Software Design Specification Document

Prediction of Fake Reviews on Amazon

Adnan Shaikh - 60

Divit Shah - 61

Rachit Shaha - 62

Samyak Jain - 63

Version: (1) **Date: (03/09/2022)**

Table of Contents

1 Introduction **1**

1.1 System Overview 1
1.2 Definitions, Acronyms, and Abbreviations 1
1.3 References 1

2 Design Considerations **3**

2.1 Assumptions 3
2.2 Constraints 3
2.3 System Environment 3
2.4 Design Methodology 4

3 Architectural (High-level) Design **5**

3.1 Overview 5
3.2 Rationale 6
3.3 Conceptual (or Logical) View 7

4 Low Level Design **8**

4.1 Chrome Extension 8
4.2 Database and Flask as Backend 9
4.3 Web Scraping 9
4.4 Machine Learning Model

5 User Interface Design **10**

5.1 Application Control 10
5.2 Chrome Extension 10

1 Introduction

1.1 System Overview

This project aims to help users by making them aware of fake reviews on Amazon for the product they want to buy. This enhances their shopping experience thus helping them in making better decisions.

There are four major components in our system architecture. They are,

- *Chrome extension i.e. frontend*
- *Web scraping*
- *Database & Flask as backend*
- *Machine Learning Model*

The chrome extension will act as the frontend with which the user will interact. It will help retrieve required data from the user**(Amazon URL)**. The product's data will be scraped using web scraping techniques. Useful information will be stored in the database for future use. The machine learning model will give back the likelihood of a product to have fake reviews and some other insights and predictions which will help the user to make informed decisions.

1.2 Definitions, Acronyms, and Abbreviations

1.2.1 Acronyms

Mentioned below are the abbreviations used in this software design specification.

- SDS - Software Design Specification
- ML - Machine Learning
- URL - Uniform Resource Locator
- UI - User Interface
- XML - Extensible Markup Language
- AJAX - Asynchronous Javascript and XML
- JSON - Javascript Object Notation
- ASIN - Amazon Standard Identification Number
- XPath - XML Path Language
- CSS - Cascading Style Sheets

1.3 References

1. Zehui Wang, Yuzhu Zhang, Tianpei Qian. Fake Review Detection on Yelp. http://cs229.stanford.edu/proj2017/final-reports/5229663.pdf

2. Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon-gyo Jung, Bernard J. Jansen. Creating and detecting fake reviews of online products. 2021, The Authors. Published by Elsevier Ltd. Creating and detecting fake reviews of online products - ScienceDirect

3. Jianmo Ni, Jiacheng Li, Julian McAuley. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. University of California, San Diego. https://cseweb.ucsd.edu//~jmcauley/pdfs/emnlp19a.pdf

4. Scrapy Documentation. https://docs.scrapy.org/en/latest/

5. Chrome developers documentation for chrome extensions. https://developer.chrome.com/docs/extensions/

6. Flask Documentation. https://flask.palletsprojects.com/en/2.0.x/

7. MongoDB documentation. https://docs.mongodb.com/

2 Design Considerations

2.1 Assumptions

The user should have a chromium based browser (e,g, Google Chrome) installed on their environment and should know how to use a chrome extension.

Automatic fake reviews detection is a challenging problem in deception detection, and it has tremendous political and social implications in the real world. The extracted scraped data is assumed to be genuine which would eventually affect the model. In feature engineering, the parameters we choose are assumed to have a greater impact than those we don't.

2.2 Constraints

Extension is currently limited to amazon e-commerce website.

It should be used with amazon on the active tab.

Product that is to be analyzed should be available in amazon's stock with its price/mrp clearly stated.

If provided with the choice, the user should select the size of the product before using the extension.

Extension does not support refurbished/used products.

Python is recommended to implement Machine Learning Algorithms so as to make them more accessible to use and comprehend.

Machine Learning Algorithms are best executed on high-end machines for improved performance.

2.3 System Environment

Software:

Any Chromium based Browser.

Hardware:

Desktop/Laptop compatible with Windows/Mac Operating System.

Stable Internet connection.

2.4 Design Methodology

- *Frontend*

  From possibilities such as Website, Chrome extension, Application GUI, and Desktop application for the frontend, the Chrome extension was selected since it improved the user experience and added functionality to the project.

- *Web scraping*

  Web scraping tools and technologies come in a number of different forms. Selectorlib, Selenium, Scrapy, and BeautifulSoup are a few of them. Scrapy was chosen because of its speed and error-handling capabilities.

- *Database*

  In today's world, there are abundant open source database softwares available. NoSQL is a new and emerging technology that is perfect for this scenario.

- Backend

  Flask was chosen above Django, Node, Java, and other options because of its ability to customize code to a great degree.

- *Machine Learning Model*

  Among Supervised learning, Semi-supervised learning, and Unsupervised learning, Reinforcement learning was the best option for developing the project in the machine learning segment. In this circumstance, reinforcement learning would assist in determining the best course of action to maximize reward.

