Conversational Systems Design

Lecture 2

BITS Pilani
Pilani Campus

Kedar Kanhere
02/06/2024

Session Content but before that

Quiz and Assignment Tentative Schedule

Exam Type	Start Date	End Date	Marks
Quiz 1	13/07/2024 7:00 PM	14/07/2024 7:00 PM	5 (Best 2 of 3)
Quiz 2	31/08/2024 7:00 PM	01/09/2024 7:00 PM	5 (Best 2 of 3)
Quiz 3	07/09/2024 7:00 PM	08/09/2024 7:00 PM	5 (Best 2 of 3)
Assignment 1	20/06/2024	18/07/2024	15
Assignment 2	25/08/2024	22/09/2024	15

Session Content but before that

Webinar will tentatively be on these dates:

19-Jun-24
10-Jul-24
14-Aug-24
4-Sep-24

Session Content

- Importance of text pre-processing
- Overview of past learnings (NLP)
  - Stemming
  - Tokenization
  - POS Tagging
- Text to Speech
  - Understanding T2S algorithms basic layer
  - Different algorithms working
  - Code and example via different libraries
  - Importance of T2S Synthesis, ASR etc.

Importance of text pre-processing

- **Text pre-processing** is a crucial step in Natural Language Processing (NLP) that involves cleaning and preparing text data for analysis.

- It includes techniques such as:
  - Tokenization
  - Stop Word Removal
  - Stemming
  - Lemmatization
  - Text Cleaning

Importance: These steps help in converting raw text into a more manageable and analyzable format, facilitating better model performance and more accurate results.

Tokenization

- Tokenization is the process of splitting text into smaller units called tokens, which can be words, phrases, or symbols.

  - Example:Input: "Natural Language Processing is fascinating.
  - "Output: ["Natural", "Language", "Processing", "is", "fascinating"]

- Tools
  - NLTK: A leading platform for building Python programs to work with human language data.
  - SpaCy: An open-source software library for advanced NLP in Python.
  - Tokenizer APIs: Available in various programming languages and platforms.

- Importance: Tokenization is the first step in the text pre-processing pipeline and is crucial for the performance of subsequent steps.

Tokenization

Inconsistent Data Representation Without tokenization, text data remains as large, unmanageable strings, making it difficult to analyze and process.

Model Performance: Machine learning models require numerical or categorical input. Without tokenization, converting text into a suitable format is impossible, leading to poor model performance.

Difficulty in Feature Extraction: Tokenization allows for extracting features such as word frequencies, n-grams, and more. Skipping this step hinders effective feature extraction.

Ineffective Text Cleaning: Tokenization is often the first step in text cleaning. Without it, removing stop words, punctuation, and performing stemming/lemmatization becomes challenging.

Error Propagation: Errors in initial steps propagate through the pipeline, leading to inaccuracies in tasks like sentiment analysis, NER, and POS tagging.

Stop-Words

- Stop words are common words that are usually ignored in text processing because they do not carry significant meaning.
  - Examples: Common stop words include "a", "the", "and", "in", "to", etc.
- Purpose:
  - Noise Reduction: Removing stop words helps in reducing the noise in the text data.
  - Efficiency: It reduces the size of the text data, making processing faster and more efficient.
- Tools:
  - NLTK: Provides a predefined list of stop words and functions for their removal.
  - SpaCy: Offers built-in support for stop word removal in various languages.
- Example:
  - Input: "The quick brown fox jumps over the lazy dog."
  - Output: "quick brown fox jumps lazy dog"
- Importance: Removing stop words helps in focusing on the words that are more likely to be significant in the analysis, thereby improving the performance of NLP models.

Stemming and Lemmatization

- Stemming: Reduces words to their base or root form by removing suffixes. It may not always produce a real word.

- Lemmatization: Converts words to their base form (lemma) using morphological analysis. It always returns a valid word.

- Examples:
  - Stemming: "running" -> "run", "jumps" -> "jump“
  - Lemmatization: "better" -> "good", "running" -> "run“

- Tools:
  - NLTK:
    - PorterStemmer for stemming
    - WordNetLemmatizer for lemmatization
  - **SpaCy**: Offers built-in lemmatization capabilities.

Stemming and Lemmatization

- Differences:
  - Accuracy: Lemmatization is generally more accurate than stemming.
  - Complexity: Stemming is simpler and faster but less accurate.
- Use Case: Choose stemming for quick and dirty text processing; use lemmatization for tasks requiring higher accuracy.
- Importance: Both techniques help in normalizing words to their base forms, which reduces the dimensionality of the text data and improves the performance of NLP models.
- Problems if we don’t do stemming or lemmatization:
  - High dimensionality
  - Inconsistent Data
  - Reduced model performance and difficulty in text analysis
  - Lower accuracy in search and retrieval

Case Normalisation

- Case normalization is the process of converting all characters in the text to a uniform case, either lower case or upper case, to ensure consistency.

