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

Table 1.0 shows that the word “Atlanta” is a proper noun referring to a specific location, so it should be tagged as NNP rather than NN. The use of NN does not correctly mark “Atlanta” as a unique entity, which is essential for accurate named entity recognition. In the second sentence, “dinner” functions as a mass noun in this context, so it should be tagged as NN rather than NNS. These discrepancies show how context can affect the tagging of words with multiple potential roles (Taylor et al., 2003).

For the third sentence, the verb “have” should be tagged as VBP to indicate a non-third-person singular present verb, but it was marked as VB. Finally, in the fourth sentence, the word “Can” acts as a modal verb indicating ability or permission, so it should be tagged as MD rather than VBP. This misclassification fails to capture its modal function, which is critical for understanding the question's intent (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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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"]

*# Process each sentence*

*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)

*# Reconstruct the raw sentence for automatic tagging*

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.