Text to Image Generation

Abhishek Sah (B15CS001), Akash Gupta (B15CS003)
Mentor: Dr. Chiranjoy Chattopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur

Motivation: To gain hands on experience with how to design a Natural Language Processing system which solves some well-defined problem in CS. The project addresses a real world scenario which can be further extended for large uses in text to graphics fields or AIs.

Objective: To create an application which takes as input a textual description of a geometric scene and in natural language produces the image. (Field: NLP and Graphics)

Example Input: "*Draw a green color circle of radius 50cm. Also there is a rectangle of 100cm width and length 50cm and its color is red. The circle is at 150cm top of the rectangle.*"

Output:

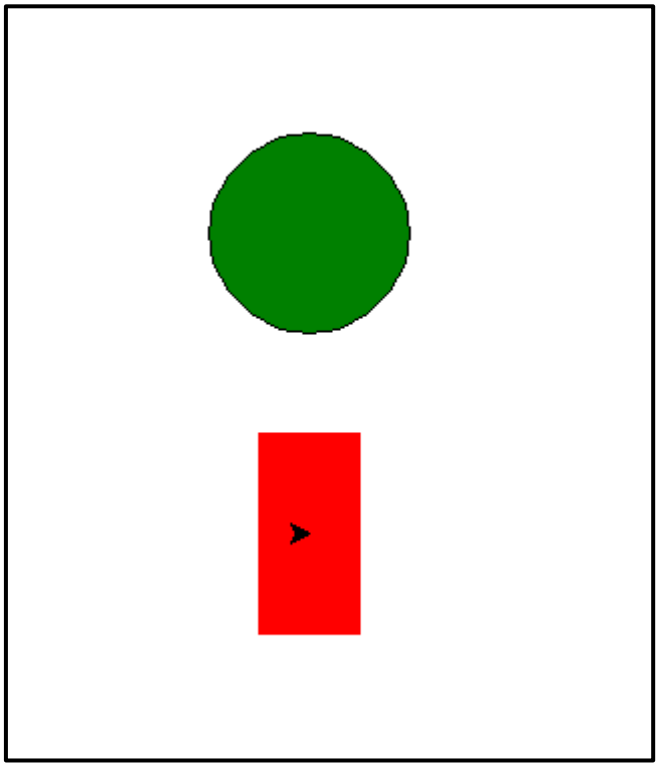

Fig 1

Challenges:

- Finding the attributes of the shapes and mapping those attributes to shapes.
- POS (Parts of speech) Tagging, getting a correct unambiguous chunking grammar to extract the chunks to the best accuracy.
- Drawing the actual shapes relative to each other and to the scale.

Methodology/Algorithm: There were 2 parts of the problem. The first part was parsing the input text to find out what shapes are there and their attributes and relative position. The second part was to draw those shapes to the scale.

For part one, we employed 4 methods for parsing (**Python NLTK**) and analysed their results,

1. String Matching: A simple keyword search algorithm which works well only if input text is of a fixed pattern. (See: Fig 1)

2. POS Tagging (without context): It tags the words in input text according to Parts of Speech definition. The tagged texts were used for further chunking using a grammar rule that we defined. Although context was not taken into consideration, this approach failed where input text had ambiguous chunks. (See Fig 2 and 3)

3. POS Tagging (with context): It tags the words in input text taking context into consideration. The POS form of a word may change according to the context. We tried Bigram Tagger, Trigram Tagger and HMM Tagger of the NLTK library. The results were not impressive.

4. **String Matching + POS Tagging**: It gave the **best accuracy**. We applied POS tagging without context and on top of it string matching was performed. The chunking grammar was updated accordingly to get the best accuracy. (See Fig 4)

For part two, we used the **Python Turtle** library to sketch the shapes, fill the color and place them at correct relative position.

Results:

- **String matching algorithm** was pretty much rule based. It had very strict assumption on input text. If those assumptions were followed, it gave the best accuracy.

- **POS Tagging** relaxed a few assumptions on input text but it failed on ambiguous POS chunks. Ambiguity here refers to 2 chunks having same composition of tags.

  A few cases where it produces correct result and a failing scenario.