- Purpose:
  - Consistency: Ensures that words are treated equally regardless of their case.
  - Reduction of Redundancy: Helps in reducing redundancy by treating "Apple" and "apple" as the same word.
- Example:
  - Input: "Natural Language Processing“
  - Output: "natural language processing“
- Tools:
  - Python String Methods: .lower() and .upper()
  - NLTK: Provides functions for case normalization.
  - SpaCy: Built-in support for case normalization.
- Importance:
  - Improves Text Processing: Case normalization simplifies text processing by reducing the number of unique tokens.
  - Enhances Model Performance: Models become more efficient as they deal with fewer variations of the same word.

- Note: Case normalization is particularly useful when the case of the text does not carry significant meaning for the analysis.

Text Cleaning

- Text cleaning involves removing unwanted elements from the text to make it suitable for analysis.
- Common Techniques:
  - Removing Punctuation: Eliminating characters such as periods, commas, and exclamation marks.
  - Removing Special Characters: Removing symbols like #, $, %, etc.
  - Removing Numbers: Excluding digits unless they are relevant to the analysis.
  - Removing HTML Tags: Stripping HTML content from web-scraped text.
  - Handling Contractions: Expanding contractions (e.g., "don't" to "do not").

- Tools:
  - Regular Expressions (Regex): Powerful for pattern matching and substitution.
  - NLTK: Provides functions for various text cleaning tasks.
  - SpaCy: Built-in functions for text cleaning.

- Example:
  - Input: "Hello, world! Visit us at https://example.com #NLP“
  - Output: "Hello world Visit us at example com NLP“

- Importance:
  - Enhances Data Quality: Cleaned text is more consistent and easier to analyze.
  - Improves Model Accuracy: Cleaner data leads to better-performing models by reducing noise and irrelevant information.

Named Entity Recognition (NER)

- Named Entity Recognition (NER) is a subtask of information extraction that identifies and classifies named entities in text into predefined categories such as person names, organizations, locations, dates, etc.
- **Categories**:
  - **Person**: Names of people (e.g., "John Doe")
  - **Organization**: Names of organizations (e.g., "Google")
  - **Location**: Geographical locations (e.g., "Paris")
  - **Date/Time**: Dates and times (e.g., "January 1, 2020")
  - **Others**: Money, percentages, etc.
- **Purpose**:
  - **Information Extraction**: Helps in extracting structured information from unstructured text.
  - **Improved Search and Retrieval**: Enhances the performance of search engines and information retrieval systems.
- **Tools**:
  - **SpaCy**: Provides a pre-trained NER model and tools for custom training.
  - **NLTK**: Offers NER capabilities through its chunking module.
  - **Stanford NER**: A widely used NER tool developed by Stanford University.

Importance: NER is essential for understanding the context and extracting relevant information from large volumes of text, making it a key component in various NLP applications.

Named Entity Recognition (NER)

Rule-based Approaches: Use predefined rules and patterns to identify entities.

- **Example**: Regular expressions to identify dates and email addresses.
- **Pros**: Simple to implement and interpret.
- **Cons**: Limited flexibility and scalability.

Machine Learning Approaches: Use statistical models trained on labeled data to identify entities.

    - Conditional Random Fields (CRF)
    - Hidden Markov Models (HMM)
- **Pros**: More flexible and accurate than rule-based approaches.
- **Cons**: Require labeled training data and computational resources.

Named Entity Recognition (NER)

- **Deep Learning Approaches**: Use neural networks to automatically learn features and patterns from data
    - Recurrent Neural Networks (RNN)
    - Long Short-Term Memory (LSTM)
    - Transformer-based models (e.g., BERT)
  - **Pros**: High accuracy and ability to capture complex patterns.
  - **Cons**: Require large datasets and significant computational power.
- **Example**:
  - **Input**: "Apple is looking at buying U.K. startup for $1 billion."
  - **Output**: Entities: [("Apple", "ORG"), ("U.K.", "LOC"), ("$1 billion", "MONEY")]

Importance:

Choosing the right technique depends on the specific requirements of the task, available data, and resources. Machine learning and deep learning approaches are preferred for their accuracy and scalability.

Part of Speech Tagging (POS)

- Part-of-Speech (POS) tagging is the process of assigning a part of speech to each word in a sentence. The parts of speech include nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, and interjections

  - **Understanding Syntax**: Helps in understanding the grammatical structure of sentences.
  - **Disambiguation**: Resolves ambiguities by providing context to words (e.g., "book" as a noun vs. "book" as a verb).

Example The quick brown fox jumps over the lazy dog."

