Smart Interaction

Prof. Irfan Siddavatam, Tappan Ajmera, Gaurav Agarwal, and Mohit Pattni.

***Abstract*—Almost all teaching nowadays is done using presentations to ensure better understanding of the concepts through images and animations. And yet explaining certain things becomes difficult. The images and the animations are only sufficient to explain what they are depicting. Smart Interaction is a system that is capable of generating the said images dynamically, using voice input from the user. Our system is developed on Visual Studio 13. It uses python script that uses Google Application Program Interface to parse voice to text and uses Stanford CoreNLP trained model for sentiment analysis and lexical parsing of the question. We have currently limited our domain to a basic set of straightforward 2D geometrical problems having only one shape per question (limited to square, circle, rectangle and triangle).**

***Keywords*—Natural Language Processing, Stanford CoreNLP, Visual Studio, python, geometry, shapes, voice input.**

I. Introduction

We live in a time where we are moving from chalkboard teaching to one we call modern teaching. Institutions have now started incorporating computers or laptops with Wi-Fi connections in the classroom, LCD projectors, interactive whiteboards. These modern methods make it interesting for the students with the help of images, animations and videos. Also, research shows that visual media helps the students in not only understanding the concept better but also in retaining it for a longer period of time. These modern methods are not only advantageous for the students, but also for the teachers. It saves a lot of time not having to write on the board; syllabus is covered faster. Also, visuals are more explanatory[1].

However, in spite of all the advantages above, one must realise that a lot of time is spent in creating these visuals and these visuals are limited to what they are created for. A question regarding an aspect not a part of the visual means resorting back to the board or creating a new visual. Considering all this it would be more convenient if the visuals could be created as per need, specific requirement and be modified on the go.

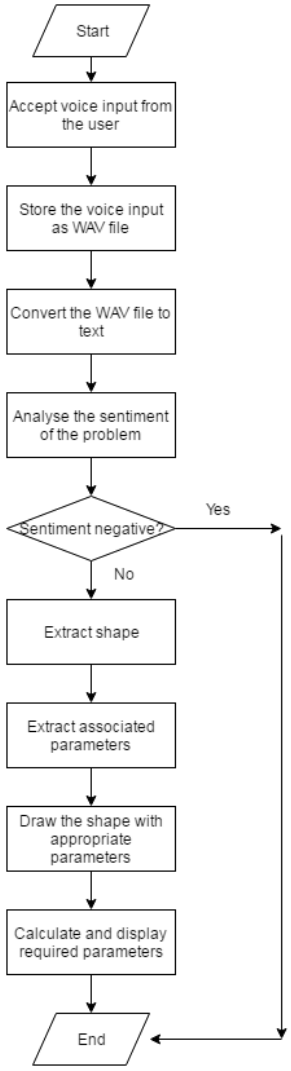
In this paper we describe our system which is aimed at solving the problems mentioned above. A system that takes in voice input from the user which is parsed to text and then lexically parsed to give the shape and its attributes mentioned in the problem.

To be able to make such a system we had to able make the machine infer the meaning of the said sentence by having a grammatical and semantic framework in place which could characterize the language in ways enabling computationally tractable implementations of semantic and syntactic analysis [3]. We used Stanford CoreNLP framework for this purpose. It has a trained semantic and syntactical model but also allows us to train ours over it.

In Section II we talk about our approach. Section III has the experimental results and section IV consists of our concluding remarks.

II. Our Approach

This section has the flowchart of our system followed by the explanation of the modules that make up the system.



A. Speech to Text

It is well understood that speech recognition by a machine is a complex problem. Various factors like accent, pronunciation, speed and volume matter. Thus we needed a speech engine that was robust to noise, accurate and fast. Our natural choice was

the Google Speech Engine which provided us all these features. Their new updates are based on neural network accoustic models that use Connectionist Temporal Classification(CTC) and sequence discriminative training techniques. These models are a special extension of recurrent neural networks (RNNs) that are more accurate, especially in noisy environments, and they are blazingly fast![4]

However, the issue with directly integrating Google's API was no language model or vocabulary configuration, limited to 50 requests per day and no commercial support. Hence we found a workaround. We used Python SpeechRecognition 3.3.3 which had a support for Google's speech API.

We use NAudio library for C# to record the audio as a Waveform Audio File Format(WAV).Then using the process class of C# we invoke Python code which transcribes this WAV file using Google Speech Engine.

This is much more faster and accurate than directly sending the audio input to the server. It gives us complete control of when to start and stop recording and makes it easier for the user too.

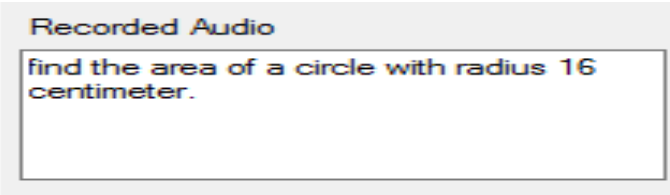


Fig. 1: Recording Audio

B. Sentiment Analysis

The next step was understanding if the user actually wanted the figure drawn. Thus the need for sentiment analysis.