1. **Input**: *Draw a green color circle having dimensions of radius 50cm.*
   Output:

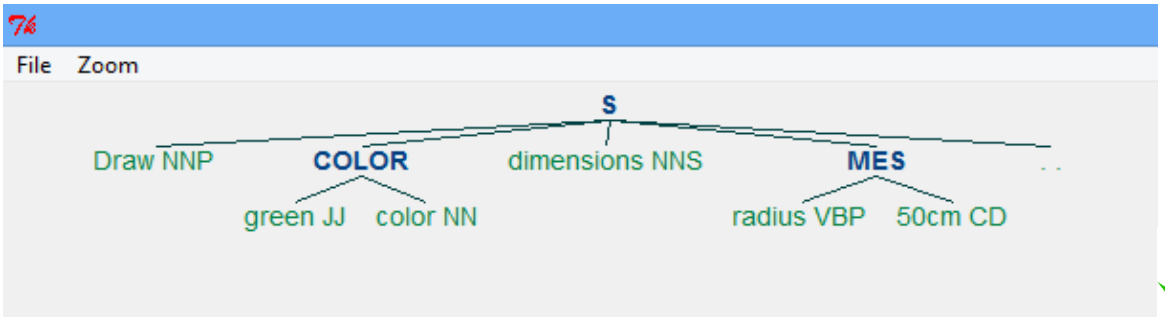

Fig 2

2. **Input:** *Draw a circle of green color and 100cm in radius.* (Ambiguous as per grammar definition)
   Output:

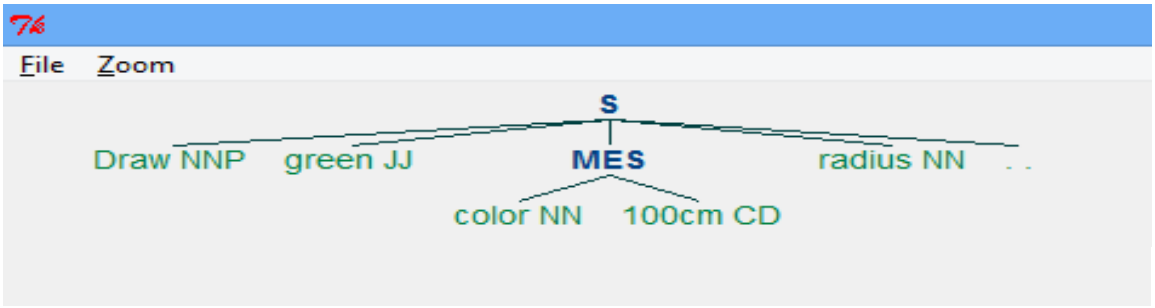

Fig 3

- String matching on top of POS Tagging gave the **best accuracy**. It had very fewer assumptions on the input text and resolved ambiguities as well.

  Input: Draw a circle of green color and 100cm in radius
  Output:

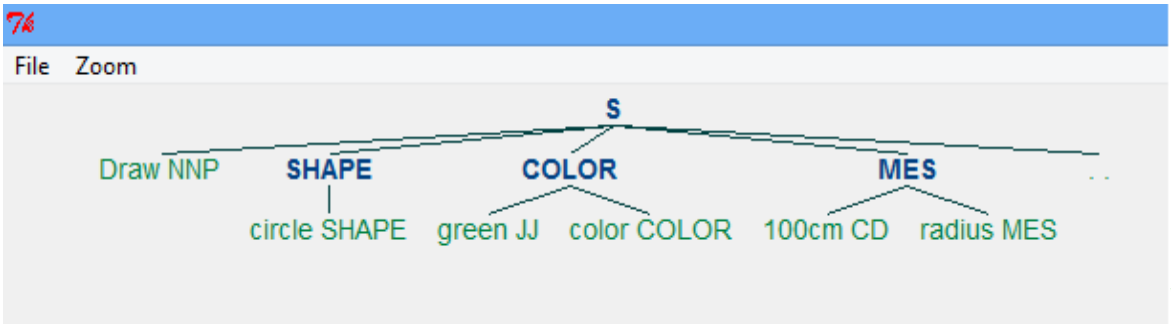

Fig 4

Conclusion:

- The complexity of the problem of parsing text and extracting useful information grows as the scope of the problem. String matching where gives best results restricts the input text to be of some fixed format.
- No POS tagger is "the best". It all depends on the dataset on which it is being trained.
- In the domain of our problem statement, String matching on top of POS tagging works quite efficiently.

The complete project lives here: https://github.com/abhishek-iitj/Text-To-Image-

References:

1. http://www.nltk.org/book/
2. https://docs.python.org/2/library/turtle.html

Date: Mentor's signature