Faiza Abdullah

Lab 05

ITAI 2373 Natural Language Processing (NLP)

Professor: Patricia Mcmanus

Reflective Journal: Lab 05 - Part-of-Speech Tagging in the Real World

Lab 05 was a deep dive into Part-of-Speech (POS) tagging, a foundational NLP process that assigns grammatical roles (e.g., noun, verb, adjective) to words in a sentence. This lab illuminated how POS tagging underpins many real-world applications, from disambiguating phrases like "Apple stock" versus "apple pie" in search engines to powering autocorrect and chatbots. A key insight was understanding how POS tagging enables machines to parse sentence structure, making it critical for tasks like sentiment analysis, information extraction, and machine translation. The hands-on exploration of NLTK and SpaCy revealed their complementary strengths: NLTK's fine-grained Penn Treebank tags are ideal for linguistic research, while SpaCy's intuitive Universal Dependencies tags excel in production environments.

Another significant takeaway was the challenge of handling messy, real-world text. Informal language, slang, and disfluencies (e.g., "Um", "ain't") often confuse taggers, as seen in the Messy Text Challenge. However, I was surprised to find that both NLTK and SpaCy achieved 100% success rates in my Colab environment, likely due to newer library versions handling informal terms like "Um" as INTJ or UH instead of X. This highlighted the impact of model updates on performance and the importance of version control in NLP workflows. The lab also emphasized the trade-offs between tagger speed, accuracy, and robustness, particularly when processing diverse text types like social media posts or technical jargon.

Challenges Encountered

The lab presented technical challenges that deepened my understanding of NLP tools and debugging. In Activity 1 (Your First POS Tags), I encountered a familiar issue from Lab 02: a missing NLTK resource error (Resource punkt_tab not found). The notebook initially required nltk.download('punkt'), but tokenization failed until I added:

```
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
```

This resolved the issue, allowing NLTK to tokenize and tag the sentence correctly. This experience reinforced the importance of ensuring all necessary NLTK resources are downloaded, especially in Colab environments where dependencies may not be pre-installed.

In Lab Exercise 1 (Messy Text Challenge), I initially expected some words (e.g., "Um", "ain't", "smh") to be tagged as X (unknown), as informal text often challenges taggers. However, both NLTK and SpaCy reported 100% success rates, with no words tagged as X. After investigation, I found that my Colab environment used NLTK 3.9.1 and SpaCy 3.8.7 with en_core_web_sm, which are newer than older versions where "Um" might be tagged as X. For example, SpaCy tagged "Um" as INTJ (interjection) and "ain't" as VERB or PART, while NLTK tagged them as UH and VBP, respectively. This discrepancy, as noted in the notebook, arises from updates in tokenization algorithms and model training, highlighting the need to document library versions in NLP projects.

In Activity 3 (The Ambiguity Challenge), adding a new example sentence ("The record was broken during the race.") worked seamlessly, with "record" correctly tagged as NN (noun) by NLTK and NOUN by SpaCy. This success underscored the taggers' ability to handle context-driven ambiguity

when syntactic clues are clear, but it also prompted me to reflect on more complex cases where context is insufficient.

No major errors occurred in other exercises, but the visualization in Lab Exercise 3 (Tagger Performance Benchmarking) required careful adjustment of plot sizes to ensure readability, teaching me the importance of tailoring visualizations for clarity. Overall, these challenges enhanced my debugging skills and emphasized the need to verify library versions and dependencies.

Connections to Real-World Applications

POS tagging's real-world relevance became vivid through the lab's exercises. In Lab Exercise 2 (Customer Service Analysis), analyzing call transcripts for sentiment and urgency showed how POS tagging can prioritize customer service tickets. For example, extracting adjectives like "terrible" or "excellent" and urgency indicators like "immediately" enables automated routing of urgent complaints to specialized teams. This connects to my experience with customer service chatbots, where misinterpreting urgency can frustrate users, highlighting the need for accurate POS tagging.

The Messy Text Challenge mirrored social media analysis, where platforms like X process posts with slang and emojis (e.g., "lol", "□♀□"). POS tagging helps classify posts as positive or negative, aiding brand sentiment analysis. For instance, tagging "awesome" as ADJ or "smh" as NOUN can inform sentiment scores, which companies use to gauge public opinion. This application resonates with my observation of brands responding to customer feedback on social media.