We trained our own set of 125 problems on Stanford CoreNLP's sentiment analysis model. Each word was tagged with a sentiment value, 2 being neutral, 1 slightly negative, 0 very negative, 3 slightly positive and 4 very positive. Each of the words in the problem were then combined into phrases according to the Sentiment Treebank structure required by the sentiment analysis engine. These phrases were also given a sentiment value and were similarly combined to get the original problem into the Treebank structure. The Treebank structure for 'I do not like circle so draw a square'
(2 (2 (1 (1 (2 I) (1 (2 do) (1 (1 not) (3 like)))) (2 circle)) (2 (2 so) (2 (2 draw) (2 (2 a) (2 square))))) (2 .))

Fig. 2: Sentiment Value

This dataset was then used to train the Recursive Deep Learning model of Stanford CoreNLP repeatedly until we achieved a maximum accuracy as follows:

```
Approximate Negative label accuracy: 0.994792
Approximate Positive label accuracy: 0.896552
Combined approximate label accuracy: 0.972000
Approximate Negative root label accuracy: 1.000000
Approximate Positive root label accuracy: 1.000000
Combined approximate root label accuracy: 1.000000
```

Fig. 3: Sentiment Model Accuracy

The reason for using such a small dataset was our narrow domain. The variety of the words used in formulating the problems in our domain is limited; sufficing our current need for accuracy.

C. Extracting the shape

Each word in the problem is compared with the list of keywords - the shapes - defined in an array. If we find a hit, the sentence is sent for further processing. Else, the process is terminated.

D. Association of shape parameters with the given values

We made different classes for each shape, each having a dictionary containing the possible parameters for a specific shape like sides, lengths, diagonals, area, perimeter, circumference etc. Our algorithm for achieving association is as follows :

1) Tokenize the given sentence using PTBTokenizer.
2) Store every word of the sentence along with its dependencies in an ArrayList.
3) Parse the ArrayList to find a word that matches the property in the dictionary of the associated shape class.
4) If step 3 is true then parse its dependencies till a numerical value is found and update the value in the dictionary.
5) Else search for the next keyword match till the end of the ArrayList.

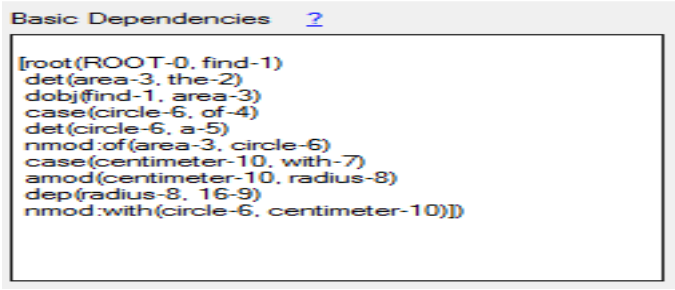


Fig. 4: Basic Dependencies

```
 done [1.0 sec].
(ROOT
  (S
    (VP (VB find)
      (NP
        (NP (DT the) (NN area))
        (PP (IN of)
          (NP
            (NP (DT a) (NN circle))
            (PP (IN with)
              (NP
                (ADJP (JJ radius)
                  (NP (CD 16)))
                (NN centimeter)))))))
    (. .)))

root(ROOT-0, find-1)
det(area-3, the-2)
dobj(find-1, area-3)
case(circle-6, of-4)
det(circle-6, a-5)
nmod:of(area-3, circle-6)
case(centimeter-10, with-7)
amod(centimeter-10, radius-8)
dep(radius-8, 16-9)
nmod:with(circle-6, centimeter-10)
```

Fig. 5: Basic Dependency Tree

E. Drawing the images

Our application is a Windows Form Application. This enabled us to use the Graphics class in C# to draw our images. The parameters needed for drawing were simply extracted from the dictionary and passed to the drawing function after scaling the values. Our system is able to draw diagonals of the square and rectangle. For the circle we are able to draw radius, diameter, and a chord subtending a central angle. Our drawing of the triangle is currently limited to a static isosceles triangle whose display value of height and base is changed as per the problem.

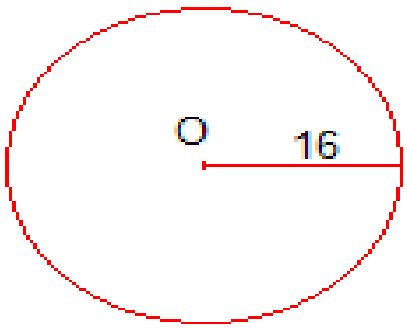


Fig. 6: Drawing Output

III. Experimental Results

Here we talk about our experimental results module-wise.

