DATA ANALYTICS PIPELINE FOR A-VIEW SYSTEM

Summer Internship 2019

Submitted in fulfillment for Internship Project

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai

By
Data Analytics Group

Principal Investigator: Professor S Sudarshan
Project Mentor and Guide: Sukla Nag

July 4, 2019

Team Members

1. Mayank Agarwal
   PES University

2. Ayush Agarwal
   Amity School of Engineering and Technology, Noida

3. Divyakumar Patel
   G H Patel College of Engineering & Technology (GCET), GTU University, Gujarat

4. Sakshi Sharma
   Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

5. Sanskriti Agrawal
   Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

The project report entitled

Data Analytics Pipeline for A-VIEW System

is

Submitted by

Divyakumar Patel
160110107012
B.E. Computer Engineering
G H Patel College of Engineering and Technology,
Gujarat Technological University, Gujarat

Under the guidance of

Ms. Sukla Nag
Senior Project Manager at IIT Bombay

Indian Institute of Technology Bombay
Powai, Mumbai

Ekalavya Summer Internship 2019

Summer Internship 2019
Project Approval Certificate

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

The project entitled Data Analytics Pipeline of A-VIEW System submitted by Mr. Mayank Agarwal, Mr. Ayush Agarwal, Mr. Divyakumar Patel, Ms. Sakshi Sharma and Ms. Sanskriti Agrawal is approved for Summer Internship 2019 programme from 19th May 2019 to 9th July 2019, at The Department of Computer Science and Engineering, IIT Bombay on successful completion of their project.

Prof. S Sudarshan	Mrs. Sukla Nag
Dept of CSE, IITB	Dept of CSE, IITB
Principal Investigator	Mentor Project Incharge

Place : IIT Bombay, Mumbai
Date : July 4, 2019

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause a disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Mayank Agarwal
PES University

Mr. Ayush Agarwal
Amity School of Engineering and Technology, Noida

Mr. Divyakumar Patel
GCET College, GTU University, Gujarat

Ms. Sakshi Sharma
PDPM IIITDM Jabalpur

Ms. Sanskriti Agrawal
PDPM IIITDM Jabalpur

Date: July 4, 2019

ABSTRACT

A-VIEW is an advanced multi-modal, multi-platform, collaborative e-learning solution which allows an instructor to teach or interact with a large number of learners transcending geographies on a real-time basis through live audio video streaming and synchronized content sharing. A-VIEW allows the instructor to perform live evaluation of the learners and to get real-time feedback from attendees on the go. This project aims to find out the number of operational and non-operational queries asked during such a lecture. The required data is collected through different channels where the video is made available. This analysis will give the administration team of A-view the required data to allocate their resources in a more structured manner. The crucial part is recognizing if a sentence asked or said is primarily a question or not and then an operational query or not. We have used a Recurrent Neural Network for the first purpose and a Naive Bayes classifier for the second. This report elaborates and summarizes the working of this pipeline created at IIT Bombay.

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide Prof. S Sudarshan. He made us believe in ourselves and push our limits to emerge as a better Computer Engineer.

We would also like to thank our project mentor Mrs Sukla Nag for her exemplary guidance, valuable information and constant encouragement throughout the project. They were always there to show us the right track when needed help. With help of her brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned.

During the internship, we got a chance to observe the stronger and weaker areas of our technical and non-technical profiles. This helped us in paying close attention to weaker aspects which eventually helped us in progressing and ultimately completing this project.

We would also like to acknowledge with much appreciation the crucial role of Mr. Rahul Kharat and Mr. Dilip Sable who ensured smooth working during the internship.

Last but not the least, we wholeheartedly thank all other colleagues working under different projects in the same internship program for helping us evolve better with their critical advice and active involvement.

Contents

Chapter 1

Introduction

1.1 Project Overview

The intent of this project is to segregate operational and non operational data from given youtube videos and chat messages.

Operational data corresponds to problems related to glitches in communication channel between the students and the teacher. Non Operational data refer to the rest sentences which comprise of doubts, greetings, etc.

The operational questions obtained can help the administration of the center to get a clear picture of errors in system connections and these errors can be rectified. With little tweaks, the project can be used to find which remote centre is in need of system servicing as the number of operational queries generated by a remote center can be found out. .

