README

I. Introduction

In A2, we have two main classes, Chat.java and Bot.java. In A3, changes has been made to improve the conversation flow of the bot by adding additional two classes that implements Stanford toolkit openNLP and Netbeans API as language processing toolkits. Further information about additional implementations will be explained at the end of this README file.

For more information and to view the complete program, visit https://github.com/pgunawan/ChatVader_v2.

II. Assignment 2 Java Classes

Our first class Chat.java:

- a BufferedReader
- a throw IOException

The BufferedReader in this class is for reading the input from the user.
When the user gives the bot an input, it puts the input into lower case therefore it can match our questions.

The exception is ignoring any errors from running the code.
When the input reads bye, the program will stop.
Extra code in this class is so we can include database into our program, however it is not in use at this moment.

Our Second class Bot.java:
- Hi Method
- Where Method
- How Method
- What Method
- Why Method
- When Method
- Question Method
- Generic Method

This class acts as a database answer for our chatbot.
First, we separated our Bot.java in 8 different methods.
When the user asks questions of generic opening that begins with Where, How, What, Why, and When
we will reply with few selected answers.
While the user gives an input of a statement, our Generic method will give the user a random answer or maybe a quote from Darth Vader
Finally if the bot receives an input question that does not begin with the Where, How, What, Why, When, the bot will call the Question method and provide an answer.

III. Additional Features (Assignment 3)

A few changes and implementations had been made since A2. Below is the list of changes that were made:

1) Natural Language Processing (NLP) methods were implemented in the class file synonyms.java and tagger.java.

a. synonyms.java

This class was coded by implementing Wordnet and java API, specifically called the RiTa API. The class provides a method to return a list of synonyms of a certain word, and recognize users' input with similar meanings as I currently have in the code. This method is also used to recognize user's greetings. So instead of listing all possibilities of greetings, the method *synonyms.syn(input, "hello")* can be called, and it will recognize user's input that has the synonym of the word "hello".

Conversation example with synonyms:
<User> Hi
<Vader> Greetings, what brings you here?
<User> Why do you wear a cape?
<Vader> Because it looks cool.
<User> Why do you wear a cloak?
<Vader> Because it looks cool.

As seen from the sample output above, ChatVader recognized the word "cape" and "cloak" as synonyms, and so it gives the same response.

b. tagger.java

This class was coded by implementing Stanford toolkit, OpenNLP called the POS tagging. POS tagging will break down user's input per word and tag the words with part of speeches. So this class provides a method that will recognize part of speech of every word the user inputs. For example, the sentence "Hello world." will look like this after it is tagged: "Hello/UH world/NN." UH stands of interjection, and NN stands for noun. This particular toolkit is useful to improve the bot's conversation by recognizing verbs, adjectives, and nouns and give output based on the user's tagged input.

Conversation example with POStagger:
<User> howdy!
<Vader> Greetings, what brings you here?
<User> tell me a joke!
<Vader> Can I use my POWER to do that?
<User> sing for me!
<Vader> I can do it better than you!

The code now recognizes nouns and exclamation points, and can give outputs according to that.

