Lab 02 introduced me to the practical foundations of text preprocessing, and I now realize how critical this step is in any NLP (Natural Language Processing) pipeline. Before this lab, I underestimated just how much cleaning and organizing raw text affects the quality of machine learning models. Through working with both NLTK and SpaCy, I discovered that the way text is prepared can significantly change the outcomes of any analysis—whether it's sentiment detection, keyword extraction, or chatbot responses.

One of the key insights I gained is that text preprocessing is not a simple checklist, but a strategic step that must be tailored to the problem you're solving. For instance, removing stopwords and punctuation seems straightforward, but in certain use cases (like poetry analysis or legal documents), those might carry meaning. This made me more thoughtful in applying preprocessing tools rather than just doing them “by default.”

A challenge I encountered was choosing between stemming and lemmatization. At first, they seemed interchangeable, but I quickly learned that stemming can often produce non-intuitive results (like cutting "running" into "run" or even "runn") while lemmatization is more precise, turning words into their dictionary forms. SpaCy’s built-in lemmatizer felt much more natural and was easier to work with than NLTK’s Porter stemmer, which sometimes produced confusing outputs.

One surprising connection I made was imagining how I could apply preprocessing to my own projects—especially my AI startup ideas and even in automating tasks on my mushroom farm. If I wanted to build a tool that processes customer reviews or farmer inquiries, proper text preprocessing would be the first and most vital step. It ensures clean inputs, which means fewer errors downstream.

Between NLTK and SpaCy, I found SpaCy to be more intuitive, efficient, and powerful. While NLTK is good for educational purposes and has a broader selection of tools, SpaCy feels more professional and robust. It processes text faster, and its lemmatization, named entity recognition, and part-of-speech tagging features are built-in and easy to access. If I were building a real-world AI application, I would almost certainly start with SpaCy.

One question that came up for me during the lab was how text preprocessing handles multilingual data or domain-specific language (e.g., medical or agricultural jargon). I’d love to explore how customizable these preprocessing pipelines are, especially when training models in non-English languages or specialized fields.

Looking ahead, I plan to use what I’ve learned in my Generative AI startup pitch, especially when working with prompts and user input. Preprocessing can help standardize data before it's sent to a model—making it easier to ensure that the model interprets the input correctly and responds in a way that makes sense. I also see how this applies to blockchain verification of text-based prescriptions, one of my earlier coding projects, where tokenization and text cleaning would help extract structured data from unstructured doctor notes.

Overall, Lab 02 helped me understand that text preprocessing is not just a technical step, it’s a form of language awareness. It reminds me that machines don’t “read” the way humans do. We have to carefully prepare text so that our algorithms can make sense of language, which is often messy, inconsistent, and full of ambiguity.