By improving the system connectivity, aview online virtual classrooms will be easily set up and students can access all educational videos without any disturbance.

Upgrading the aview system, live meetings, presentations and lectures can be conducted on a large scale.

1.2 What is A-VIEW System?

A-VIEW is an advanced multi-modal, multi-platform, collaborative e-learning solution which allows an instructor to teach or interact with a large number of learners transcending geographies on a real-time basis through live audio video streaming and synchronized content sharing. A-view records chat data in SQL tables & log data in text files.

A-VIEW allows the instructor to perform live evaluation of the learners and to get real-time feedback from attendees on the go.

A-VIEW can also act as an online meeting tool that can support online social collaboration and interactions with multiple users from various locations simultaneously.

Online virtual classrooms can be easily set up with A-VIEW leveraging its multi-screen display capability. The multi screen display capability of A-VIEW provides a near life classroom experience where the instructor and the content being shared can be seen simultaneously, legibly on large screens.

Ask a Question Program using A-VIEW where IIT Bombay Professors answer students/faculties questions in their chosen field of Engineering.

A-VIEW can be used as an effective tool to conduct online workshops by connecting experts and audience from around the globe through its live video and audio streaming capabilities. This clubbed with its multi-modal and multi-display capabilities can deliver an unforgettable experience in conducting online workshops.

A-VIEW was used to train more than 25000 teachers through online workshops organized by IIT Bombay.

A-VIEW can be used to interact with up to 8 simultaneous users at a time. This capability of A-VIEW lets it pitch itself as a competitive online meeting and social collaboration tool.

The benefits of opting A-VIEW for setting up online classes and meetings are:

- A-VIEW is free for Higher Education in India
- Users can contribute to the content
- Designed to act as Educational and Meeting tool simultaneously
- Highly Adaptable
- Variable Number of Displays
- Low Bandwidth Usage (Rural Areas)
- Highly Customizable
  - For Educational System
  - For Multiple Languages
  - For Existing Infrastructure

1.3 What is Data Analytics?

Data Analysis is the process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains. In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.

Analysis refers to breaking a whole into its separate components for individual examination. Data analysis is a process for obtaining raw data and converting it into information useful for decision-making by users. Data are collected and analyzed to answer questions, test hypotheses or disprove theories.

Data analytics is a broad term that encompasses many diverse types of data analysis. Any type of information can be subjected to data analytics techniques to get insight that can be used to improve things.

For example, manufacturing companies often record the runtime, downtime, and work queue for various machines and then analyze the data to better plan the workloads so the machines operate closer to peak capacity.

Types of Data Analytics:

Data Analytics is broken down into four basic types.

- **Descriptive Analytics:** describes what has happened over a given period of time. Have the number of views gone up? Are sales stronger this month than last?
- **Diagnostic Analytics:** focuses more on why something happened. This involves more diverse data inputs and a bit of hypothesizing. Did the weather affect beer sales? Did that latest marketing campaign impact sales?
- **Predictive Analytics:** moves to what is likely going to happen in the near term. What happened to sales the last time we had a hot summer? How many weather models predict a hot summer this year?
- **Prescriptive Analytics:** suggests a course of action. If the likelihood of a hot summer is measured as an average of these five weather models is above 58%, we should add an evening shift to the brewery and rent an additional tank to increase output.

Data analytics is important because it helps businesses optimize their performances. Implementing it into the business model means companies can help reduce costs by identifying more efficient ways of doing business and by storing large

amounts of data.

Gaming companies use data analytics to set reward schedules for players that keep the majority of players active in the game. Content companies use many of the same data analytics to keep you clicking, watching, or re-organizing content to get another view or another click.

Data analytics techniques can reveal trends and metrics that would otherwise be lost in the mass of information. This information can then be used to optimize processes to increase the overall efficiency of a business or system.

Some of the sectors that have adopted the use of data analytics include the travel and hospitality industry, where turnarounds can be quick. This industry can collect customer data and figure out where the problems, if any, lie and how to fix them.

Healthcare combines the use of high volumes of structured and unstructured data and uses data analytics to make quick decisions. Similarly, the retail industry uses copious amounts of data to meet the ever-changing demands of shoppers. The information retailers collect and analyze can help them identify trends, recommend products, and increase profits.