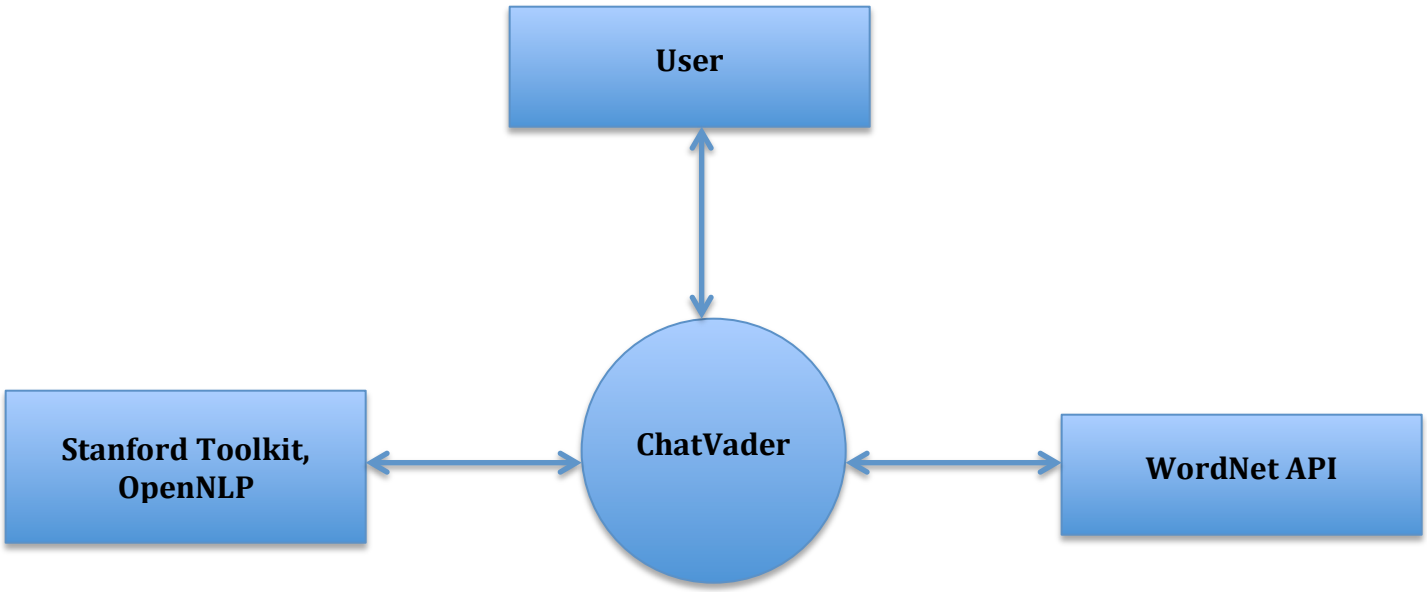
2) Minor changes on the Bot.java class

On last assignment, Bot.java class has all the possible inputs and outputs all written in one single class, which makes the class too long and unnecessary. However, I removed the overflowing switch statements for the responses and put them in a text file instead. Buffered reader was used in the method to extract lines of output from the text file. This way, more output can be integrated without making the Bot.java class any longer.

3) One extra topic and five different outputs were added to the conversation. The one extra topic was coded at the end of Bot.java class. And the five outputs that handles conversation outside of the topic was added in generic.txt.

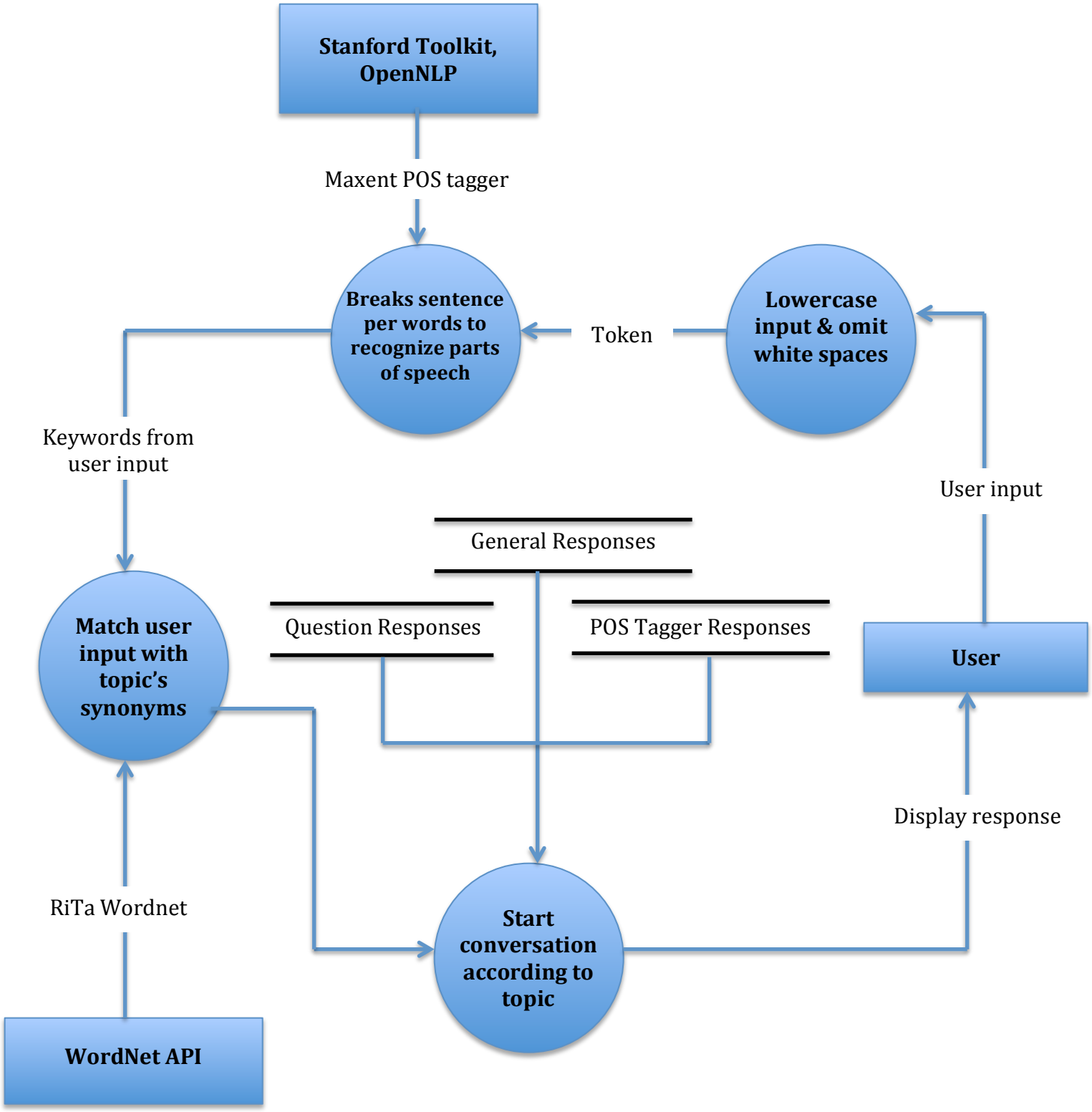
IV. DFD

Level 0



The Level 0 DFD provides a brief yet high level overview of the program. The main program is ChatVader that will handle user input, and parse it and process the input using Stanford toolkit and WordNet API. Both Stanford toolkit and Wordnet are Natural Language processing. Their job is to parse the input and search for a certain word that is needed in the program system.

