Chloe Tu
ITAI 2373 – Natural Language Processing
Professor: Anna Devarakonda
Date: September 9, 2025

Reflection Journal: Lab 02 Basic NLP Preprocessing Techniques

Introduction:

This lab on basic NLP preprocessing techniques provided a crucial look into the foundational steps required to prepare human language data for analysis by computers. Before this, I hadn't fully appreciated the complexity hidden within seemingly simple text or the necessity of cleaning and standardizing it. Working through different techniques like tokenization, stop word removal, stemming, and lemmatization, and comparing different libraries like NLTK and spaCy, revealed that preprocessing isn't a one-size-fits-all process. Instead, it involves making deliberate choices with significant impacts on downstream NLP tasks, a key insight that shaped my understanding throughout this exploration.

Key insights about text preprocessing and its importance:

A major insight from this lab is understanding how preprocessing fundamentally impacts any NLP task. It's not just a simple cleaning step. It is a critical transformation changing raw human language into a format computer can process. Without it, variations like different capitalizations punctuation or word forms would confuse programs. The "Simple text" example in Step 4 shows this. Subsequent cleaning steps demonstrate how standardizing text reduces noise and vocabulary size. This is crucial for making algorithms efficient and accurate. Steps 8 9 and 17 show how much text is removed. Understanding this initial transformation is key to building reliable NLP systems.

Challenges encountered while working through the concepts:

One significant challenge encountered was realizing there is no single best way to preprocess text. Each technique like tokenization or stop word removal requires decisions with trade-offs. For example, Conceptual Question 4 discussed NLTK's handling of contractions and punctuation. It showed that splitting "It's" works for some tasks but not others where linguistic structure matters. Conceptual Question 8 explored removing all punctuation. I concluded this is not always good. Punctuation is important for sentiment in examples like "This is great!". The messy social media text in Step 6 and its cleaning in Step 14 also presented challenges. Conceptual Question 11 discussed balancing noise removal like URLs with keeping important things like emojis and hashtags. These elements carry sentiment in social media. This highlighted that preprocessing needs careful thought not just standard steps.

Connections made between preprocessing techniques and real-world applications:

The lab clearly connected preprocessing to real-world technology. The comparison of stemming and lemmatization in Steps 10 11 and 12 showed this. Conceptual Question 10 explained how stemming's speed suits search engines. They quickly match word variations like "running" or "ran" to a root like "run". Lemmatization focuses on valid base words like "better" to "good"

shown in Step 12. This makes it better for sentiment analysis where meaning is important. Analyzing different text types in Step 17 showed preprocessing must adapt. Challenges like product review ratings need specific cleaning for real-world systems.

Questions that arose during the exploration of the material:

Several questions came up during the lab pushing for deeper understanding. Conceptual Question 7 looked at NLTK and spaCy's different stop word lists shown in Step 7. This made me wonder how stop words are defined. Is there one perfect list or should they be task specific? I also wondered how preprocessing handles things not covered like negation which is key for sentiment but involves stop words like "not". How about sarcasm or idioms? More advanced techniques might be needed for these complex language features.

Comparisons between different approaches (NLTK vs SpaCy, stemming vs lemmatization):

The lab gave a clear comparison of NLTK and spaCy. NLTK uses individual functions for tasks like tokenization in Step 4 stemming in Step 10 and stop words in Step 7. It feels modular giving fine control. SpaCy uses an integrated pipeline in Steps 5 9 and 11. It gives richer features like POS tags and lemmas automatically. This makes it more efficient for tasks needing deeper analysis. The comparison of stemming and lemmatization in Step 12 showed a clear trade-off. Stemming is faster but can create non-words. Lemmatization uses dictionaries and keeps meaning. The choice depends on whether speed or accurate meaning is more important for the application as discussed in Conceptual Question 10.

Future applications - how you might use these techniques in projects:

In the future I plan to use these preprocessing techniques in various projects. For text classification like categorizing news or analyzing customer feedback I would use a pipeline. It would include cleaning tokenization stop word removal carefully considering if words are needed for the task like negation for sentiment and lemmatization. This prepares the text features. If working with social media for trend analysis or sentiment tracking I would adapt the advanced cleaning from Step 14. I would ensure platform specific things like hashtags or emojis are handled correctly. Understanding these basic preprocessing steps is a necessary base for building effective NLP models for many different applications.

Conclusion

My biggest takeaway from this lab is just how crucial text preprocessing really is for building solid NLP applications. It's a balancing act: you must weigh the pros and cons of each technique, from stripping away the junk to keeping the data that matters most. Speed versus accuracy is always a major decision, and it all depends on the project's goals. Even with the endless quirks of human language, this lab gave me a powerful set of tools. I feel much more prepared to start my next NLP project, understanding that a good outcome starts with a great foundation.