**SYNOPSIS**

The report on “Improvised Stack Search” discusses the result of the developed software using “PYTHON”. Basically it is an application to search the [StackOverflow](http://stackoverflow.com) site using a query.

In this software there have only client side user who is performing the user-specific tasks. The “Client” can search query and get the most relevant answer.

**1. ABSTRACT**

**2. INTRODUCTION**

Improvised Stack Search is an application designed primarily to search the [StackOverflow](http://stackoverflow.com) site using a query and obtain a specific number of relevant questions, and their most relevant answer. The application is written in Python and is designed to run on various Cloud Platforms with minimum changes.

**3. LITERATURE REVIEW**

In 2018, Dezhic [1] described the applications of knowledge graphs to operate on unstructured data. In 2014, Guzman et. al. [3] described the usage of sentiment analysis on comments to estimate the

acceptability of a commit and in 2019, Chauhan et. al. [2] used sentiment analysis on comments to rank Youtube videos.

In 2018, Moslem [4] showed how to use Natural Language ToolKit library to calculate edit distance of two sentences.

**4. PROPOSED SOLUTION**

This section describes the architectural flow of the proposed solution, and the technologies used in developing the solution.

**4.1 Technologies Used**

It uses the [StackAPI](https://stackapi.readthedocs.io/en/latest/), [NetworkX Library](https://networkx.github.io/), [Natural Language ToolKit](https://www.nltk.org/) and [IBM Watson Natural Language Understanding API](https://www.ibm.com/watson/services/natural-language-understanding/) for obtaining data and determining relevant solutions. The application is designed as a [Flask](http://flask.pocoo.org/) application, which can be run on Cloud Platforms, like Amazon Web Services and IBM Cloud Platform.

**4.1.1 StackAPI** [5]

StackAPI is a simple Python wrapper for the Stack Exchange API.

StackAPI was written to scratch an itch and in the process make my life a bit easier than messing withcurlorurllibwhile doing so.

Retrieving data from the API is simple:

from stackapi importStackAPI

SITE = StackAPI('stackoverflow')

comments = SITE.fetch('comments')

This will return the 500 most recent comments on Stack Overflow, using the default filter the API provides. The valuepassed tofetchis an end point defined in the Stack Exchange API.

**4.1.2 Natural Language ToolKit (NLTK)** [6]

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-useinterfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries forclassification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLPlibraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics,plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers,and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, opensource, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and“an amazing library to play with natural language.”

**4.1.3 NetworkX Library** [7]

The structure of NetworkX can be seen by the organization of its source code. The package provides classes for graphobjects, generators to create standard graphs, IO routines for reading in existing datasets, algorithms to analyze theresulting networks and some basic drawing tools.

Most of the NetworkX API is provided by functions which take a graph object as an argument. Methods of the graphobject are limited to basic manipulation and reporting. This provides modularity of code and documentation. It alsomakes it easier for newcomers to learn about the package in stages. The source code for each module is meant to beeasy to read and reading this Python code is actually a good way to learn more about network algorithms, but we haveput a lot of effort into making the documentation sufficient and friendly. If you have suggestions or questions pleasecontact us by joining the NetworkX Google group.

Classes are named using CamelCase (capital letters at the start of each word). functions, methods and variable namesare lower\_case\_underscore (lowercase with an underscore representing a space between words).

The first choice to be made when using NetworkX is what type of graph object to use. A graph (network) is a collectionof nodes together with a collection of edges that are pairs of nodes. Attributes are often associated with nodes and/oredges. NetworkX graph objects come in different flavors depending on two main properties of the network:

Directed: Are the edgesdirected? Does the order of the edge pairs(u, v)matter? A directed graph isspecified by the “Di” prefix in the class name, e.g.DiGraph(). We make this distinction because manyclassical graph properties are defined differently for directed graphs.

Multi-edges: Are multiple edges allowed between each pair of nodes? As you might imagine, multiple edgesrequires a different data structure, though clever users could design edge data attributes to support this function-ality. We provide a standard data structure and interface for this type of graph using the prefix “Multi”, e.g.,MultiGraph().

