“PREDICTIVE SEARCH”

A

Project Report

Submitted in partial fulfillment of the

Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

With specialization

Banking, Finance, Services and Insurance

by

Name	**Roll No.**
Siddhant Bishnoi	**R133214045**
Sujay Venaik	**R133214047**
Prerna Arora	**R133214031**

Under the guidance of

Ms. Sachi Chaudhary Shukla
Assistant Professor
School of Computer Science and Engineering

School of Computer Science & Engineering

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

December 2017

CANDIDATE'S DECLARATION

We hereby certify that the project work entitled **"Predictive Search "** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in BANKING, FINANCE, SERVICES AND INSURANCE and submitted to the School of Computer Science & Engineering, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **August**, **2017** to **December**, **2017** under the supervision of **Ms. Sachi Chaudhary Shukla**.

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Siddhant Bishnoi
R133214045
Sujay Venaik
R133214047
Prerna Arora
R133214031

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 11 December 2017

Sachi Chaudhary Shukla
Assistant Professor
SoCSE, UPES

Dr. T.P. Singh
Program Head
Center for Information Technology
University of Petroleum & Energy Studies, Dehradun – 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Ms. Sachi Chaudhary Shukla, Assistant Professor**, for all advice, encouragement and constant support that she has given us through out our project work. This work would not have been possible without her support and valuable suggestions.

We sincerely thank to our respected **Dr. T.P. Singh**, **Program Head** for his great support in doing our project.

We are also grateful to **Dr. Manish Prateek, Director-SOCSE** and **Dr. Kamal Bansal, Dean SOCSE, UPES** for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Siddhant Bishnoi**	**Sujay Venaik**	**Prerna Arora**
Roll No.	**R133214045**	**R1333214047**	**R133214031**

ABSTRACT

Ever since Google introduced auto-complete in 2004, predictive search has become a welcome part of our internet interactions, helping us search faster, find results quicker, and discover answers to questions we didn't even know we had.

As predictive search becomes more powerful, tools like Google now have become capable of delivering relevant, personalized information to users, all but eliminating the need for search as we know it.

Here predictive search feature uses a predictive search algorithm based on popular searches to predict a user's search query as it is typed, providing a dropdown list of suggestions that changes as the user adds more characters to the search input.

This may seem like a minor feat, but people type considerably slower than they read, and predictive search saves users quite a bit of time by not making them always have to type their full query.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 BACKGROUND

Predictive search has become a cornerstone of a search engine, to make search easier and serving as further justification for keeping track of so many users' queries.

There are many ways to build a predictive text system, but most predictive text systems have default settings which can be configured/changed/learned from the user. In learning based systems, the system learns that usually a set of words in a particular sequence result in a particular complete word.

Machine learning algorithms that make predictions on given set of samples. Supervised machine learning algorithm searches for patterns within the value labels assigned to data points.

There can be dictionary-based text predictors, which rely on a dictionary (of a particular language) and suggest words/corrections based on this dictionary. On the other hand, there can be non-dictionary predictors which predict based on statistics, the probability of a certain letter (or a set of letters) to be a prefix to a word.

1.2 REQUIREMENT ANALYSIS

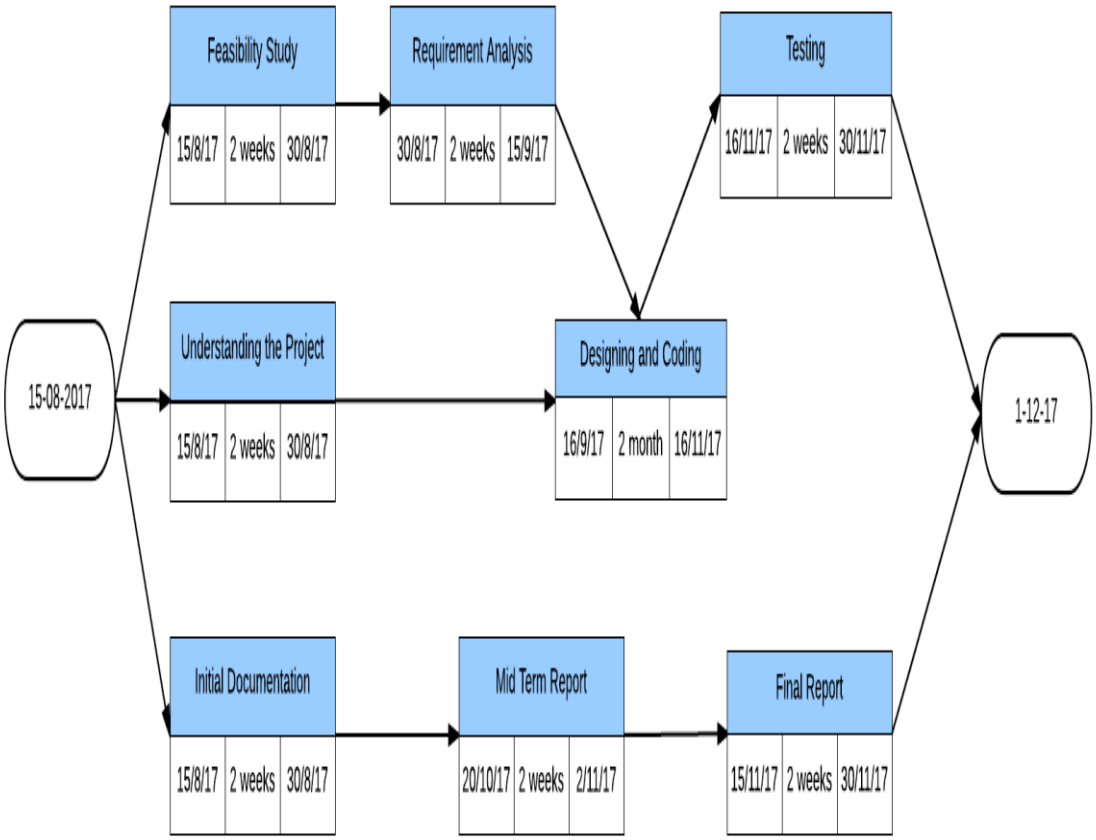
SOFTWARE REQUIREMENTS:

Operating System	:	Linux
Programming Language	:	Python
Text Editor	:	Sublime text 2.0

HARDWARE REQUIREMENTS

Processor	: Core 2 duo or higher
Hard Disk	: 20 GB
RAM	: 4 GB
Network Availability	: Yes

1.3 PERT CHART



2. LITERATURE REVIEW

2.1 EXISTING SYSTEM

Predictive search feature uses a predictive search algorithm based on popular searches to predict a user's search query as it is typed, providing a dropdown list of suggestions that changes as the user adds more characters to the search input.

