NLTK functions and the command input NLP

Consider the two most probable user cases, one is that the command is regarded as a sentence, close to the natural language the player uses – then the input string would usually have a capital letter at the top and a period at the end; another is that the command is regarded as a mechanical spell – then the user is simply typing a series of letters that strictly matches the predefined COMMAND_SET, no capital form, no punctuations. The two cases would lead to different possible input that requires different ways of processing, and then different user interactions with the game. Since this is still a very small project, I mainly followed the first assumption and chose my processing strategy.

The input text processing only happens after the enter key is pressed.

The processing can be grouped as 4 steps:

> Convert to lowercase -> tokenize -> lemmatize -> compare with the predefined COMMAND_SET

The lowercase conversion is in not always necessary -- it depends on if we want to apply strict rules to the upper/lowercases for players to type, so in my function I left a variable to choose between, and currently the conversion is set true. Beside the reason that I assume it is more likely to be complete sentences, it is just in general easier for processing to have the conversion in the beginning as far as I know, so I kept it this way. However, this would add too much freedom, as some insane letter combinations would also work (e.g.”MoVE leFT”), which is not the most wanted results.

There is also a concern with punctuations. Now I have chose to omit all punctuations just for the simplicity, but it also adds unwanted freedom in the input when “move left…?” is also working.

Tokenize and lemmatize are done with NLTK’s functions. I assumed there might be variations in natural language, for instance, “move left”, ”moving left”, and “move leftward” means the same, and lemmatizing in theory should help me decrease such variance. However, the results didn’t turn out so well, because the WordNetLemmatizer by default takes “left” as the past tense of “leave”. I tried to integrate part-of-speech tags to help fix the issue, but it didn’t work in the end. So now the lemmatizing only lemmatize the nouns, in other words a pure resource waste in my code.

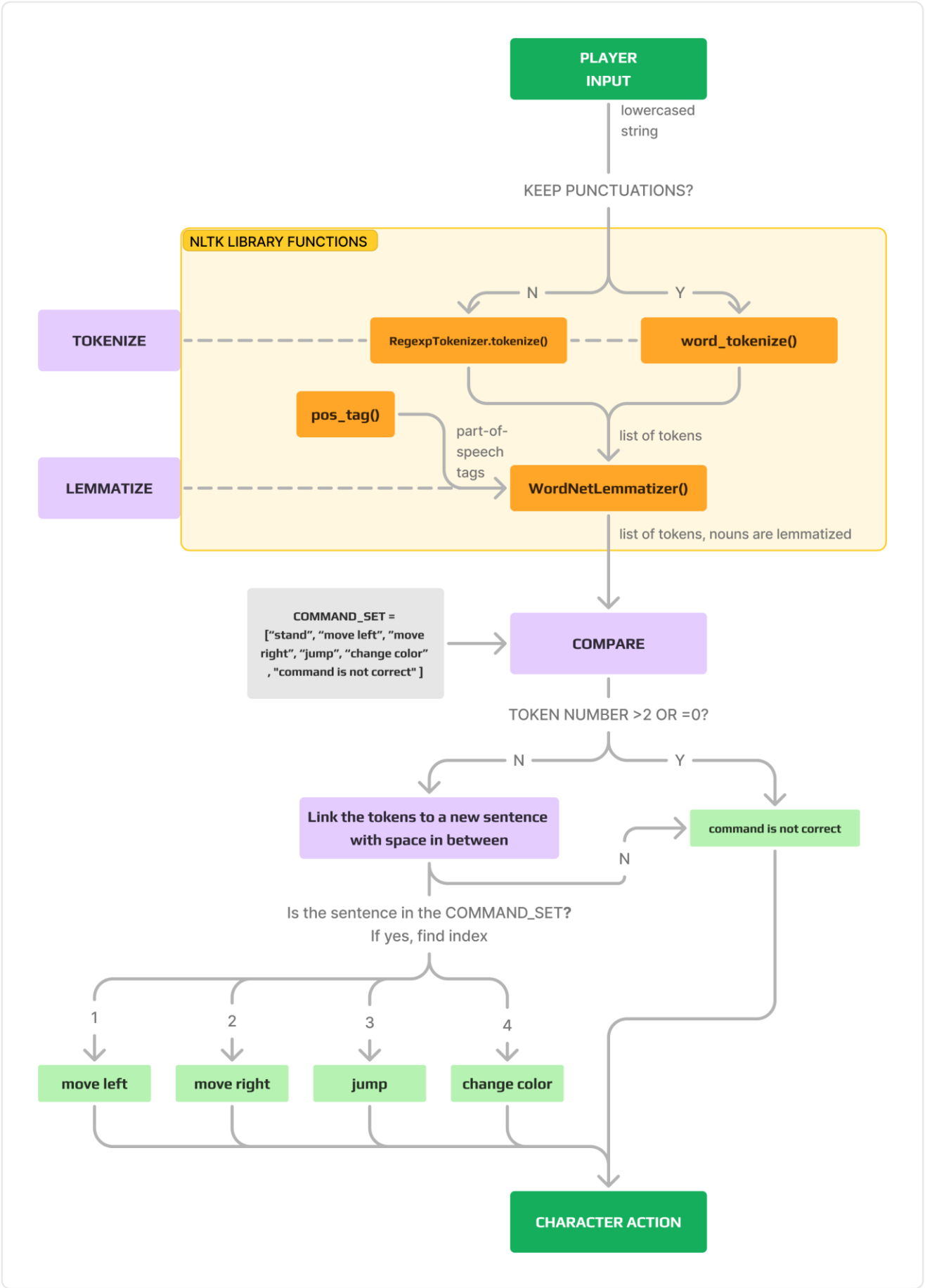
The last step is done with a simple string comparison, after I re-link the lemmatized tokens as a sentence.

