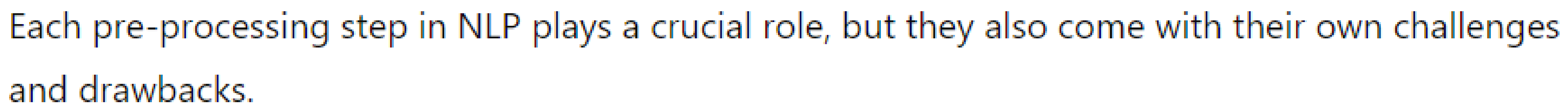
2. **Lowercasing**: Converting all characters to lowercase to ensure uniformity, as “Apple” and “apple” should be treated as the same word.
3. **Stopword Removal**: Removing common words like “is,” “the,” and “and,” which do not carry significant meaning but appear frequently in the text.
4. **Punctuation Removal**: Eliminating punctuation marks (e.g., periods, commas, question marks) as they are generally not useful in understanding the meaning of the text.

5. **Stemming and Lemmatization**: These techniques reduce words to their base or root form.
   - *Stemming*: Cuts off word endings to get the base form (e.g., "running" becomes "run").
   - *Lemmatization*: Converts a word to its dictionary form (e.g., "better" becomes "good").
6. **Removing Special Characters**: Deleting characters like hashtags, numbers, or symbols that might not be relevant depending on the use case.

7. **Text Normalization**: This includes a variety of techniques to standardize the text. It could involve removing extra spaces, converting text to a common format (e.g., standardizing abbreviations like "u" to "you"), or expanding contractions (e.g., “can’t” becomes “cannot”).

8. **N-grams**: Sometimes, individual words do not capture enough meaning. N-grams are combinations of words (bigrams = two-word combinations, trigrams = three-word combinations) that capture context. For instance, in "New York City," it's useful to keep "New York" together as a bigram.

9. **Encoding Text**: Since machines work with numbers, not words, you'll need to convert text into numerical representations, such as:
   - **Bag of Words (BoW)**: Represents the text as a vector of word frequencies.
   - **TF-IDF (Term Frequency-Inverse Document Frequency)**: Weighs words by how important they are in the document relative to a corpus.
   - **Word Embeddings**: More advanced techniques like Word2Vec or GloVe that capture the semantic meaning of words by placing them in a multi-dimensional space.


Each pre-processing step in NLP plays a crucial role, but they also come with their own challenges and drawbacks.

1. Tokenization

Drawbacks and Challenges:

- **Ambiguity**: Some languages, like Chinese or Japanese, don’t have clear word boundaries, making tokenization harder.
- **Compound words**: Tokenizers might split important compound words (like "New York") or hyphenated words incorrectly.
- **Handling special cases**: Contractions (like "can't"), possessives, and multi-word expressions can create confusion. For example, should "can’t" be tokenized as "can" and "not"? This depends on context.

Challenge: Tokenizers need to balance splitting text into manageable pieces without losing semantic meaning or context.

2. **Lowercasing**

Drawbacks and Challenges:

- **Loss of information**: Lowercasing can remove distinctions between words where case matters. For example, "US" (United States) and "us" (a pronoun) would become the same.

Challenge: Deciding when lowercasing is appropriate, as it can result in loss of key distinctions.

3. **Stopword Removal**

Drawbacks and Challenges:

- **Important words could be removed**: Stopwords like "not" or "no" can change the meaning of a sentence entirely. For example, "This is not bad" becomes meaningless if “not” is removed.
- **Language dependency**: Stopwords vary by language and context. A one-size-fits-all approach doesn’t work globally.

Challenge: It’s difficult to determine an optimal stopword list for every task without losing important information.

4. Punctuation Removal

Drawbacks and Challenges:

- **Loss of context**: Punctuation can provide useful signals, especially for sentiment analysis or sentence boundaries. For instance, exclamation marks indicate excitement or surprise, and removing them may result in loss of emotional tone.
- **Affects abbreviations and acronyms**: Removing punctuation indiscriminately can distort text, e.g., "U.S." becoming "US" (different meanings in different contexts).

Challenge: Deciding which punctuation to remove and which to keep to preserve the meaning of the text.

5. Stemming and Lemmatization

Drawbacks and Challenges:

- **Over-simplification (Stemming)**: Stemming is often too crude. It might cut off too much of the word, making the result less interpretable (e.g., "studying" becoming "studi"). The stemmed word may not even be a real word.
- **Performance overhead (Lemmatization)**: Lemmatization requires a dictionary lookup to find the root form, which can be computationally expensive.
- **Language variance**: Stemming and lemmatization rules are language-specific. What works well in one language might not generalize to another.

Challenge: Finding the balance between simplicity (stemming) and accuracy (lemmatization), especially when working with large datasets or multi-lingual corpora.

6. **Removing Special Characters**

Drawbacks and Challenges:

- **Loss of important information**: Special characters like hashtags (#), mentions (@), or even emojis may carry semantic value, especially in social media data. For example, #AI could indicate that the text is discussing artificial intelligence.
- **Context-dependence**: What's considered a "special character" depends heavily on the context of the task. Removing all non-alphabetic characters can be problematic for text that involves code, numbers, or specific domains (like mathematical texts).

Challenge: Careful decision-making is needed to determine what to remove and what to keep based on the application.

7. Text Normalization

Drawbacks and Challenges:

- **Domain dependency**: Expanding contractions or standardizing abbreviations might depend on the specific domain. For example, “AI” in a casual conversation might need to be expanded to "artificial intelligence," but in a different setting, abbreviations are acceptable.
- **Over-normalization**: Excessive normalization can result in loss of valuable information. For instance, converting "10k" to "ten thousand" might alter the data format in ways that models are not trained to handle.

Challenge: Designing the normalization rules to avoid overly simplifying text or distorting its meaning.

8. N-grams

Drawbacks and Challenges:

- **High dimensionality**: As the value of "n" increases (bigrams, trigrams), the number of possible combinations increases exponentially, which can lead to a very sparse dataset (a large number of zero values).
- **Context fragmentation**: N-grams capture only local context and miss long-distance dependencies between words in a sentence. For instance, in “I want to go to New York City,” bigrams like "go to" may lose connection to the rest of the phrase.

Challenge: Balancing between small "n" values (which might miss context) and large "n" values (which may increase computational complexity).

9. **Encoding Text**

Drawbacks and Challenges:

- **Bag of Words (BoW)**: This method ignores the order of words entirely, which means it loses context. For example, "I love dogs" and "Dogs love me" would have the same representation in BoW, though they mean different things.
- **TF-IDF**: While it improves upon BoW by weighing words, it still doesn't capture semantic relationships or word order.

General Challenges Across Steps:

- **Language diversity**: Many of these techniques are designed for English and may not translate well to other languages. For example, tokenization and stemming rules can be very different across languages like German, Chinese, or Arabic.
- **Context dependence**: Certain tasks (like sentiment analysis) require retaining elements like punctuation or stopwords, while other tasks (like topic modeling) benefit from removing them.
- **Computational complexity**: More advanced pre-processing methods (e.g., lemmatization, word embeddings) require significant computational resources, especially for large datasets.