2.1.1 AUTOCOMPLETE TECHNIQUE

Autocomplete, or word completion, is a feature in which an application predicts the rest of a word a user is typing.

Autocomplete speeds up human-computer interactions when it correctly predicts the word a user intends to enter after only a few characters have been typed into a text input field. It works best in domains with a limited number of possible words, when some words are much more common, or writing structured and predictable text.

2.1.2 BLOCK MATCHING TECHNIQUE

All block matching technique are proposed to achieve one or more of the three objectives. They are

- to reduce the computational complexity,
- get true motion (high quality) and
- to reduce the bit rate (higher compression ratio).

The search process started from the core block (i.e., the block at the center of frame) to be searched using the previous frame. The main idea of the proposed predictive search system is to use the determined motion vectors of the already searched blocks to predict the best search area of the unsearched neighbor block. The prediction depends on the number and positions of the previously searched blocks taken into consideration to do prediction.

2.1.3 DICTIONARY BASED SYSTEM

In dictionary-based systems, as the user presses the number buttons, an algorithm searches the dictionary for a list of possible words that match the keypress combination, and offers up the most probable choice. The user can then confirm the selection and move on, or use a key to cycle through the possible combinations.

2.1.4 NON DICTIONARY BASED SYSTEM

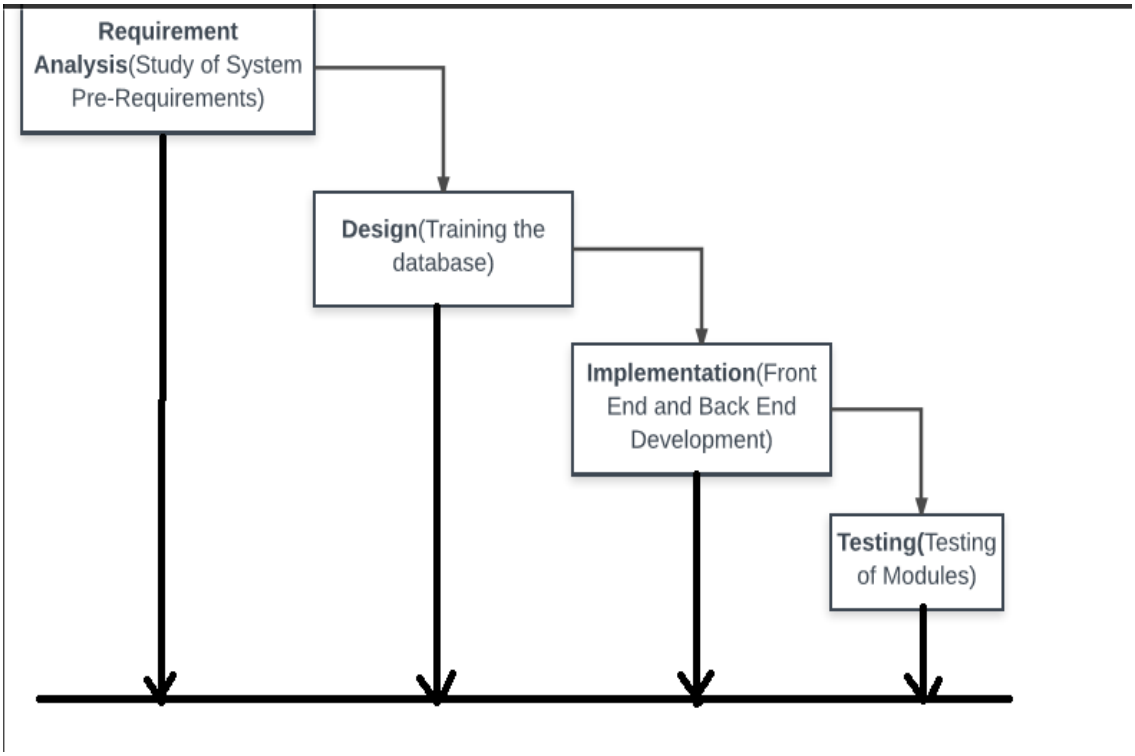
A non-dictionary system constructs words and other sequences of letters from the statistics of word parts. To attempt predictions of the intended result of keystrokes not yet entered, disambiguation may be combined with a word completion facility.

3. MAIN OBJECTIVE

- To provide accurate results faster.
- To include a factor of probability so that the search results change on every hit by the user

4. METHODOLOGY

4.1 METHOD MODELLING



4.2 MODULES:

4.2.1 FRONT END

The interface is made using HTML, CSS, JavaScript. It populates the result as per the user types the query by breaking it and matching it with the dataset available using re-indexing and gives us a list of results predicted.

4.2.2 BACK END

The Backend makes use of the default database of Django i.e. SQLITE. The data set is present in the form of a json file which has structures for each contact embed in it. The set presently contains 100 contacts that are synced using the Models file. The models file is the one in which classes are defined for the database structures and the database tables are made based on it.

Once we have the data, before starting the Django server we need to migrate the data i.e. perform any updations if are made on the models file of the app and make the required changes on the database table.

4.2.3 AUTO COMPLETE

Autocomplete, or word completion, is a feature in which an application predicts the rest of a word a user is typing. In graphical user interfaces, users can typically press the tab key to accept a suggestion or the down arrow key to accept one of several.

Autocomplete speeds up human-computer interactions when it correctly predicts the word a user intends to enter after only a few characters have been typed into a text input field. It works best in domains with a limited number of possible words (such as in command line interpreters), when some words are much more common (such as when addressing an e-mail), or writing structured and predictable text (as in source code editors).