I would say the actual NLP involved in my game demo is not much, since the commands are still few and simple, but I do have considered some directions I can go to, if the user case is more certain. Here is what I have thought about:

1. Ambiguity
   Just like the “move left”, ”moving left” and “move leftward” example, handling ambiguity is always a key in semantics. It could start with simply calculating a similarity between the user input and the COMMAND_SET; It could also be done with more complicated AI models that have word or sentence embeddings.
2. Correction and hint
   For now I only return a notification saying “command is not right” if the input matches none in the COMMAND_SET. In a real scenario, there should be a hint if the mis-match happens because of typos or ambiguity. The user should be able to see something like “Do you mean ‘move left’?” and change to the correct input.

Anyway, this is my first Pygame project ever, and I had a great time making and learning new things in three days. The maze was made by me from scratch (the tiles are free assets online), and I made it to have a little sprite animation. As a game enthusiast, I still see many things improvable game-wise speaking. For example, I don’t have an animation for action failure, but just text notification saying “unable to move”; The goals up there are not very tempting and I don’t have a point system; The game/story setting isn’t clear just through the game window itself, and I have to tell people how to play it… But yeah, it was fun. 😊

Section 1


PLAYER INPUT
lowercased string
KEEP PUNCTUATIONS?
NLTK LIBRARY FUNCTIONS
N
Y
TOKENIZE
RegexpTokenizer.tokenize()
word_tokenize()
pos_tag()
part-of-speech tags
list of tokens
LEMMATIZE
WordNetLemmatizer()
list of tokens, nouns are lemmatized
COMMAND_SET = ["stand", "move left", "move right", "jump", "change color" , "command is not correct"]
COMPARE
TOKEN NUMBER >2 OR =0?
N
Y
Link the tokens to a new sentence with space in between
command is not correct
N
Is the sentence in the COMMAND_SET? If yes, find index
1
2
3
4
move left
move right
jump
change color
CHARACTER ACTION