Chapter 2

Technologies Used

Figure 2.1: Technologies Used

- **Hypertext Markup Language (HTML)** is the markup language designed to be displayed by a browser. HTML 5 is currently used as a standard.
- **Cascading Style sheets(CSS)** is a simple mechanism to add style to HTML pages. Eg: - Increasing or decreasing margins, etc.
- **JavaScript (JS)** is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. It helps in making the web page responsive.
- **Python** is an interpreted, high-level programming language which is widely used in the purpose of data analytics. The wide variety of libraries encased in it form the perfect tools required by a data analyst.
- **Keras** is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- **Flask** is a micro web framework written in Python.It is called so because it doesnt require libraries or tools. It can directly take input through routes process and return output.
- **Selenium** is a portable framework for testing web applications.
- **Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides powerful APIs to plot graphs. We have used it to plot our graph of loss vs epochs , accuracy vs epochs, etc.
- **Numpy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Chapter 3

Installation

This chapter covers all the description and installation procedure of all the tools required in this project. All the softwares and packages used are Open Source software and freely available.

3.1 System Requirements

We have developed the whole system on the following hardware, and have used the following software.

3.1.1 Hardware Requirements

3.1.1.1 Hardware

- Quad core Intel i5 8th gen processor or multiprocessor-based computer with a 2Ghz or greater processor
- 64-bit system
- Network Interface Card

3.1.1.2 Disk Space

- 1 GB space for inferencing
- 5-6 GB for training

3.1.1.3 Memory

4 GB or more (recommended)

3.1.2 Software Requirements

- Operating System : Linux (Ubuntu 18.04)
- Web Browser : Chrome (Version 73.0.3683.86)

- Text Editor : Vim

3.2 DeepSpeech

DeepSpeech is an open source Speech-To-Text engine, using a model trained by machine learning techniques based on Baidu's Deep Speech research paper[10]. Project DeepSpeech uses Google's TensorFlow to make the implementation easier.

3.2.1 Prerequisites

- Python3.6[11]
- Git Large File Storage[12] (git-lfs)
- Mac or Linux environment

3.2.2 Downloading the Source Code

After the installation of git-lfs, clone the DeepSpeech repository[13] using the following command. Cloning process might take some time to complete, as a binary file of 1.7GB is also downloaded alongside the repository files.

```
$ git lfs clone https://github.com/mozilla/DeepSpeech.git
```

Since we are interested in performing Transfer Learning the model by using Indian Accent data, we choose the ***"transfer-learning2"*** branch of the DeepSpeech repository. Hence to use that instead of the default ***"master"*** branch, we checkout to the specific branch.

```
$ git checkout transfer-learning2
```

To use the pre-trained model of DeepSpeech for deriving results, we need to download that model as:

```
$ wget
https://github.com/mozilla/DeepSpeech/releases/download/v0.5.1/
deepspeech-0.5.1-models.tar.gz

$ tar xvfz deepspeech-0.5.1-models.tar.gz
```

Now, we download the checkpoints created while training the original model by the DeepSpeech officials, so that we can resume our training from the desired checkpoint as per our requirements. The checkpoint archive can be downloaded from: https://github.com/mozilla/DeepSpeech/releases/download/v0.5.1/deepspeech-0.5.1-checkpoint.tar.gz.

After downloading and extracting the above archive, we need to prepare the directory structure for the successful execution of the learning process. For ease, we will name the extracted checkpoint folder as **"checkpoints"**. This folder will be needed inside the DeepSpeech folder.

For the purpose of training, we require data in the form of speech files and the sentences which they correspond to. Since we were not having data to train on Indian English accent, we have used a dataset offered by Mozilla itself under the name of **Common Voice**[14]. This dataset is Open Source and can be downloaded from (https://voice.mozilla.org/en/datasets). This contains nearly 26000 Indian accent audio files along with six different files to map the audio files to the sentence. These files are: **test.tsv, train.tsv, dev.tsv, validated.tsv, invalidated.tsv, other.tsv**. These files are those which map a sentence to their corresponding audio file.

These files have a common structure which contains the following columns:

- **client_id:** Speaker ID.
- **path:** Relative path of the audio file.
- **sentence:** Supposed transcription of the audio.
- **upvotes:** Number of people who said audio matches the text.
- **downvotes:** Number of people who said audio does not match text.
- **age:** Age of the speaker, if speaker reported it.
- **gender:** Gender of the speaker, if speaker reported it.
- **accent:** Accent of the speaker, if speaker reported it.