- POS Tags:
  - The (DT) quick (JJ) brown (JJ) fox (NN) jumps (VBZ) over (IN) the (DT) lazy (JJ) dog (NN)

Tools:

- **NLTK**: Provides a comprehensive POS tagging module.
- **SpaCy**: Offers efficient and accurate POS tagging capabilities.
- **Stanford POS Tagger**: A robust tool developed by Stanford University.

Importance:

POS tagging is fundamental for many NLP tasks such as parsing, text-to-speech conversion, and information extraction. It enables a deeper understanding of the syntactic and semantic properties of text.

Part of Speech Tagging (POS)

- **Rule-based Tagging**: Uses a set of hand-crafted rules to assign POS tags.
  - **Example**: "If a word ends in 'ly', tag it as an adverb (RB).“
- **Statistical Tagging**:Uses probabilistic models based on the likelihood of a word's POS tag given its context.
  - Hidden Markov Models (HMM)
  - Maximum Entropy Models
- **Machine Learning Approaches**: Use supervised learning techniques to predict POS tags.
  - Support Vector Machines (SVM)
  - Conditional Random Fields (CRF)
- **Deep Learning Approaches**:Uses neural networks to automatically learn features from data.
  - Recurrent Neural Networks (RNN)
  - Long Short-Term Memory (LSTM)
  - Transformer-based models (e.g., BERT)

Example:

- Sentence: "The cat sat on the mat."
- POS Tags: The (DT) cat (NN) sat (VBD) on (IN) the (DT) mat (NN)

Sentiment Analysis

- Sentiment analysis, also known as opinion mining, is the process of determining the sentiment expressed in a piece of text. It classifies text into positive, negative, or neutral sentiments.
  - **Understanding Public Opinion**: Helps businesses understand customer opinions and feedback.
  - **Market Research**: Analyzing trends and opinions about products, services, or events.
  - **Social Media Monitoring**: Tracking sentiment trends on social media platforms.

Applications:

- **Customer Feedback Analysis**: Understanding customer satisfaction and improving services.
- **Brand Monitoring**: Tracking public sentiment towards a brand.
- **Political Sentiment**: Analyzing public opinion on political issues or candidates.
- **Product Reviews**: Assessing the sentiment in product reviews to gauge consumer reactions.

Sentiment Analysis

Techniques:

Lexicon-based Methods: Use predefined lists of positive and negative words.

Machine Learning Approaches: Train classifiers (e.g., SVM, Naive Bayes) on labeled data.

Deep Learning Approaches: Use neural networks (e.g., LSTM, BERT) for higher accuracy.

Example:

Input: "I love the new design of your website!"

Output: Positive

Importance:

Sentiment analysis provides valuable insights into the emotions and opinions expressed in text, enabling better decision-making and strategy formulation.

Text to Speech Synthesis

- Text-to-Speech (TTS) synthesis is the process of converting written text into spoken words using computational methods.
- Purpose:
- Accessibility: Provides access to information for visually impaired users.
- User Experience: Enhances user interaction in virtual assistants and conversational agents.
- Automation: Automates tasks that require reading text, such as news reading or announcements.
- Components:
  - Text Analysis:
    - Tokenization: Breaking text into smaller units such as sentences and words.
    - Linguistic Analysis: Determining the part of speech, phonetic transcription, and prosody (intonation, stress, rhythm).
  - Acoustic Modeling:
    - Phoneme Synthesis: Generating speech sounds based on phonetic transcription.
    - Prosody Generation: Adding natural intonation, stress, and rhythm to synthesized speech.
  - Speech Synthesis:
    - Concatenative Synthesis: Combining pre-recorded speech segments.
    - Formant Synthesis: Using mathematical models to generate speech sounds.
    - Waveform Synthesis: Using neural networks to generate high-quality, natural-sounding speech.

Text to Speech Synthesis

- Tools:
  - Google Text-to-Speech: A widely used TTS engine with high-quality voices.
  - Amazon Polly: A cloud-based service that converts text into lifelike speech.
  - IBM Watson Text to Speech: Provides a range of customizable voices.
  - Open Source Tools: eSpeak, Festival, and MaryTTS
- Example:
  - Input: "Welcome to the world of Text-to-Speech synthesis.“
  - Output: (Spoken audio)
- **Importance**: TTS synthesis plays a crucial role in improving accessibility and enhancing user experiences across various applications and devices.

Text Generation Techniques

Text generation involves creating coherent and contextually relevant text from a given input or set of rules.

Key Techniques:

Rule-based Systems : Use predefined rules and templates to generate text.

Example: Fill-in-the-blank templates for automated report generation

Markov Chains : Use probabilistic models based on the likelihood of word sequences.

Example: Generating text by predicting the next word based on the previous one

Recurrent Neural Networks (RNNs): Use neural networks with loops to maintain context over sequences.

Example: Generating poetry or short stories

Long Short-Term Memory Networks (LSTMs) : A type of RNN designed to better handle long-term dependencies.