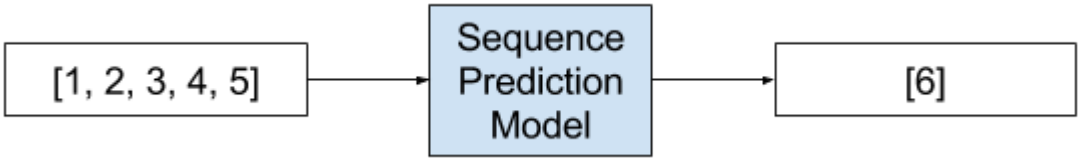
Autocomplete or word completion works so that when the writer writes the first letter or letters of a word, the program predicts one or more possible words as choices. If the word he intends to write is included in the list he can select it, for example by using the number keys. If the word that the user wants is not predicted, the writer must enter the next letter of the word. At this time, the word choice(s) is altered so that the words provided begin with the same letters as those that have been selected.

4.2.4 LSTM: Long Short Term Memory Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

Sequence prediction is a problem that involves using historical sequence information to predict the next value or values in the sequence.

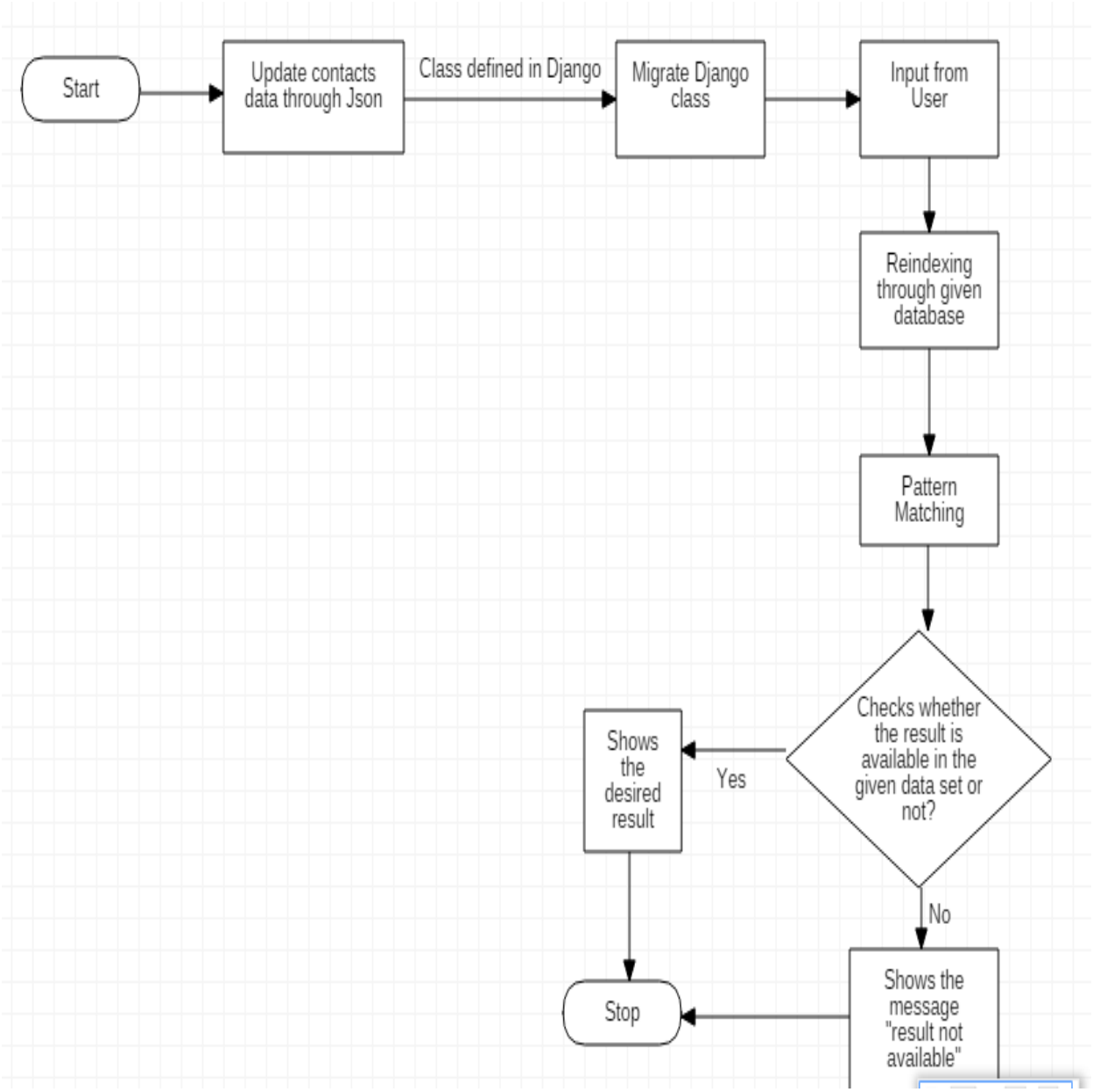
The sequence may be symbols like letters in a sentence or real values like those in a time series of prices. Sequence prediction may be easiest to understand in the context of time series forecasting as the problem is already generally understood.



Example of a Sequence Prediction Problem

A prediction model is trained with a set of training sequences. Once trained, the model is used to perform sequence predictions. A prediction consists in predicting the next items of a sequence. Till now the predictions made by our system is fetched through the dictionary or the data set we have made which is limited.

4.3**PROCESS FLOW DIAGRAM**



5. TECHNOLOGIES USED

5.1 PYTHON

- The programming language used for the project is Python on which Django framework is based.
- Python is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming (django, Zope, Google App Engine, and much more). But it also can be used for desktop applications (Blender 3D, or even for games pygame). Python can also be translated into binary code like java
- Open source general-purpose language.
- Object Oriented, Procedural, Functional

5.2 Django FRAMEWORK:

- Our applications is made on Django programming framework.
- Django emphasizes reusability and pluggability of components, rapid development, and the principle of don't repeat yourself.
- Django uses Python which is the most popular programming language.