3 Architectural (High-level) Design

3.1 Overview

There are four major components in our system architecture. They are,

- *Chrome extension i.e. frontend*

  A basic chrome extension created using the documentation available at https://developer.chrome.com/docs/extensions/mv3/. The extension is created using HTML, CSS & Javascript. It will act as an user interface for this web application. Its main functionality will be to retrieve the url of the current active tab and to show the results after processing the data.

- *Web scraping*

  First it is checked if the data of the selected product is already available in the database. If not available, the required data is scraped and stored in the database for future use.

  This is done so that we do not have to send requests to amazon again and again for the same product and unnecessarily increase the traffic on amazon.

  For scraping the required data *Scrapy,* a python based web-scraping framework is used.

- *Database & Flask as backend*

  Database is required to store and manage the data, For this purpose *MongoDB* is used. MongoDB's horizontal, scale-out architecture can support huge volumes of both data and traffic.

  Backend is written in *Flask,* a python based micro-framework. It connects all the other important components together i.e. frontend, web scraper, database and the ML model thus It is the hub of this web app.

- *Machine Learning Model*

  The data is first retrieved from the database and is provided as an input to the ML model. The model then determines if the product reviews are genuine or not by analyzing the data.

  Finally, a report is prepared that includes the model's prediction and is displayed to the user.

3.2 Rationale

- *Chrome extension(Frontend)*

  This application has a very little user interaction and content to be displayed to the user. Instead of visiting the website, the extension makes it simpler for the user to use it on top of a website. The extension provides more functionalities which helps in automating the process even further by fetching the url of the current tab.

- *Web scraping*

  Scrapy is a web scraping framework built especially for web scraping and written entirely in Python. One of the biggest advantages of Scrapy is speed. Since it's asynchronous, Scrapy spiders don't have to wait to make requests one at a time, but it can make requests in parallel. This increases efficiency, which makes Scrapy memory and CPU efficient compared to the other web scraping tools.

- *Database*

  When compared to relational databases, NoSQL databases are often more scalable and provide superior performance. In addition, the flexibility and ease of use of their data models can speed development in comparison to the relational model, especially in the cloud computing environment.

  In this project, there is no need for querying or fetching data from multiple tables at the same time. Thus NoSQL databases such as MongoDB are more suitable for this project.

- Backend

  Flask is a light-weight framework popularly categorized as a micro framework. Flask comes with some standard functionalities and allows developers to add any number of libraries or plugins for an extension. One of Flask's greatest strengths is its minimalism and simplicity. No restrictions means that the developer can implement everything exactly as they want it.

  Django is a web application framework that takes care of many of the standard functionalities to build secure and maintainable websites. Django makes it simple for Python designers to jump into web applications rapidly without requiring planning into the app's framework ahead of time.

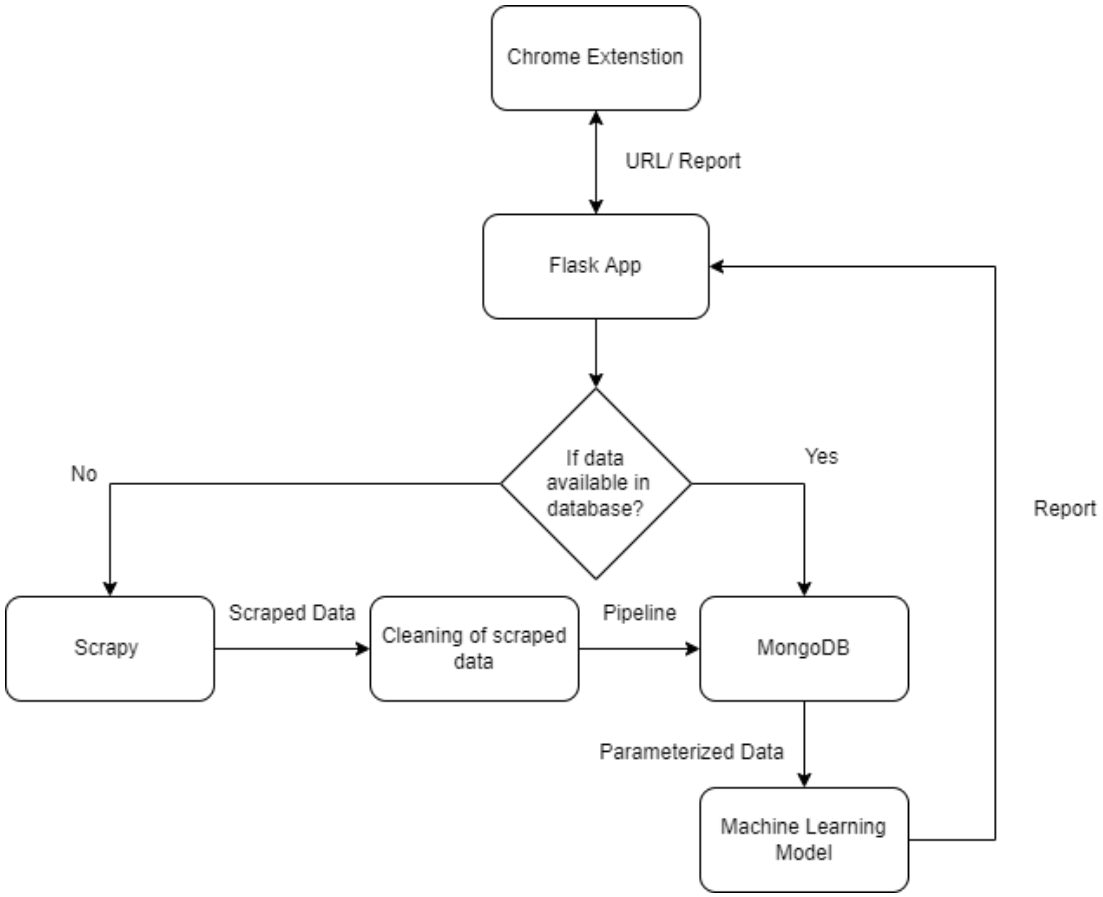
  In this case, flask proved to be a better choice as there is no complex website but a simple chrome extension. Most crucial part of this project is situated at