Level 1


Stanford Toolkit, OpenNLP
Maxent POS tagger
Breaks sentence per words to recognize parts of speech
Token
Lowercase input & omit white spaces
Keywords from user input
User input
General Responses
Question Responses
POS Tagger Responses
Match user input with topic's synonyms
User
Display response
RiTa Wordnet
Start conversation according to topic
WordNet API

Level 1 DFD provides more details on what is going on in the backend of the program. The program starts when the user inputs a sentence into the chat agent. User input will be considered and firstly, the program will ignore uppercase letters and omit white spaces. The process continues and Maxent POS tagger will parse the sentence and tag every single word in the sentence with the appropriate part of speech. This part of speech tags is useful when the program began to collect keywords from the input and match them with synonyms that was implemented through RiTa Wordnet API. The synonyms will send information to the database where all possible answers were stored, such as 'question responses', 'POS tagger responses', and 'general responses'. This process will finalize the code and return the appropriate string back to the user.

V. Github

For more details on Github documentation, visit
https://github.com/pgunawan/ChatVader_v2.

VI. Extractable Features

Below is the list of features that can be extracted from the code:

1) POS tagging user input via Stanford toolkit OpenNLP
2) Recognize synonyms from user via RiTa Wordnet Java API
3) File reader

VII. Program Limitations

1) Agent is only able to handle conversations in English
2) Bot can only handle conversation on certain topics.
3) Synonym recognition can only match the user's input with 50 synonym words. So if the user type anything that is the synonym of the word, but the word itself is not in the synonyms list from the API, then bot will not be able to recognize the word.
4) Agent does not have a database, so it cannot learn.
5) Agent get confused when user type combinations that agent does not recognize, for example, combining questions with exclamation sentences.

VIII. Sample Outputs

Output with 30 turns:
<User> hullo
<Vader> Hello, what brings you here?
<User> the cosc assignment...
<Vader> Give yourself to the Dark Side. It is the only way you can save your COSC 310 grade. Yes, your thoughts betray you.
<User> can you help me?
<Vader> Do you really think I have time to answer this questions?
<User> write me the programming part!
<Vader> I can do it better than you!

<User> teach me openNLP!
<Vader> That is easy!
<User> teach me!
<Vader> I can do it better than you!
<User> then do it!
<Vader> DO IT YOURSELF!
<User> you know, I am your creator
<Vader> DON'T LIE TO ME!
<User> I AM your inventor
<Vader> Impressive. Most impressive. Obi-Wan has taught you well. You have controlled your fear. Now, release your anger. Only your hatred can destroy me.
<User> why so angry?
10<Vader> You do not question a Sith lord!
<User> by the way, how do you use the bathroom?
<Vader> Man diapers... this stays between us right?
<User> why do you have to wear a mask?
<Vader> It looks cool.
<User> and why the cloak?
<Vader> It looks cool.
<User> sing me a song!
<Vader> I'm not good at it... can you give me a hand?
<User> yes
<Vader> Agreeing with me is always a good thing!
<User> where are you from?
<Vader> Tataouine, it's beautiful during the winter.
<User> where is that?
<Vader> Doesn't matter! It can be wherever I want!
<User> who is your father?
<Vader> This information cannot be revealed to the weaklinks like yourself..
<User> why?
<Vader> The question is not why, but how?
<User> can you tell me a joke?
20<Vader> Alright...Let me catch me catch my breath first.
<User> why are you so evil?
<Vader> I need to get my revenge.
<User> revenge to who?
<Vader> This information cannot be revealed to the weaklinks like yourself..
<User> help me with cosc
<Vader> You must be lost!
<User> yes
<Vader> I thought so.
<User> help me with java apis. i am so lost
<Vader> Can you give me a hand?
<User> you're the one who should give me a hand!
<Vader> I have my power to do that. Unlike you..
<User> do it!

<Vader> I cannot help the weaklings like you..
<User> does it matter?
<Vader> Why do you ask?
<User> when do you think you have time to talk again?
<Vader> Greetings, Time is just an illusion.
<User> bye
<Vader> Good day sir.

Bad output 1:
<User> hello
<Vader> Greetings, what brings you here?
<User> the cosc assignment
<Vader> I bet you are lost!
<User> i am!
<Vader> DO IT YOURSELF!
<User> chill, dude
<Vader> Greetings, what brings you here?

The problem with this output is that it recognize chill as a form of greeting from the synonyms list.

Bad output 2:
<User> hello
<Vader> Greetings, what brings you here?
<User> you know, i am your creator!
<Vader> I'm not good at it... can you give me a hand?

The problem with this output is that the agent recognizes a verb combined with an exclamation point so it will give the output provided for this combination.