5.3 SQLite

- For database connectivity we use SQLite.
- SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.
- It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

5.4 JavaScript

- JavaScript is a very powerful client-side scripting language.
- JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

6. LIMITATIONS

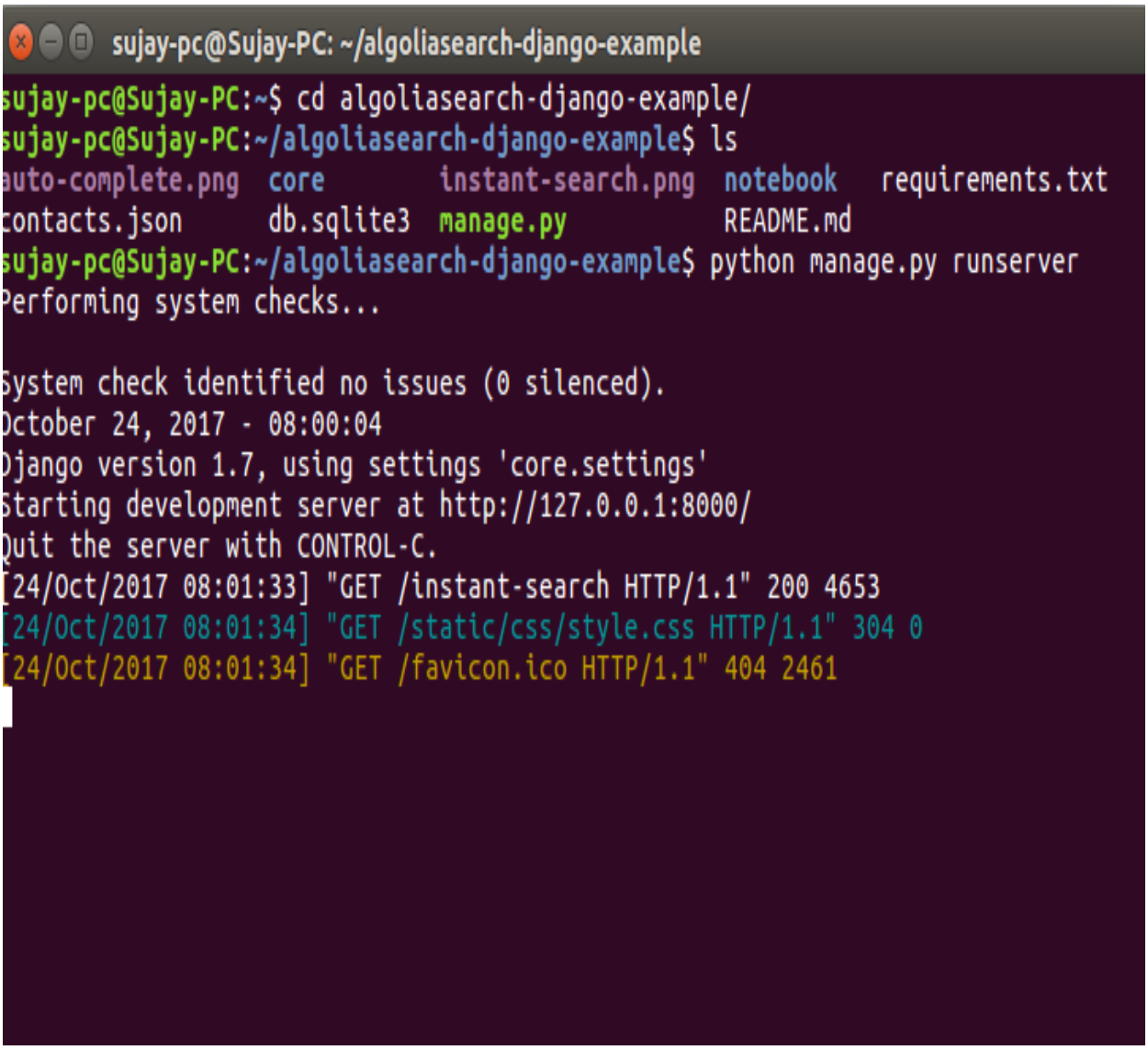
- Till now our system has limited data set. We need to increase the data set to increase the accuracy.
- Also work on re indexing can be performed so as to increase the results