The basic graph classes are named: Graph,DiGraph,MultiGraph, and MultiDiGraph

Empty graph-like objects are created with:

>>>G = nx.Graph()

>>>G = nx.DiGraph()

>>>G = nx.MultiGraph()

>>>G = nx.MultiDiGraph()

The graph internal data structures are based on an adjacency list representation and implemented using Pythondictio-narydatastructures. The graph adjacency structure is implemented as a Python dictionary of dictionaries; the outerdictionary is keyed by nodes to values that are themselves dictionaries keyed by neighboring node to the edge at-tributes associated with that edge. This “dict-of-dicts” structure allows fast addition, deletion, and lookup of nodesand neighbors in large graphs. The underlying datastructure is accessed directly by methods (the programming in-terface “API”) in the class definitions. All functions, on the other hand, manipulate graph-like objects solely viathose API methods and not by acting directly on the datastructure. This design allows for possible replacement of the‘dicts-of-dicts’-based datastructure with an alternative datastructure that implements the same methods

**4.1.4 IBM Watson Natural Language Understanding** [8]

Analyze various features of text content at scale. Provide text, raw HTML, or a public URL and IBM Watson Natural Language Understanding will give you results for the features you request. The service cleans HTML content before analysis by default, so the results can ignore most advertisements and other unwanted content.

You can create [custom models](https://cloud.ibm.com/docs/services/natural-language-understanding?topic=natural-language-understanding-customizing) with Watson Knowledge Studio to detect custom entities, relations, and categories in Natural Language Understanding.

Natural Language Understanding includes a set of text analytics features that you can use to extract meaning from unstructured data.

* Emotion

Detects anger, disgust, fear, joy, or sadness that is conveyed in the content or by the context around target phrases specified in the targets parameter. You can analyze emotion for detected entities with entities.emotion and for keywords with keywords.emotion.

* Entities

Identifies people, cities, organizations, and other entities in the content.

* Relations

Recognizes when two entities are related and identifies the [type of relation](https://cloud.ibm.com/docs/services/natural-language-understanding?topic=natural-language-understanding-relation-type-systems). For example, an awardedTo relation might connect the entities "Nobel Prize" and "Albert Einstein".

* Semantic Roles

Parses sentences into subject, action, and object form.

* Sentiment

Analyzes the general sentiment of your content or the sentiment toward specific target phrases. You can analyze sentiment for detected entities with entities.sentiment and for keywords with keywords.sentiment .

**4.1.5 Flask Framework** [9]

“Micro” does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The “micro” in microframework means Flask aims to keep the core simple but extensible. Flask won’t make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don’t.

By default, Flask does not include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be “micro”, but it’s ready for production use on a variety of needs.

## Configuration and Conventions

Flask has many configuration values, with sensible defaults, and a few conventions when getting started. By convention, templates and static files are stored in subdirectories within the application’s Python source tree, with the names templates and static respectively. While this can be changed, you usually don’t have to, especially when getting started.

## Growing with Flask

Once you have Flask up and running, you’ll find a variety of extensions available in the community to integrate your project for production. The Flask core team reviews extensions and ensures approved extensions do not break with future releases.

As your codebase grows, you are free to make the design decisions appropriate for your project. Flask will continue to provide a very simple glue layer to the best that Python has to offer. You can implement advanced patterns in SQLAlchemy or another database tool, introduce non-relational data persistence as appropriate, and take advantage of framework-agnostic tools built for WSGI, the Python web interface.

Flask includes many hooks to customize its behavior. Should you need more customization, the Flask class is built for subclassing. If you are interested in that, check out the [Becoming Big](http://flask.pocoo.org/docs/1.0/becomingbig/#becomingbig) chapter. If you are curious about the Flask design principles, head over to the section about [Design Decisions in Flask](http://flask.pocoo.org/docs/1.0/design/#design).

Flask depends on the [Jinja](https://www.palletsprojects.com/p/jinja/) template engine and the [Werkzeug](https://www.palletsprojects.com/p/werkzeug/) WSGI toolkit.

Jinja2 is a modern and designer-friendly templating language for Python, modelled after Django’s templates. It is fast, widely used and secure with the optional sandboxed template execution environment:

<title>{% block title %}{% endblock %}</title>

<ul>

{% for user in users %}

<li><a href="{{ user.url }}">{{ user.username }}</a></li>

{% endfor %}

</ul>

## Features of Jinja2:

* sandboxed execution
* powerful automatic HTML escaping system for XSS prevention
* template inheritance
* compiles down to the optimal python code just in time
* optional ahead-of-time template compilation
* easy to debug. Line numbers of exceptions directly point to the correct line in the template.
* configurable syntax

## The [Werkzeug](https://www.palletsprojects.com/p/werkzeug/) WSGI toolkit:

Werkzeug is a comprehensive [WSGI](https://wsgi.readthedocs.io/en/latest/) web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.

