Faiza Abdullah

Lab 02

ITAI 2373 Natural Language Processing (NLP)

Professor: Patricia Mcmanus

Reflective Journal: Lab 02 - Basic NLP Preprocessing Techniques

Key Insights on Text Preprocessing and Its Importance

One of the most striking realizations from this lab was how much "invisible" work text preprocessing does in making raw text usable for NLP systems. I hadn't fully appreciated how messy human language is—full of contractions, emojis, and inconsistent formatting—until I saw how preprocessing standardizes it into something algorithms can handle. The lab showed me that preprocessing isn't just a preliminary step but a critical decision-making process that shapes downstream outcomes. For instance, choosing whether to remove stop words or keep emojis can make or break a sentiment analysis model. This insight deepened my understanding of why preprocessing is the foundation of NLP, as it directly influences the quality of features fed into machine learning models. The idea that "no universal solution" exists was particularly eye-opening, pushing me to think critically about tailoring preprocessing to specific tasks rather than applying a one-size-fits-all approach.

Another key takeaway was the trade-off between information preservation and noise reduction. I initially thought cleaning text meant stripping away as much as possible, but the lab highlighted how over-aggressive cleaning, like removing hashtags in social media text, can discard valuable context. This nuanced perspective has shifted how I view data preparation, emphasizing the need to balance simplicity with richness to capture the essence of the text.

Challenges Encountered

One significant challenge was grappling with the error in Step 4, where NLTK's word_tokenize and sent_tokenize failed due to a missing 'punkt_tab' resource. The original code assumed the 'punkt' package was sufficient, but running it threw an error, which was frustrating because the lab was fully coded to focus on concepts, not debugging. Fixing it by adding nltk.download('punkt_tab') taught me about the dependencies underlying NLTK's tokenization and the importance of ensuring resource availability. This hiccup, while minor, underscored the practical realities of working with NLP libraries, where setup issues can derail analysis if not addressed.

Conceptually, I struggled with deciding when to use stemming versus lemmatization. The lab's comparison showed stemming's speed but also its errors, like "studies" → "studi," while lemmatization produced valid words like "study." However, understanding when these errors are tolerable versus when accuracy is critical was initially confusing. It took reflecting on task-specific needs—like search engines prioritizing speed versus sentiment analysis needing precision—to clarify this. This challenge highlighted the complexity of preprocessing choices and the need for a deeper grasp of task requirements.

Connections to Real-World Applications

The lab's diverse text types (simple, academic, social media, news, reviews) made it easy to see how preprocessing applies to real-world scenarios. For example, the social media text with emojis and hashtags (e.g., "OMG! ☕ #coffee") connected directly to social media analytics, where companies like Starbucks might analyze tweets to gauge customer sentiment. Removing emojis or hashtags, as shown in advanced cleaning, could strip away emotional or topical cues, potentially

misrepresenting user feedback. This made me think about how platforms like Twitter (or X) rely on preprocessing to filter noise while preserving sentiment for brand monitoring.

Another connection was to chatbots, like those used in customer service. The lab's exploration of stop word removal showed how words like "not" are crucial for understanding intent (e.g., "not happy" vs. "happy"). This resonated with my experience using chatbots that sometimes misinterpret queries, likely due to poor preprocessing. These connections underscored how preprocessing decisions ripple through to user-facing applications, affecting their reliability and effectiveness.

Questions That Arose

Several questions emerged during the lab. First, how do practitioners decide the "right" preprocessing pipeline in practice? The lab emphasized task-specific choices, but real-world datasets are often messier and less predictable than the sample texts. Are there standard heuristics or automated tools to optimize these decisions? Second, how do modern NLP models, like transformers, handle preprocessing differently? The lab focused on traditional techniques, but I wonder if newer models reduce the need for manual preprocessing or require different approaches. Finally, the social media text cleaning raised a question about multilingual data: how do emojis or slang translate across languages, and what preprocessing challenges arise there? These questions reflect my curiosity about scaling these techniques to more complex, real-world contexts.

Comparisons Between Approaches

Comparing NLTK and spaCy was a highlight of the lab. NLTK's tokenization was simpler and faster but less robust, splitting "It's" into three tokens ("It", "'", "s"), which felt overly aggressive for tasks like sentiment analysis. spaCy, with its language model, kept "It's" intact and provided rich features like POS tags and lemmatization, making it more suitable for industrial applications.

However, spaCy's reliance on en_core_web_sm felt heavier, and I could see NLTK's lightweight nature being preferable for quick prototyping or resource-constrained environments. This comparison taught me that spaCy is likely better for production systems where accuracy matters, while NLTK suits educational or experimental settings.

Stemming versus lemmatization was another compelling comparison. Stemming's speed was appealing, but its errors, like "better" → "better" instead of "good," were jarring. Lemmatization's accuracy, mapping "better" to "good," felt more reliable, especially for tasks requiring semantic understanding. However, lemmatization's dependence on spaCy's model made it slower, which could be a bottleneck in real-time systems. This trade-off crystallized the importance of aligning preprocessing with task goals—speed for search, accuracy for sentiment.

Future Applications

I am excited about applying these techniques in future projects. One idea is building a sentiment analysis tool for customer reviews, using spaCy's lemmatization and minimal cleaning to preserve sentiment cues like "not" or exclamation marks. The lab's standard pipeline would be a great starting point, ensuring key terms like "fantastic" are normalized without losing context. Another potential project is a topic modeling system for social media posts, where I would retain hashtags and emojis (avoiding advanced cleaning) to capture trends accurately. These projects would leverage the lab's insights on tailoring preprocessing to task needs, balancing noise reduction with information retention.

I also see myself using these techniques in chatbot development. The lab's focus on stop words and tokenization highlighted their role in understanding user intent, which I could apply to create a more responsive chatbot for a specific domain, like tech support. Experimenting with spaCy's POS tagging to parse user queries could enhance the bot's ability to handle complex inputs. These

applications feel achievable thanks to the lab's clear demonstration of preprocessing workflows, giving me a solid foundation to build upon.

Conclusion

This lab was a transformative experience, shifting my perspective from viewing preprocessing as a mundane task to recognizing it as a strategic process that shapes NLP outcomes. The challenges, like the Step 4 error and conceptual trade-offs, pushed me to think critically about implementation and decision-making. The real-world connections to social media and chatbots made the techniques feel relevant and impactful, while my lingering questions inspire me to explore further. Comparing NLTK/spaCy and stemming/lemmatization clarified their strengths, equipping me to make informed choices in future projects. I'm eager to apply these insights to build robust NLP systems, confident that this lab has given me a strong starting point.

References (Used in Notebook)

https://spacy.io/usage/models

https://spacy.io/usage/linguistic-features#tokenization

https://www.nltk.org/api/nltk.tokenize.html

https://spacy.io/usage/linguistic-features#lemmatization

https://www.nltk.org/api/nltk.stem.html

https://spacy.io/api/language#defaults

https://www.nltk.org/api/nltk.corpus.html

https://spacy.io/usage/processing-pipelines