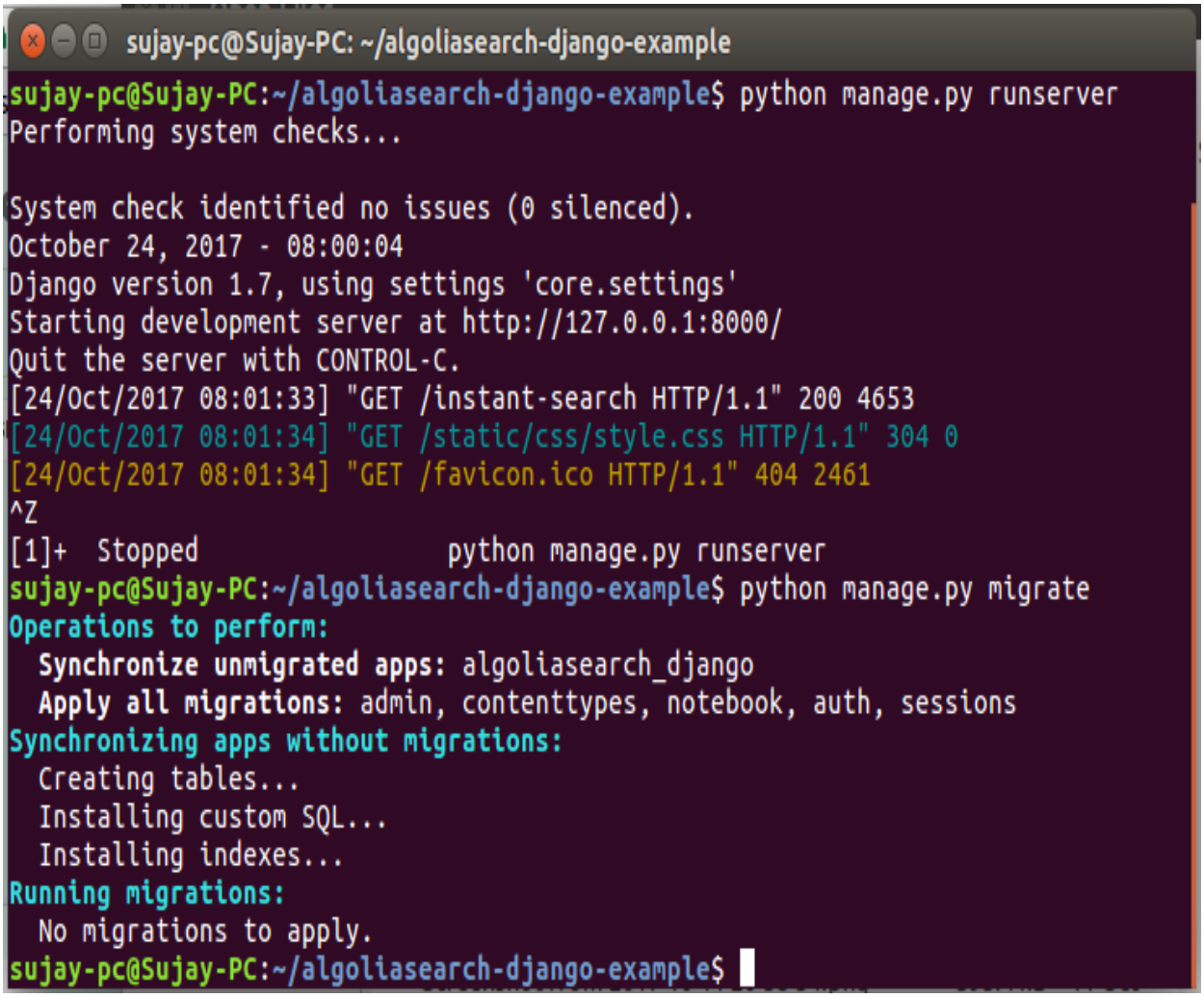
7. FUTURE ENHANCEMENTS

There is always a room for improvements in any software package, however good or efficient it may be done. But the most important thing is flexible to accept further modification. We can always increase predictions by increasing the data set provides as it will lead to more accurate results.