Example: Generating more coherent paragraphs and articles.

Transformer Models: Use self-attention mechanisms to capture long-range dependencies in text.

Example: GPT-3 generating articles, stories, and dialogue.

State-based and Rule-based Dialogue Systems

Dialogue systems designed to follow specific states or rules to manage interactions with users.

State-based Dialogue Systems : Systems that manage conversations using predefined states and transitions between those states.

Rule-based Dialogue Systems: Systems that use predefined rules and templates to generate responses and manage interactions.

Examples : If user says "Hello", respond with "Hi, how can I help you today?“

If user asks for account balance, respond with "Please provide your account number.“

Components:

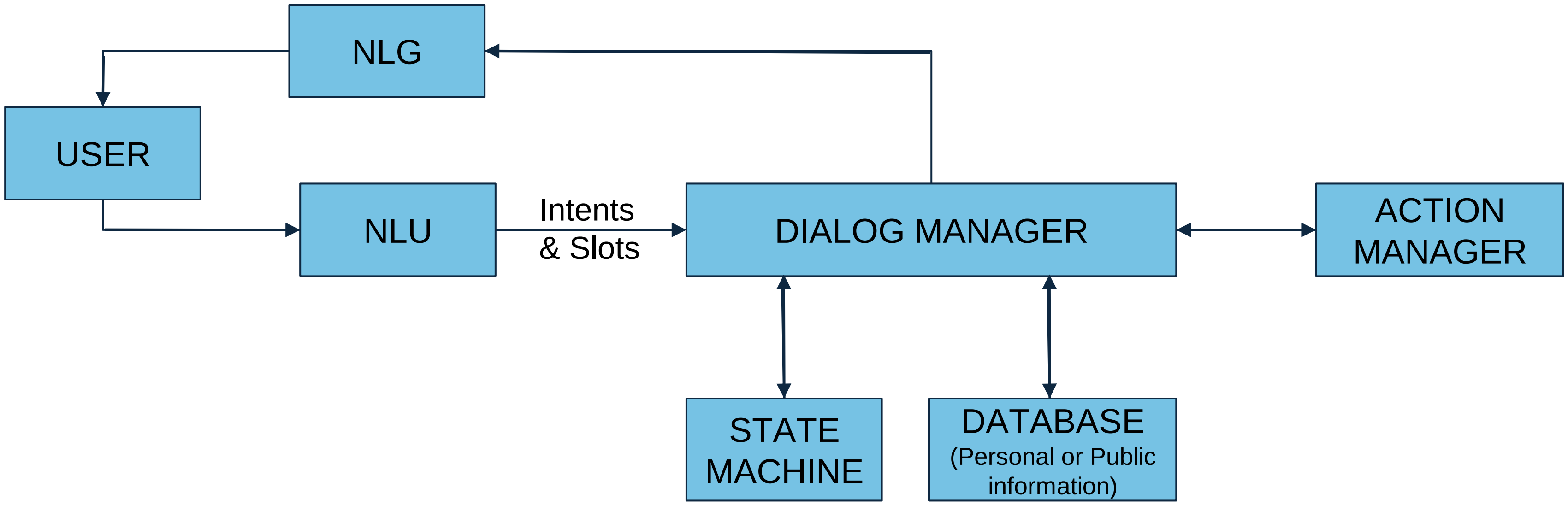
Rule Engine: Processes user input based on predefined rules.

Template Manager: Uses templates to generate responses.

Context Manager: Maintains context of the conversation to apply relevant rules.

State-based and Rule-based Dialogue Systems

State-based Dialogue Systems



Introduction to Intent Recognition and Slot Filling

Intent Recognition: The process of identifying the goal or purpose behind a user's input in a conversation.

Purpose: Helps in understanding user requests and guiding the conversation appropriately.

Example Intents:

- Booking a flight

Slot Filling: Extracting specific pieces of information (slots) from the user's input that are necessary to complete the intent.

Purpose: Provides detailed information required to fulfill the user's request.

Example

- Slots for Flight Booking:
  - Destination
  - Departure
  - DateReturn
  - Date
  - Number of Passengers

Introduction to Intent Recognition and Slot Filling

Components:

- Natural Language Understanding (NLU) Module: Processes user input to recognize intents and extract slots.
- Dialogue Manager: Uses recognized intents and filled slots to manage the conversation flow and fulfill the user's request.

Techniques:

Rule-based Methods: Use predefined patterns and templates to recognize intents and extract slots.

Machine Learning Approaches: Train classifiers on labeled datasets to predict intents and extract slots.

Deep Learning Approaches: Use neural networks, particularly sequence-to-sequence models, to handle more complex and varied inputs.

Example:

User Input: "I want to book a flight to New York on June 5th.“

Recognized Intent: Book Flight

Extracted Slots:

- Destination: New York
- Departure Date: June 5th