the backend i.e. processing the data and getting some insights with not much to showcase to the user. Hence flask is more suitable.

- *Machine Learning Model*

  Reinforcement learning is a subset of Machine Learning concerned with choosing appropriate action in order to maximize reward in a given scenario. It is used by a variety of software and machines to determine the best feasible action or path in a given situation. In this project, Reinforcement Learning is best suitable in order to gain optimal output.

3.3 Conceptual (or Logical) View



4 Low Level Design

4.1 Chrome Extension

4.1.1 User Interface (HTML, CSS)

This is the interface which the user will see and interact with. It will be used to start the process and then show the result at the end.

4.1.2 Manifest file (JSON)

A *manifest.json* file is required for any Chrome extension. The *manifest.json* file contains information that defines the extension. The format of the information in the file is JSON.

4.1.3 Background and Content Scripts(JavaScript)

The background page is an HTML page that runs in the extension process. It exists for the lifetime of your extension, and only one instance of it at a time is active. For the scope of this project, an async function has been defined in the background script to fetch the url of the current tab and return it.

Content scripts are files that run in the context of web pages. By using the standard Document Object Model (DOM), they are able to read details of the web pages the browser visits, make changes to them and pass information to their parent extension. For the scope of this project, the content script contains an event listener for a button on the UI which calls the function from the background script and then sends the returned URL to the flask app using AJAX call and shows the status on the UI.

4.2 Database and Flask as backend(Python)

4.2.1 Config file(JSON)

A JSON file stored in the root folder containing all the file paths to be accessed from a single location.

4.2.2 Database(MongoDB)

MongoDB a NoSQL database is used as there is no need for querying or fetching data from multiple tables at the same time. This database stores data scraped from Amazon every time the extension is used on a new product. This means that over time as the database grows the number of requests to Amazon will decrease. Thus increasing the overall speed of response given to the user.

4.2.3 Flask App(Python)

This is the backend of this project. It will act as the hub and connect all the other components of the project.

- It receives the AJAX call from the chrome extension containing the URL.
- Extracts the ASIN number from the URL and then stores the URL in a file.
- The ASIN is used as a primary key in the database and hence can be used to check if the product data is already available. If the data is available, it is passed to the ML model. Otherwise the data is scraped from Amazon using Scrapy.
- The prediction from the ML model is then shown to the user as output.

4.3 Web Scraping

4.3.1 Scrapy settings file(python)

The scrapy settings file contains all the settings related with scraping process e,g, we can set the user agent, no. of concurrent requests etc. All scrapy spiders follow the settings mentioned here.

4.3.2 Scrapy spiders

This file is where the actual scraping happens.

- It takes a list of URLs as input.
- Sends a request to each of the URLs and accepts the source code of the page as response.
- Using CSS selectors and XPaths we select the required tags and save its innerHTML content(text).

4.3.3 Scrapy items

The Data from the spider is stored in a custom made model for the scraped items. A different model class for each scraper is defined in the file *items.py*. This data is in the form of key-value pairs and can be saved as a csv or json file.

4.3.4 Scrapy pipeline

Scrapy has an inbuilt pipelining feature which allows us to pipeline the data saved in an item class into a database. In this case the data is transferred to MongoDB using a flask module *pymongo.*

4.4 Machine Learning Model

4.4.1 Learning type(Reinforcement)

Reinforcement learning enables decision making. It is about learning the optimal behavior in an environment to obtain maximum reward. Reinforcement learning was chosen because of the fact that the agent learns from the previous behavior and receives observations and a reward from the environment and sends actions to the environment.

4.4.2 Feature Engineering

Our project is product-centric which means that the parameters we chose for Machine learning will be product-centric and not review-centric. We have already explored a

review-centric approach which requires Natural Language Processing, but a product-centric approach is way more optimal. We chose MRP, Date, Price, Discount, Review, Upvotes, etc. as parameters for our ML model.

5 User Interface Design

5.1 Application Control

A simple chrome extension with a single clickable button (as of now).

5.2 Chrome Extension