Werkzeug is Unicode aware and doesn’t enforce any dependencies. It is up to the developer to choose a template engine, database adapter, and even how to handle requests.

## Running code as Flask Server

python search.py

#### Output:

\* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: \*\*\*-\*\*\*-\*\*\*

**4.2 Architecture of the proposed solution**

This section describes the architecture of the proposed solution. The architecture primarily comprises of 5 sections – (a) Obtaining Keywords, (b) Obtaining questions using keywords of the query, (c) Graph and Edit Distance-based question ranking, (d) Sentiment analysis based answer picking and (e) Deployment as a Flask Application. Fig. - 4.1 describes the Data Flow Diagram of the proposed solution.

**4.2.1 Obtaining keywords**

The statement for which keywords are to be fetched, is passed on the IBM Watson Natural Language Understanding API. The result of the API call is processed and a set of keywords is obtained. Keywords(tags) of questions can also be obtained from the the result of StackAPI search.

**4.2.2 Obtaining questions using keywords of the query**

StackAPI search is invoked to obtain a set of questions from Stackoverflow, using a string obtained by concatenating the keywords of the query, delimited by space. If the search returns no question,

or returns question less than the provided limit, then the search is invoked again for each subset of 3 keywords from the set of keywords.

**4.2.3 Graph and Edit Distance-based question ranking**

A simple graph is created in which nodes are the keywords obtained from title of the questions and edges are formed among all keywords of each question. The shortest paths for each keyword pairs are obtained using NetworkX Library. The edit distances between the query and each questions are calculated using NLTK Library. The score (Sq) of each question is obtained as follows:

d <= Corresponding edit distance of the question

P <= set of shortest paths

Pn <= n(P)

T <= set of tags of the questions

i=0

L <= ∑ n(Pi ∩ T)

Pn

Sq <= (1/d + (L/(Pn × n(T))))/2

The questions are then arranged in descending order of the score (Sq) obtained, and subsequently ranked.

**4.2.4 Sentiment analysis-based answer picking**

The answers for each question and their corresponding comments are obtained using StackAPI, by passing a set of question identifiers. The sentiment analysis score of each comment is obtained by using IBM Watson Natural Language Understanding API. The score (Sa) of each answer is calculated as follows:

Sc <= set of sentiment analysis scores of the comments of the answer

Cn <= n(Sc)

i=0

K <= ∑ Sci

Cn

Sa <= K/Cn

The top answer based on the score (Sa) is selected.