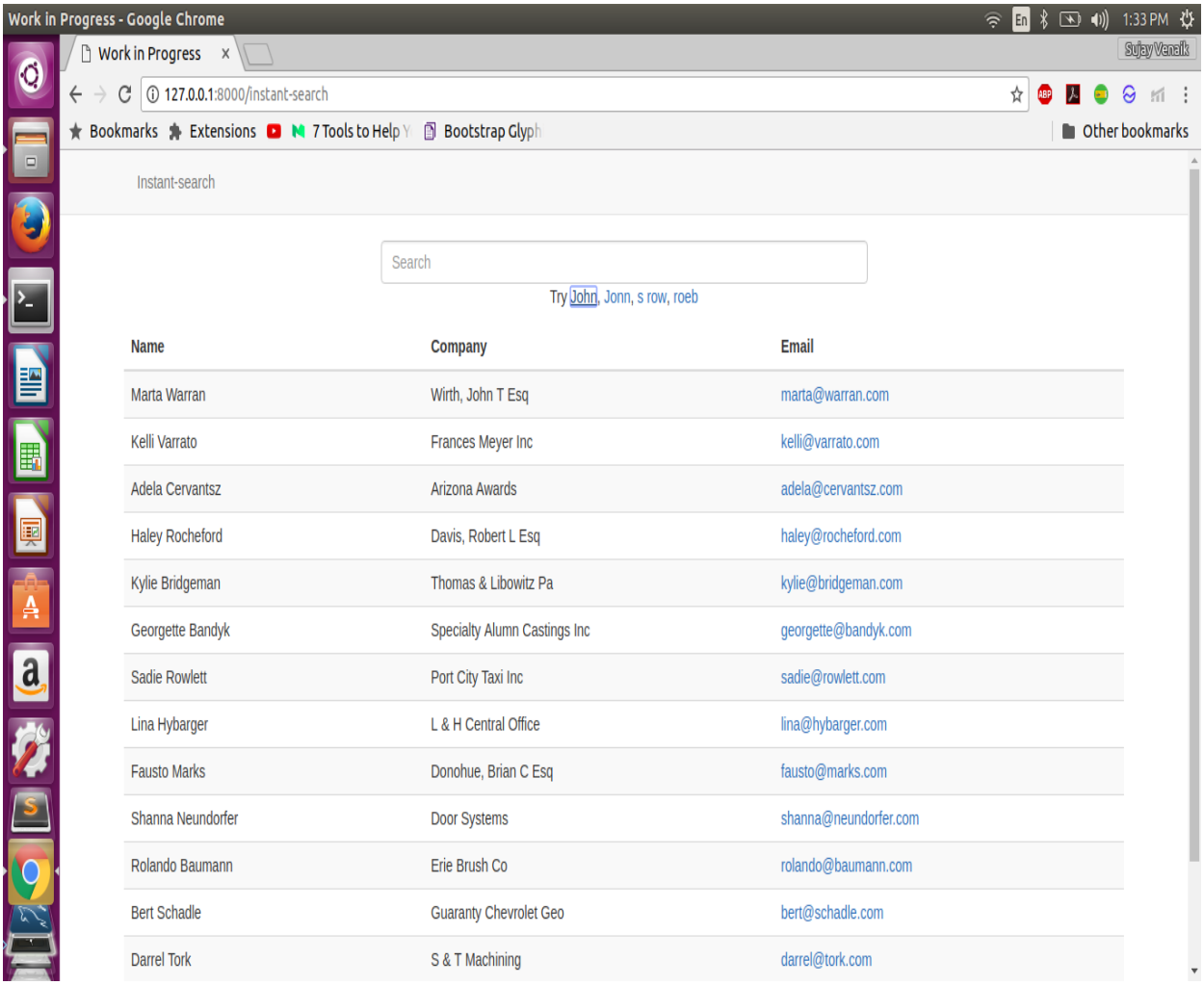
8. SNAPSHOTS

Starting the Server

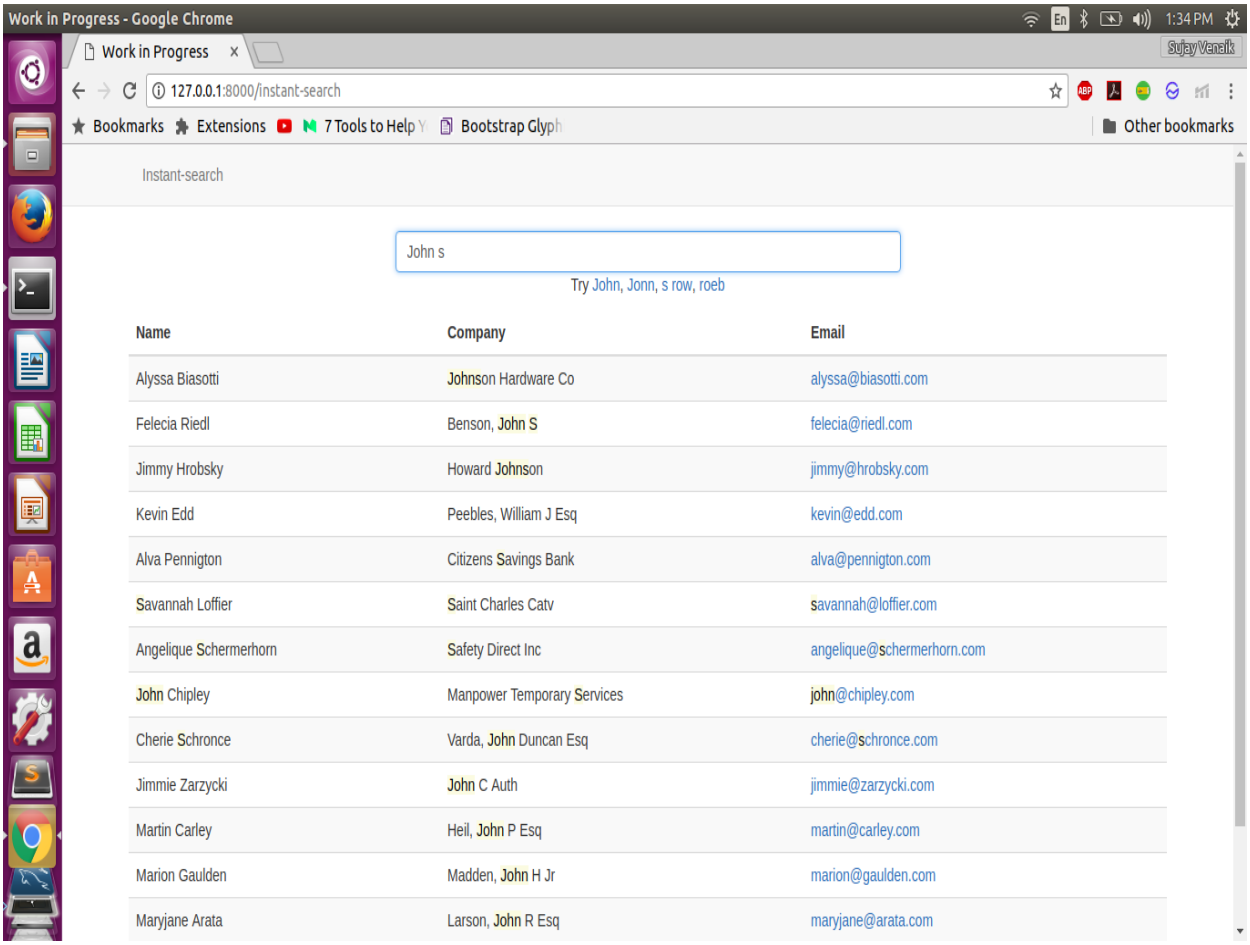



sujay-pc@Sujay-PC: ~/algoliasearch-django-example
sujay-pc@Sujay-PC:~/algoliasearch-django-example$ python manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
October 24, 2017 - 08:00:04
Django version 1.7, using settings 'core.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[24/Oct/2017 08:01:33] "GET /instant-search HTTP/1.1" 200 4653
[24/Oct/2017 08:01:34] "GET /static/css/style.css HTTP/1.1" 304 0
[24/Oct/2017 08:01:34] "GET /favicon.ico HTTP/1.1" 404 2461
^Z
[1]+ Stopped python manage.py runserver
sujay-pc@Sujay-PC:~/algoliasearch-django-example$ python manage.py migrate
Operations to perform:
Synchronize unmigrated apps: algoliasearch_django
Apply all migrations: admin, contenttypes, notebook, auth, sessions
Synchronizing apps without migrations:
Creating tables...
Installing custom SQL...
Installing indexes...
Running migrations:
No migrations to apply.
sujay-pc@Sujay-PC:~/algoliasearch-django-example$

