**Peen Treebank Tagset Exercise**

Abuchi Godswill Okeke

School of Computer and Information Sciences, University of the Cumberlands

MSAI532 - M51 Natural Language Processing

Dr. Toni Farley

March 08, 2025

**Methodology**

In this exercise, I wrote a Python script that automatically identifies potential tagging errors by comparing manually provided tags with those generated by NLTK’s POS tagger. The script processes sentences with words tagged in the "word/tag" format. It splits each sentence into tokens and tags, then reconstructs the sentence. Next, it uses NLTK’s tokenizer and POS tagger to automatically generate predicted tags for the sentence. The provided tags are then compared to the predicted tag.

**Results**

**Table 1.0**

*One Potential Tagging Errors Detected in each of the Sentences Provided*

|  |  |
| --- | --- |
| **Sentences** | **Potential tagging errors** |
| 1. I need a flight from Atlanta. | * Potential tagging error for word '**Atlanta**':  **provided tag** = *NN***, predicted tag** = *NNP* |
| 1. Does this flight serve dinner. | * Potential tagging error for word '**dinner**':  **provided tag** = *NNS*, **predicted tag** = *NN* |
| 1. I have a friend living in Denver. | * Potential tagging error for word '**have**':  **provided tag** = *VB*, **predicted tag** = *VBP* |
| 1. Can you list the nonstop afternoon flights. | * Potential tagging error for word '**Can**':  **provided tag** = *VBP*, **predicted tag** = *MD* |

Table 1.0 shows that in the first sentence, the word "Atlanta" was incorrectly tagged as NN (common noun). Proper nouns referring to specific entities, such as cities, should be tagged as NNP (proper noun, singular). Misclassifying "Atlanta" as a common noun impacts tasks that rely on accurate identification of named entities (Taylor et al., 2003).

In the second sentence, the word "dinner" was incorrectly tagged as NNS (plural noun). The correct tag should be NN (noun, singular or mass) because "dinner" refers to a singular entity in this context. Incorrect tagging obscures the intended grammatical meaning (Taylor et al., 2003).

The third sentence contains a tagging error with the verb "have," incorrectly marked as VB (verb, base form). The appropriate tag is VBP (verb, non-3rd person singular present), aligning with the subject "I." VB typically appears in infinitive or imperative forms, which are not applicable here.

Finally, in the fourth sentence the modal verb "Can" was incorrectly tagged as VBP. According to the Penn Treebank guidelines, modal verbs expressing ability or permission should be tagged as MD. Mislabeling "Can" fails to recognize its modal function, affecting sentence interpretation. These examples demonstrate the necessity of accurate POS tagging for effective syntactic analysis and language processing (Taylor et al., 2003).

**Conclusion**

This exercise helped me understand how Python and NLTK can be used to check for tagging errors in sentences. By comparing manual tags with the ones predicted by the POS tagger, I saw how mistakes can happen, especially with proper nouns, verb forms, and modal verbs. It showed me that while automated tools are useful, they aren’t perfect and still require human review. I also learned how context plays a big role in tagging accuracy. Overall, this was a great way to see the challenges of natural language processing and why improving tagging models is so important.

<https://github.com/Buchiexplores/MSAI-532/blob/main/week9/pos_tag_analysis.py>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Python Script\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*import* nltk

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

*# List of sentences with provided tokens and tags in "word/tag" format*

sentences = ["I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN",

"Does/VBZ this/DT flight/NN serve/VB dinner/NNS",

"I/PRP have/VB a/DT friend/NN living/VBG in/IN Denver/NNP",

"Can/VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS"]

*for* sentence *in* sentences:

*# Split into individual word/tag pairs and extract tokens and tags*

tokens\_with\_tags = sentence.split()

provided\_tokens = []

provided\_tags = []

*for* token *in* tokens\_with\_tags:

word, tag = token.rsplit('/', 1)

provided\_tokens.append(word)

provided\_tags.append(tag)

raw\_sentence = " ".join(provided\_tokens)

*# Use NLTK to tokenize and tag the reconstructed sentence*

tokens = nltk.word\_tokenize(raw\_sentence)

predicted\_tags = nltk.pos\_tag(tokens)

errors = []

*for* (provided\_word, provided\_tag), (predicted\_word, predicted\_tag) *in* zip(zip(provided\_tokens, provided\_tags), predicted\_tags):

*if* provided\_tag != predicted\_tag:

errors.append((provided\_word, provided\_tag, predicted\_tag))

print(f"Sentence: {raw\_sentence}")

*if* errors:

*for* error *in* errors:

print(f"Potential tagging error for word '{error[0]}': provided tag = {error[1]}, predicted tag = {error[2]}")

*else*:

print("No tagging errors found.")

print()

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Reference**

Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn Treebank: An overview. In Treebanks (pp. 5–22). Springer.