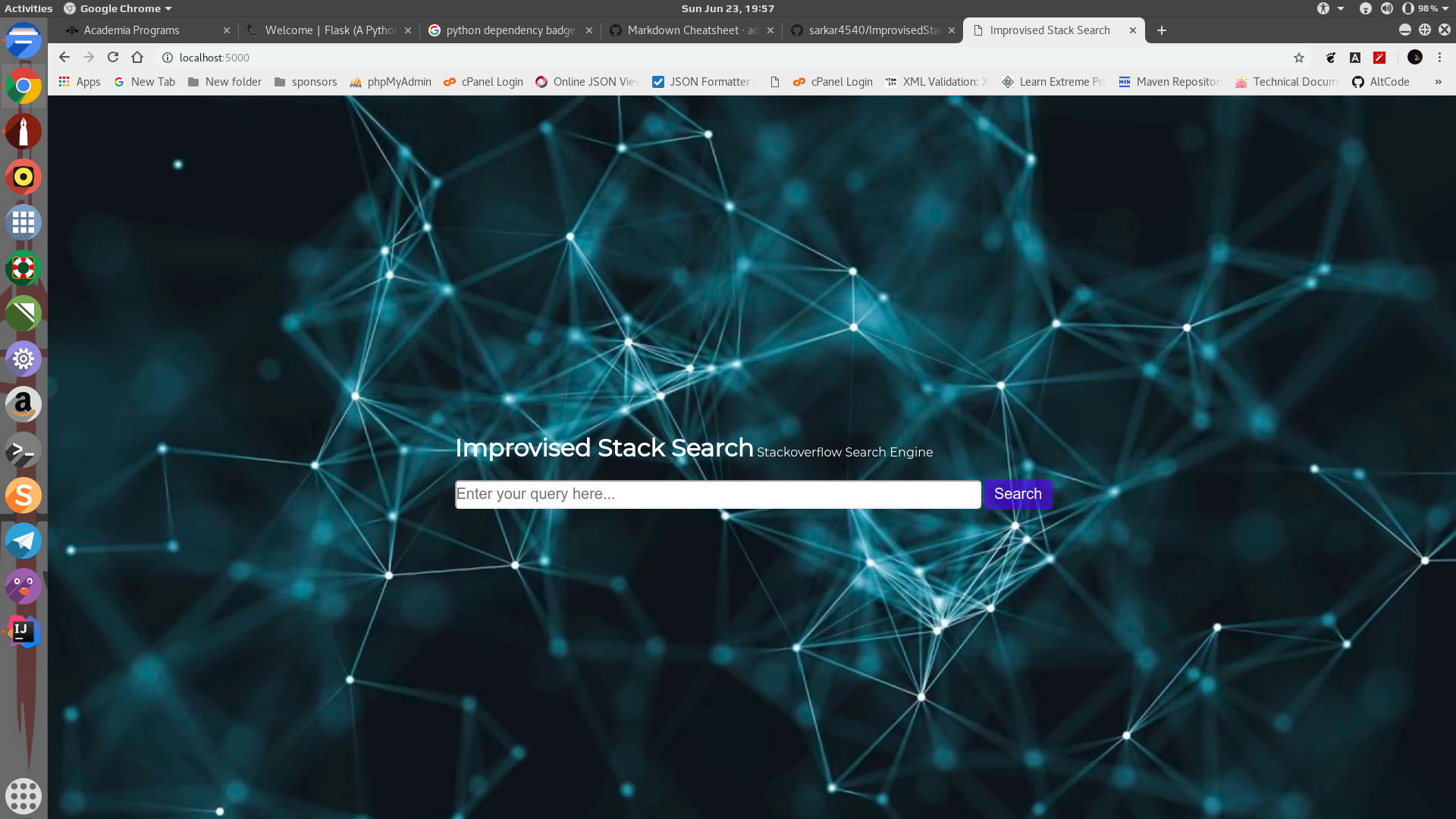
**4.2.5 Deployment as a Flask Application**

