Kevin Starmack

CS 5130

Project Proposal

2-05-2021

TEXT IN GREEN ARE ADDITIONS TO PROPOSAL BASED ON 2-2-2021 DISCUSSION

Bridging the Gap: Natural Language and Test Cases, Helping write tests through Requirements

Introduction

Many software projects start off with stakeholders needs and desires being brought to discussion before any actual development occurs. Stakeholders needs and desires are known as requirements.

Sometimes software engineers will be given a large list of what is expected of them, however not every piece of information written down is necessarily needed to fulfil the stakeholders needs and desires.

In this project, I hope to automate the procedure of writing test cases through the method of parsing valid requirements and extracting the valuable information needed to write said test case.

The very practice of writing test cases helps prepare the testing team by ensuring good test coverage across the application but writing test cases has an even broader impact on quality assurance and user experience.

Background

A background in requirements engineering and knowledge of Natural Language Processing (NLP) is a key component to being able to understand the project.

Preliminary Results

Having done some research in the field of NLP in the past, I already have good data and benchmarks to better test the effectiveness of my tool.

The Project

In this project, I will need to first do three things:

(1) Be able to parse a document and extract any and all requirements
(2) Further, I must find requirements in which a test case can be developed
(3) From these requirements, I must provide: the precondition, the assumption, and the test steps

Feasibility

This project will be feasible within the semesters timeframe, as it builds upon past work as well as its complexity is not too difficult.

Merit

This project / software tool will help teams and engineers design better test cases at a far quicker speed than just by reading manually what their requirements are.

High-Level Diagram

In Fig1, this high-level diagram will decide if a phrase or text is a requirement going through the process of “keywords” and temporal logic. Once we know what our requirements are from a given paragraph, document, etc., we can then move onto what is happening in Fig2.

Fig2 shows us that once a phrase is deemed a requirement, it will be sent to a local database when it can then go further analysis so that the test cases get written probably. As of right now, I am thinking of using the sentence before and after the requirement in order to get this info. Unsure on how to implement. The database will hold key words and phrases seen in the requirements.

With all of this in mind, I can begin to start developing my requirements, as well as collecting and researching data so that there are benchmarks to evaluate my tool down the line, such as, the requirements and the expected temporal logic of a sample size as such. The requirements will be diversified in order to evaluate the robustness of the tool. I plan to initially use Medical and educational documents with lots of temporal sentences and requirements to first try it out when developing begins.

I plan to start off with exploring NLTK, as my jumping off point, as well as researching the Stanford Typed Dependency Parser further, from the past experience I have had. It is a very useful tool.

I plan to generate abstract test cases, however if I cannot. I will move onto using the executable test cases.

Machine Learning with supervised learning will be used in this project, and a frequency. I will see which one is the best and most reliable of the two.

I will need to research and plan deeper into creating robust test cases from my requirements. By milestone 1, I must have the training data ready, my requirements to test and an idea of more tools to use

See Fig1 and Fig2 below:

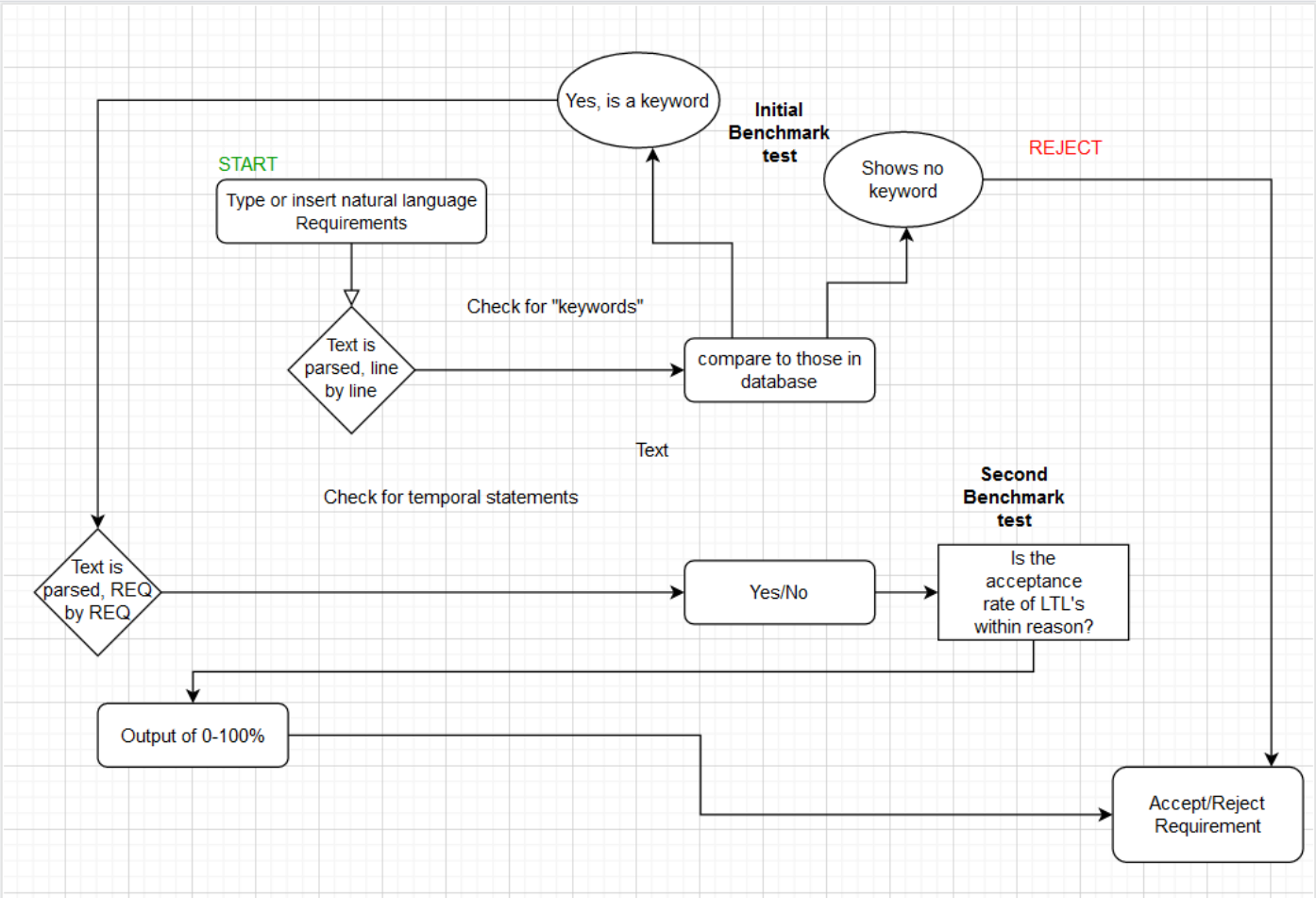

Fig1

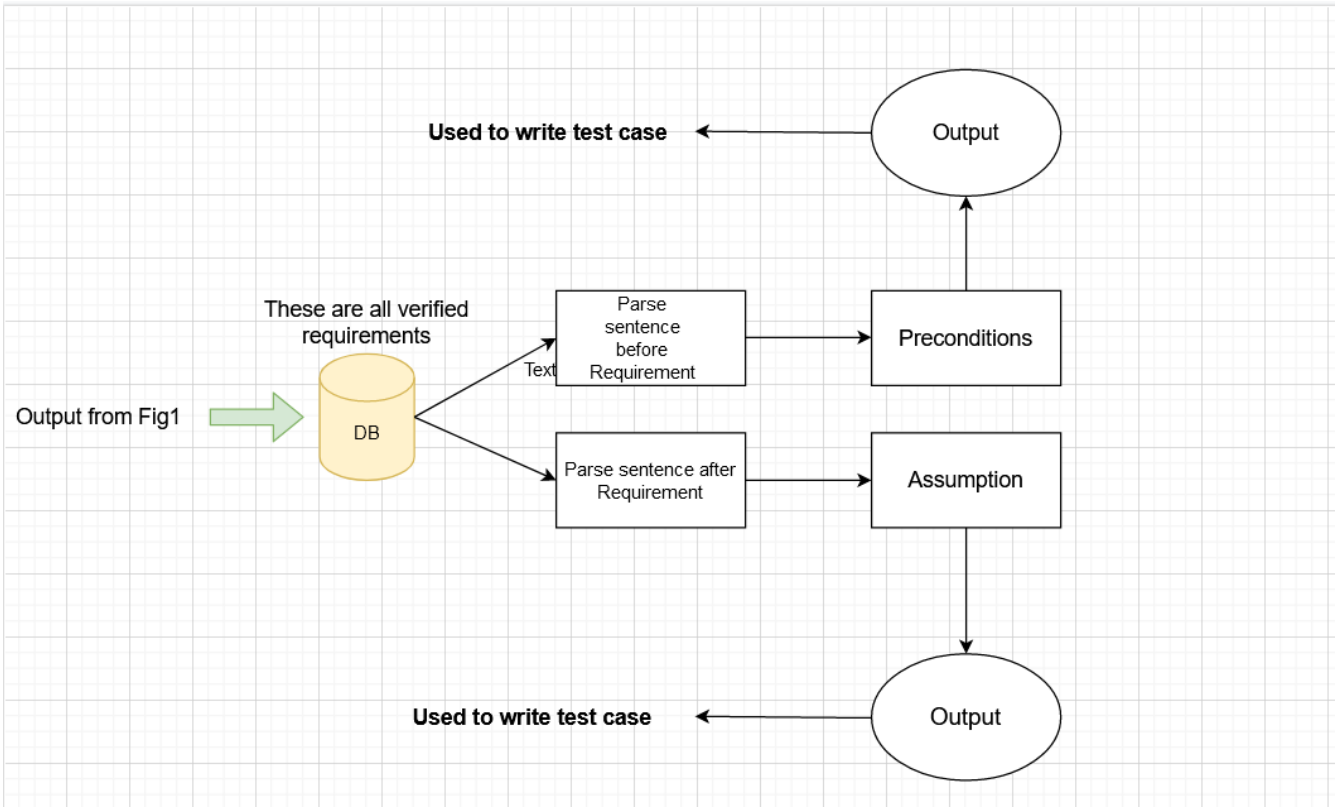
Used to write test case
Output
These are all verified requirements
Parse sentence before Requirement
Preconditions
Output from Fig1
DB
Text
Parse sentence after Requirement
Assumption
Used to write test case
Output

Fig2