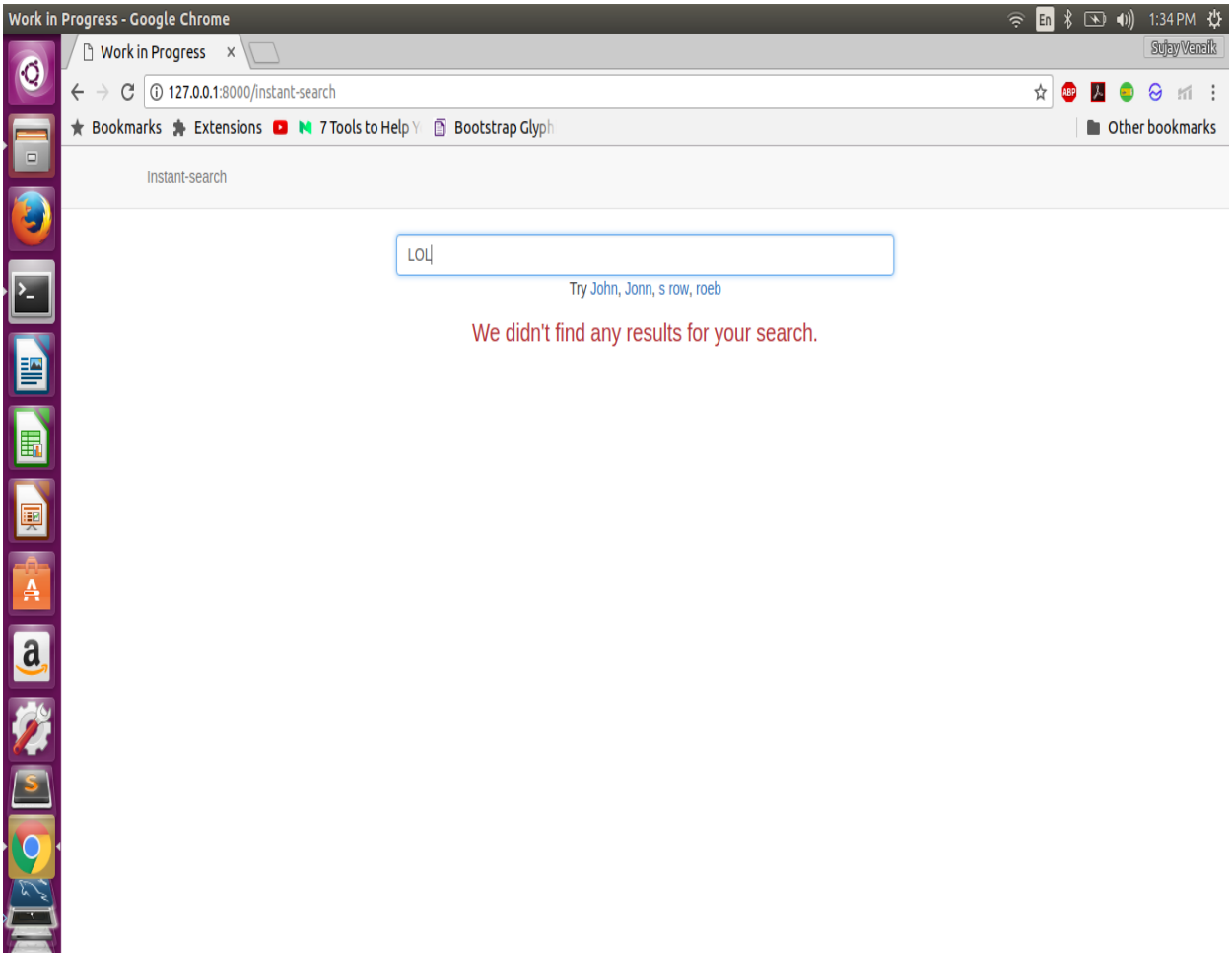
Searching Interface



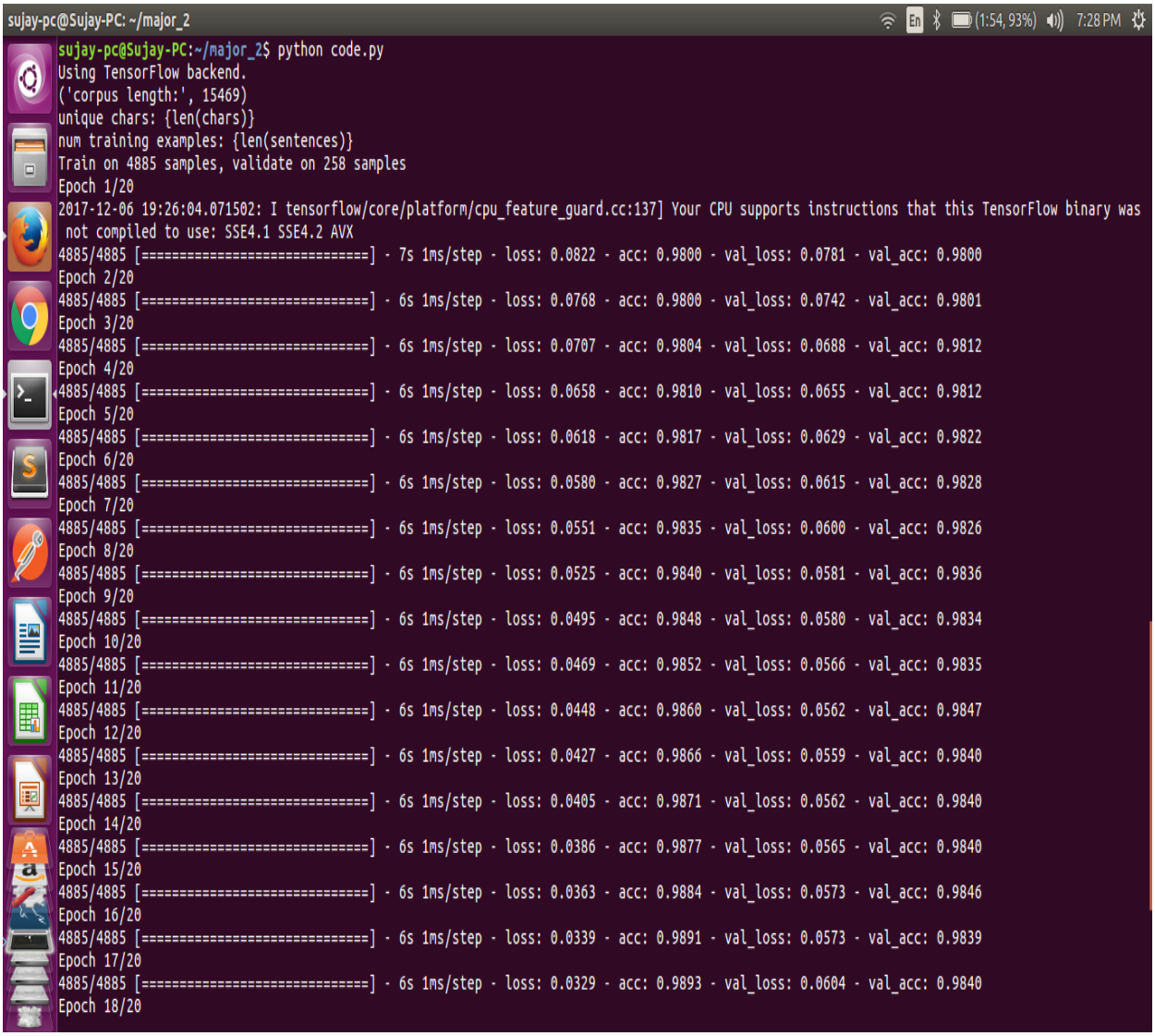
Response to the string searched

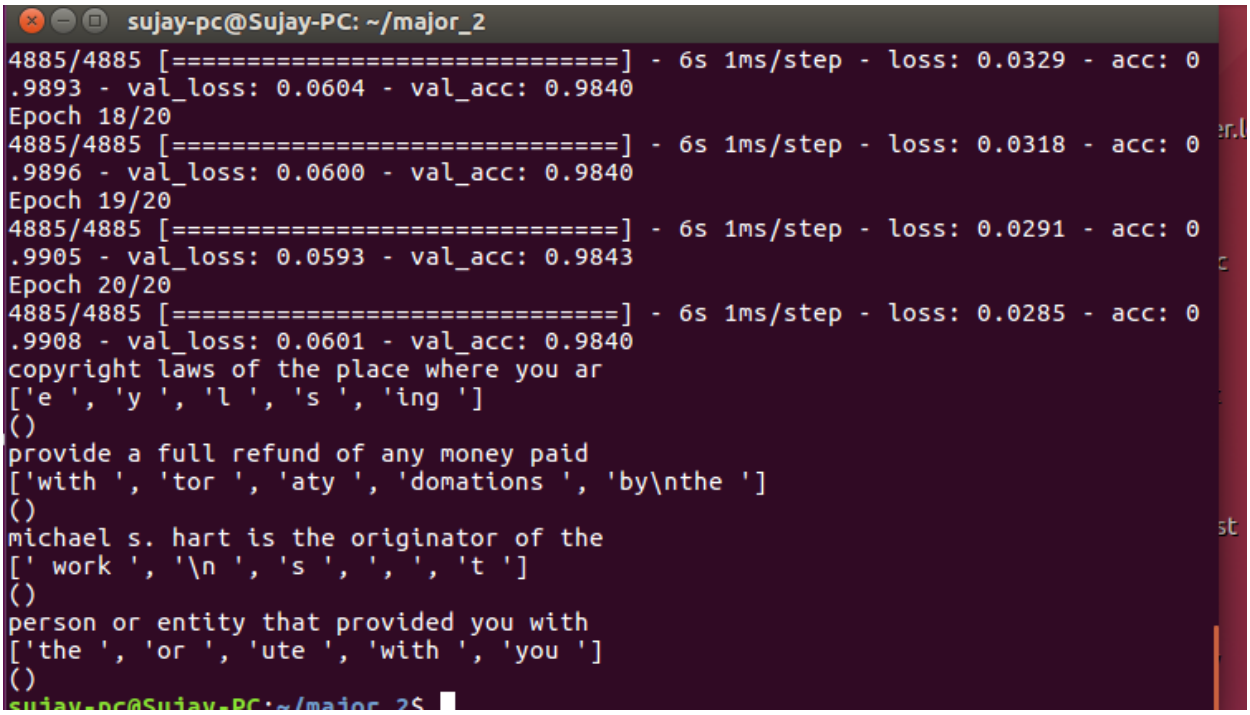


Response to the string not present in dataset



LSTM


sujay-pc@Sujay-PC: ~/major_2
(1:54, 93%)
7:28 PM
sujay-pc@Sujay-PC:~/major_2$ python code.py
Using TensorFlow backend.
('corpus length:', 15469)
unique chars: {len(chars)}
num training examples: {len(sentences)}
Train on 4885 samples, validate on 258 samples
Epoch 1/20
2017-12-06 19:26:04.071502: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
4885/4885 [==============================] - 7s 1ms/step - loss: 0.0822 - acc: 0.9800 - val_loss: 0.0781 - val_acc: 0.9800
Epoch 2/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0768 - acc: 0.9800 - val_loss: 0.0742 - val_acc: 0.9801
Epoch 3/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0707 - acc: 0.9804 - val_loss: 0.0688 - val_acc: 0.9812
Epoch 4/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0658 - acc: 0.9810 - val_loss: 0.0655 - val_acc: 0.9812
Epoch 5/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0618 - acc: 0.9817 - val_loss: 0.0629 - val_acc: 0.9822
Epoch 6/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0580 - acc: 0.9827 - val_loss: 0.0615 - val_acc: 0.9828
Epoch 7/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0551 - acc: 0.9835 - val_loss: 0.0600 - val_acc: 0.9826
Epoch 8/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0525 - acc: 0.9840 - val_loss: 0.0581 - val_acc: 0.9836
Epoch 9/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0495 - acc: 0.9848 - val_loss: 0.0580 - val_acc: 0.9834
Epoch 10/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0469 - acc: 0.9852 - val_loss: 0.0566 - val_acc: 0.9835
Epoch 11/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0448 - acc: 0.9860 - val_loss: 0.0562 - val_acc: 0.9847
Epoch 12/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0427 - acc: 0.9866 - val_loss: 0.0559 - val_acc: 0.9840
Epoch 13/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0405 - acc: 0.9871 - val_loss: 0.0562 - val_acc: 0.9840
Epoch 14/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0386 - acc: 0.9877 - val_loss: 0.0565 - val_acc: 0.9840
Epoch 15/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0363 - acc: 0.9884 - val_loss: 0.0573 - val_acc: 0.9846