In Lab Exercise 3 (Tagger Performance Benchmarking), testing taggers on technical text (e.g., "API", "JSON") showed SpaCy's strength in handling jargon, relevant to technical documentation

or chatbot systems for IT support. For example, a chatbot for a cloud service provider could use POS tagging to identify nouns like "API" and verbs like "deploy" to understand user queries. The lab's edge cases (e.g., "Buffalo buffalo") also highlighted limitations in legal or academic text analysis, where ambiguity could lead to misinterpretation.

Questions That Arose

The lab sparked several questions about scaling POS tagging to complex scenarios. First, how do practitioners choose between NLTK and SpaCy in large-scale systems with diverse text types? The lab showed SpaCy's robustness for informal text, but what about domain-specific corpora like medical or legal texts? Second, how do contextual models like BERT improve POS tagging for ambiguous cases like "buffalo" or "can"? I'm curious about their ability to leverage broader context compared to static taggers. Third, how can we mitigate biases in POS taggers, especially for non-standard dialects or languages underrepresented in training data? These questions motivate me to explore neural taggers and multilingual NLP.

Comparisons Between Approaches

Comparing NLTK and SpaCy was a key learning point. NLTK's Penn Treebank tags (e.g., DT, JJ, VBZ) are detailed, distinguishing between singular and plural nouns (NN vs. NNS), making it ideal for linguistic analysis. However, its technical tags are less intuitive for beginners. SpaCy's Universal Dependencies tags (e.g., DET, ADJ, VERB) are simpler and more readable, better suited for production systems. In Activity 2, SpaCy correctly tagged "brown" as ADJ, while NLTK tagged it as NN, showing SpaCy's superior handling of adjectives in some contexts.

The Penn Treebank vs. Universal Tagsets comparison in Activity 4 highlighted trade-offs between granularity and simplicity. Penn Treebank's detailed tags (e.g., VBD for past tense verbs) are useful for fine-grained analysis, but Universal's broader categories (e.g., VERB) are easier to map across

languages or tasks like text classification. For a search engine, I would choose Universal tags for their simplicity, as seen in the lab's tag distribution analysis.

In Lab Exercise 3, SpaCy outperformed NLTK on informal and technical text, tagging "lol" as INTJ and "API" as NOUN, while NLTK occasionally misclassified technical terms (e.g., "COVID-19" as JJ). However, NLTK was faster for small texts, a trade-off critical for real-time applications. These comparisons taught me to select taggers based on task requirements: NLTK for academic precision, SpaCy for practical robustness.

Future Applications

It was interesting and challenging at the same time to apply POS tagging to a sentiment analysis tool for social media posts. Using SpaCy to extract adjectives and verbs, as in Lab Exercise 2, I could classify posts as positive or negative, helping brands monitor customer sentiment on platforms like X. For example, tagging "awesome service" as ADJ NOUN could contribute to a positive score. I'd preprocess slang using techniques from Lab Exercise 1 to improve accuracy.

Another project could be a technical support chatbot for a software company. By leveraging SpaCy's POS tagging to identify nouns like "bug" or "feature" and verbs like "fix" or "install," the chatbot could route queries to appropriate teams. The lab's benchmarking exercise showed SpaCy's strength with technical jargon, making it ideal for this use case.

Finally, I envision a text summarization tool for news articles, using POS tagging to identify key nouns and verbs for concise summaries. The lab's customer service analysis pipeline provides a foundation, and I would integrate dependency parsing to enhance context understanding. These projects feel achievable after the lab's practical exercises.

Conclusion

Lab 05 was a transformative exploration of POS tagging, revealing its role as the "grammar police" of NLP. Overcoming challenges like the punkt_tab error and understanding version-driven differences in tagger performance strengthened my technical skills. The real-world connections to customer service, social media, and technical support made the concepts tangible, while comparisons between NLTK, SpaCy, and tagsets clarified their trade-offs. My questions about neural taggers and bias mitigation inspire further exploration, and the lab's hands-on approach equips me to build practical NLP solutions. This experience has solidified my understanding of POS tagging's role in NLP pipelines and motivated me to create impactful, ethical AI applications.

References (Used in Notebook)

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

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

https://pandas.pydata.org/docs/

https://seaborn.pydata.org/

https://matplotlib.org/stable/users/index.html

https://spacy.io/usage/linguistic-features#pos-tagging

https://universaldependencies.org/