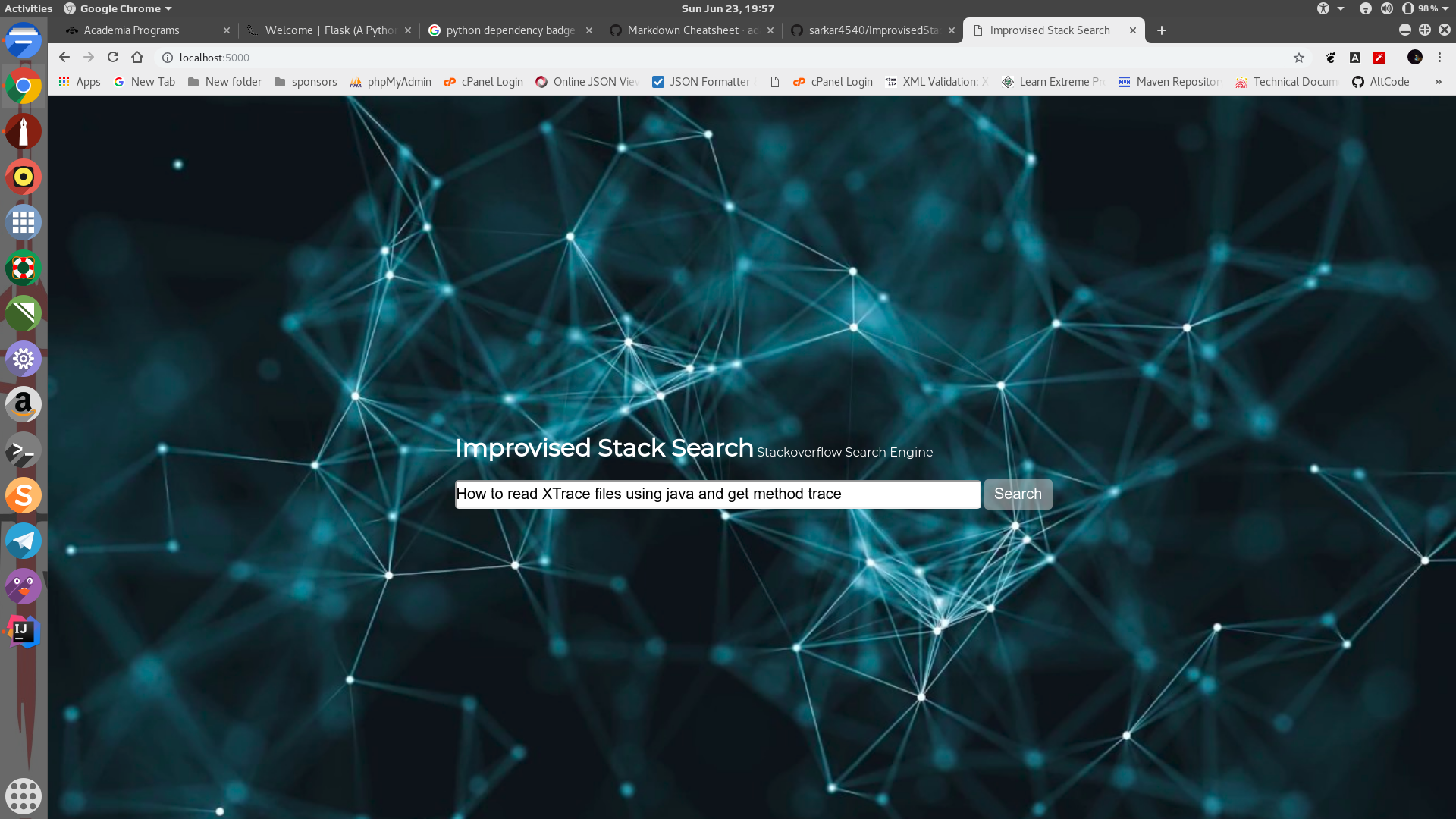
The application was designed as a Flask Application to allow usage of the solution as a Web Application and enable micro-services based architecture for the same. A HTML User Interface was developed and served as static files of the Flask Application. The user interface accepts query and sends a HTTP request to the Flask Server. The server processes the query and responds with a

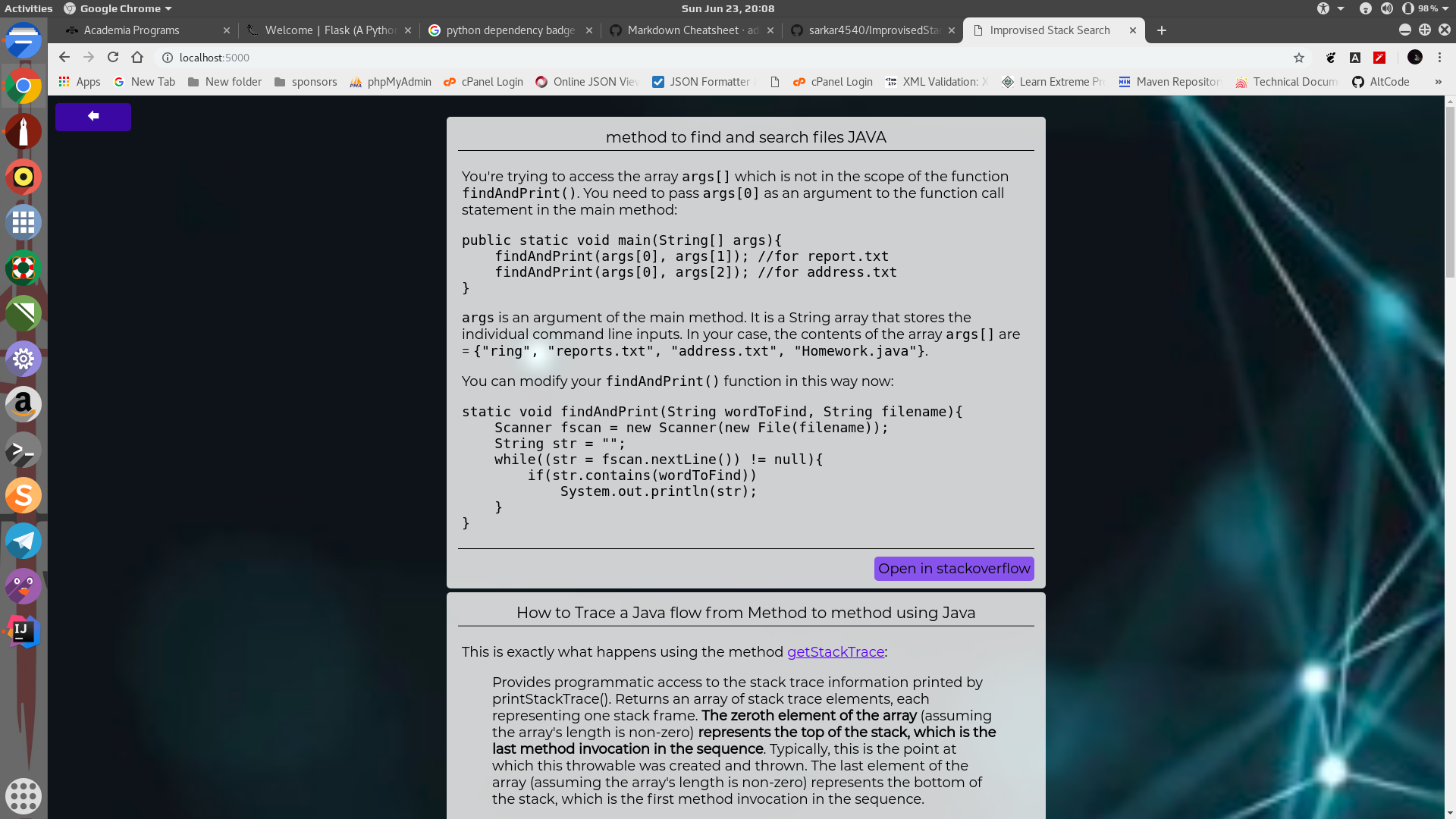
JSON containing the questions and corresponding answers, which are displayed in the User