Epoch 16/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0339 - acc: 0.9891 - val_loss: 0.0573 - val_acc: 0.9839
Epoch 17/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0329 - acc: 0.9893 - val_loss: 0.0604 - val_acc: 0.9840
Epoch 18/20


sujay-pc@Sujay-PC: ~/major_2
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0329 - acc: 0
.9893 - val_loss: 0.0604 - val_acc: 0.9840
Epoch 18/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0318 - acc: 0
.9896 - val_loss: 0.0600 - val_acc: 0.9840
Epoch 19/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0291 - acc: 0
.9905 - val_loss: 0.0593 - val_acc: 0.9843
Epoch 20/20
4885/4885 [==============================] - 6s 1ms/step - loss: 0.0285 - acc: 0
.9908 - val_loss: 0.0601 - val_acc: 0.9840
copyright laws of the place where you ar
['e ', 'y ', 'l ', 's ', 'ing ']
()
provide a full refund of any money paid
['with ', 'tor ', 'aty ', 'domations ', 'by\nthe ']
()
michael s. hart is the originator of the
[' work ', '\n ', 's ', ', ', 't ']
()
person or entity that provided you with
['the ', 'or ', 'ute ', 'with ', 'you ']
()
sujay-pc@Sujay-PC:~/major_2$

9. CONCLUSION

One of the most important principle of Software Design is to keep the software simple yet effective. This has been the guiding force during the design phase of our software. This project will populate results on the basis of predictions performed on the basis of data set as the user types the query.

BIBLIOGRAPHY AND REFERENCES

1. Joseph B. Sidowski, On-line computer text processing: A tutorial in: Behavior Research Methods & Instrumentation, vol. 6, no.2, pp. 159-166, 1974.

2. Boyer, R. S.; Moore, J. S. A fast string searching algorithm. Commun. ACM 20, pp. 762-772, 1977.

3. Donald E. Knuth, James H. Morris, Vaughan R. Pratt, Fast Pattern Matching In Strings in: Siam, Vol. 6, No. 2, pp. 323-350, June 1977.

4. R. Nigel horspool, Practical fast searching in strings in: Software-Practice and Experience, vol. 10, pp. 501-506, 1980.

5. Timo Raita, Tuning The Boyer-Moore-Horspool String Searching Algorithm in:Software-Practice And Experience, Vol. 22(10), pp. 879-884, October 1992.

6. Nimisha Singla, Deepak Garg, String Matching Algorithms and their Applicability in various Applications in: International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, pp.156-161, January 2012.