Report on Natural Language Processing Projects

1. SMS and Urdu Text Preprocessing

Overview:
This project focuses on preprocessing text data from SMS messages and Urdu news datasets to prepare the text for further NLP tasks such as sentiment analysis and classification. The preprocessing steps include data cleaning, tokenization, stop word removal, normalization, and stemming.

Code Analysis:

Step 1: Importing Libraries
The project utilizes essential libraries including `pandas` for data handling, and `nltk` for natural language processing tasks. The `nltk` library supports text cleaning, tokenization, and filtering of stop words.

Step 2: Loading the Datasets
SMS data and Urdu news content are loaded using `pandas`, with the SMS data typically in tab-separated format and the Urdu news as a CSV file. This stage ensures data is ready for preprocessing.

Step 3: Text Normalization
Text normalization is applied to ensure uniformity. The code converts text to lowercase and uses regular expressions to remove numbers and punctuation, which helps standardize text and reduce data noise.

Step 4: Tokenization
Text is split into individual tokens, i.e., words. Tokenization simplifies further analysis by breaking down text into manageable units.

Step 5: Stop Word Removal
Common stop words, which add little semantic value, are removed. `nltk` provides a predefined list for English SMS data, and a custom list is created for Urdu news text to address language-specific nuances.

Step 6: Stemming
To handle variations of words, stemming reduces words to their root forms, facilitating consistent representation. This is particularly useful in reducing complexity for analysis or model input.

Step 7: N-gram Generation
The project uses n-grams, which combine sequences of words, to capture context. N-grams support higher-level analysis by maintaining word order, helping preserve some syntactic structure.

Step 8: Sequential Data Processing
Each step is sequentially applied to both datasets, resulting in a final processed DataFrame containing columns for normalized text, tokens, tokens without stop words, stemmed tokens, and n-grams.

Output:
The processed DataFrames showcase the progression of each preprocessing step, demonstrating the text's readiness for advanced NLP tasks.

Conclusion:
Preprocessing SMS and Urdu news datasets prepares them for NLP tasks by normalizing, tokenizing, removing stop words, and applying stemming. These steps enhance model interpretability and analysis potential, serving as a foundation for sentiment analysis, classification, or other linguistic tasks.

---

2. Poetry Generation in the Style of Shakespeare and Frost

Overview:
This project aims to emulate the poetic styles of William Shakespeare and Robert Frost using n-gram models. The code loads a poetry corpus, tokenizes the text, builds n-grams, and generates poems based on the models developed.

Code Analysis:

Step 1: Importing Libraries
The project utilizes `nltk` for text processing and `random` for generating random selections, which play a role in selecting words during poem generation.

Step 2: Loading and Tokenizing the Poetry Corpus
The works of Shakespeare and Frost are loaded into memory, and text is tokenized into words, enabling the formation of sequences used in n-gram modeling.

Step 3: N-gram Model Creation
Unigrams, bigrams, and trigrams are created by measuring word co-occurrence frequencies, capturing patterns in word choice and structure for each poet.

Step 4: Choosing Starting Words
To initiate poem generation, a random starting word is chosen from each poet's vocabulary, setting the tone for the generated text.

Step 5: Poem Generation
The model generates lines of poetry by iteratively predicting words based on previous words, guided by n-gram patterns. Line lengths vary, ranging from 7 to 10 words, with stanzas composed of 4 lines each.

Output:
The project outputs generated poems, producing three stanzas per poet, each reflecting their unique stylistic nuances.

Conclusion:
This poetry generation project demonstrates n-gram models' capacity to mimic distinctive poetic styles. By constructing unigram, bigram, and trigram models, the project emulates the linguistic patterns of Shakespeare and Frost, showcasing the creative potential of NLP in stylistic imitation and automated literary composition.