Interface.

**5. RESULTS**







**6. IMPROVEMENTS**

This project is all about to search the [StackOverflow](http://stackoverflow.com) site using a query and obtain a specific number of relevant questions, and their most relevant answer.

And all the requirements to develop this type of application is given below,

* Stackapi
* Networkx
* Flask (v-0.12.2)
* Future
* Ibm-watson
* Nltk

*Note: write the steps of development of projects*

**7. CONCLUSION**

**8. REFERENCES**

[1] Dezhic, I. (May 21, 2018) *Understanding Knowledge Graphs.* Retrieved from <https://medium.com/@Dezhic/understanding-knowledge-graphs-5cb05593eb84>

[2] Chauhan, G.S., Meena Y.K. (2019) *YouTube Video Ranking by Aspect-Based Sentiment Analysis on User Feedback. In: Wang J., Reddy G., Prasad V., Reddy V. (eds) Soft Computing and Signal Processing. Advances in Intelligent Systems and Computing, vol 900. Springer, Singapore.* doi: 10.1007/978-981-13-3600-3\_6

[3] Guzman, E., Azócar, D., & Li, Y. (2014).*Sentiment analysis of commit comments in GitHub: an empirical study. Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014.*doi:10.1145/2597073.2597118

[4] Moslem, Y. (2018). *Edit Distance and Jaccard Distance Calculation with NLTK.* Retrieved from: <https://python.gotrained.com/nltk-edit-distance-jaccard-distance/>

[5] Wegner, A. (2019). *StackAPI Documentation*. Retrieved from:

<https://buildmedia.readthedocs.org/media/pdf/stackapi/stable/stackapi.pdf>

[6] Bird, S. (2017). *NLTK Documentation.* Retrieved from:

<https://buildmedia.readthedocs.org/media/pdf/nltk/latest/nltk.pdf>

[7] Hagberg A., Schult D., Swart P. (2017). *NetworkX Reference.* Retrieved from:

<https://buildmedia.readthedocs.org/media/pdf/networkx/latest/networkx.pdf>

[8] *Natural Language Understanding - IBM Cloud API Docs*. Retrieved from:

<https://cloud.ibm.com/apidocs/natural-language-understanding>

[9] (2010). *Welcome to Flask - Flask 1.0.2 documentation.* Retrieved from:

<http://flask.pocoo.org/docs/1.0/>