A. Speech to text

As mentioned in section II we had to use the python speech engine for voice recognition. Initially, we recorded the voice and sent it directly to the Google servers for transcribing. However, we had no control over the listening time. It would sometimes start listening before the user spoke or sometimes after the user started speaking leading to record the problem partially. In figure 7 we see that it recorded the word 'square' only when the the user said 'draw a square'.

```
C:\Users\Neepa>python -m speech_recognition
A moment of silence, please...
Set minimum energy threshold to 4960.31937593
Say something!
Got it! Now to recognize it...
You said hello
Say something!
Got it! Now to recognize it...
You said draw a circle
Say something!
Got it! Now to recognize it...
Oops! Didn't catch that
Say something!
Got it! Now to recognize it...
You said square
Say something!
Got it! Now to recognize it...
Oops! Didn't catch that
Say something!
```

Fig. 7: Python audio recording problem

Hence we use the WAV file to record the audio. This enables us to have 100% control over the recording start and stop time. It also makes it easier on the user. The speed of the system also increased as the server did not have to wait for the initial silence timeout or end silence timeout to understand that speech had been recorded.

B. Sentiment Analysis

The Stanford CoreNLP sentiment model was based on movie review excerpts from the rottentomatoes.com website originally collected and published by Pang and Lee(2005). It did not give us correct sentiment that we required for processing the sentence further. For e.g., the sentence 'draw a circle with radius 20 cm and find its area' was given an overall negative sentiment. We realised we needed to train their model on our data set. And so we trained it on the 125 sentences as mentioned before.

```
Draw a circle with radius 20 cm and find its area.
  Negative
```

Fig. 8: Wrong sentiment analysis before training

```
Draw a circle with radius 20 cm and find its area.
  Neutral
```

Fig. 9: Correct sentiment analysis after training

The sentiment recognition accuracy for our domain improved after training their model on our dataset as is evident from figure 9. The sentiment could be neutral or positive for the problems that needed a figure drawn. It only should not be recognised as negative.

C. Association of shape parameters with the given values

While performing tests we realised that often based on the structure of the sentence, the values were not always directly associated to the corresponding parameter. Once we incorporated that into our code, we found that as long as the problem was built correctly in English, CoreNLP could make dependencies direct or indirect, and our system could extract them.

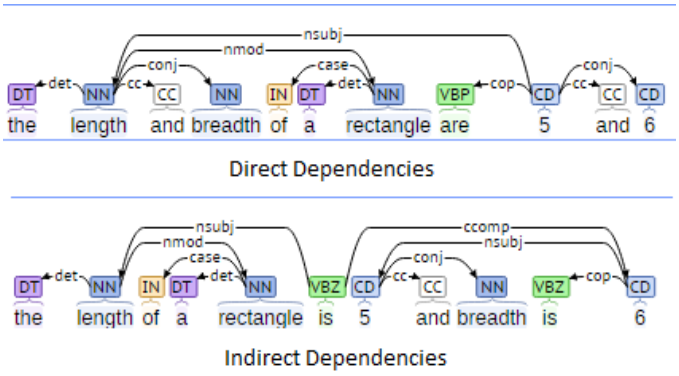


Fig. 10: Direct and indirect dependencies

IV. Conclusion

In this paper we have presented Smart Interaction, a voice controlled system to enhance teaching. Our system focuses on how natural language dialogue and intelligent systems can support human learning across educational contexts including within and outside the classroom. The goal here has been to make the system as simple and user friendly as possible. We realize that there is a lot of scope for improvement that the system can be made a lot more dynamic and interactive.

References

[1] ”Teaching with Modern and Traditional Methods.” IndiaStudyChannel.com. Thakur Aman, 13 Nov. 2011. Web. 07 Mar. 2016.¡http://www.indiastudychannel.com/resources/146615-Teaching-with-modern-and-traditional-methods.aspx¿.

[2] Manning, Christopher D., et al. ”The Stanford CoreNLP Natural Language Processing Toolkit.” ACL (System Demonstrations). 2014.

[3] Schubert, Lenhart, ”Computational Linguistics”, The Stanford Encyclopedia of Philosophy (Spring 2015 Edition), Edward N. Zalta (ed.), URL = ¡http://plato.stanford.edu/archives/spr2015/entries/computational-linguistics/¿.

[4] Sak Hasim, Andrew Senior, Kanishka Rao, Franoise Beaufays, and Johan Schalkwyk. ”Google Voice Search: Faster and More Accurate.” Google Voice Search: Faster and More Accurate. 24 Sept. 2014. Web. 20 Mar. 2016.

[5] Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. ”Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.” Web. 21 Mar. 2016.

[6] Swamy, Suma, and Ramakrishnan K.v. ”An Efficient Speech Recognition System.” Computer Science & Engineering: An International Journal CSEIJ 3.4 (2013): 21-27. Web.

[7] Vrinda, and Chander Shekhar. ”SPEECH RECOGNITION SYSTEM FOR ENGLISH LANGUAGE.” International Journal of Advanced Research in Computer and Communication Engineering 2.1 (2013). Web.

[8] Boyer, Kristy Elizabeth, Robert Phillips, Amy Ingram, Eun Young Ha, Michael Wallis, Mladen Vouk, and James Lester. ”Characterizing the Effectiveness of Tutorial Dialogue with Hidden Markov Models.” Intelligent Tutoring Systems Lecture Notes in Computer Science (2010): 55-64. Web.