Since the dataset as a whole contains accent data of many languages, while we need only Indian accent, so an initial preprocessing is required to extract only Indian accent data from these files. We scan every file and observe the **accent** column, if the label is **"indian"**, we retain the audio file corresponding it, otherwise we delete the file in order to reduce unwanted data. Following this process for every file, will get us the actual content for which we are concerned about.

After the completion of the above step, we now have to prepare the structure of this directory. Audio files must be present inside the **"clips"** folder, while the other six files in the current directory itself. We name this directory as **"tsv_dir"**.

3.2.3 Creating & Activating the virtual environment

In creating a virtual environment we will create a directory containing a python3 binary and everything needed to run deepspeech. For ease, we will name the virtual environment as venv. We will create it using this command:

```
$ cd DeepSpeech
$ virtualenv --python=python3 venv
```

Once this command completes successfully, the environment will be ready to be activated. Each time we need to work with DeepSpeech, we have to activate this virtual environment. This is done with this simple command:

```
$ source venv/bin/activate
```

The directory structure after executing these commands will be as depicted in the image below:

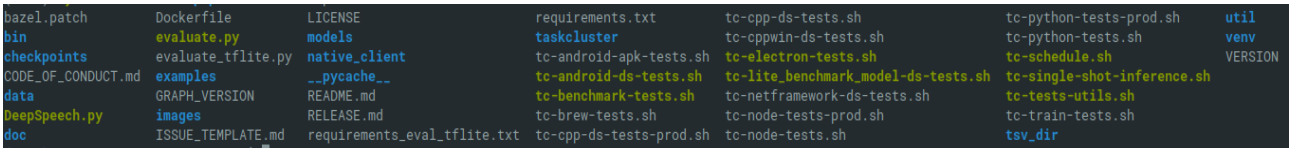


Figure 3.1: Directory Structure

3.2.4 Installing Dependencies

After following all of the above steps, we will install all the requirements which are necessary to run the model. These dependencies are listed in the **requirements.txt** file, we can install these using the following command:

```
$ pip3 install -r requirements.txt
```

After installing all the dependencies from requirements.txt, we have few more dependencies which must be installed to make sure that our model runs smoothly. These can be installed via the commands listed below.

```
$ pip3 install $(python3 util/taskcluster.py --decoder)
$ sudo apt-get install sox libsox-fmt-mp3
```

3.2.5 Converting data to required format

Since we have created the required structure of the directory, also we have gathered the data which was required for training the model; we will now bring this data into a form that DeepSpeech understands, and for that, we have to run the CommonVoice v2.0 importer (bin/import_cv2.py) as:

```
$ bin/import_cv2.py --filter_alphabet data/alphabet.txt tsv_dir
```

The above command will contain for each Once the import is done, the clips sub-directory will contain for each required .mp3 an additional .wav file. It will also add the following .csv files:

- clips/train.csv
- clips/dev.csv
- clips/test.csv

All entries in these CSV files refer to their samples by absolute paths. So moving this sub-directory would require another import or tweaking the CSV files accordingly. To use Common Voice data during training, validation and testing, you pass (comma separated combinations of) their filenames into **–train_files, –dev_files, –test_files** parameters of **DeepSpeech.py**.

3.2.6 Checkpointing

During training of a model so-called checkpoints will get stored on disk. This takes place at a configurable time interval. The purpose of checkpoints is to allow interruption (also in the case of some unexpected failure) and later continuation of training without losing hours of training time. Resuming from checkpoints happens automatically by just (re)starting training with the same **–checkpoint_dir** of the former run.

Be aware however that checkpoints are only valid for the same model geometry they had been generated from. In other words: If there are error messages of certain Tensors having incompatible dimensions, this is most likely due to an incompatible model change. One usual way out would be to wipe all checkpoint files in the checkpoint directory or changing it before starting the training.

3.2.7 Exporting a model for inference

After the completion of training process, the updated trained model is needed to be exported so that we can use it for transcription instead of the original model. If the **–export_dir** parameter is provided, a model will have been exported to this directory during training.

3.2.8 Starting the training/Fine Tuning

To start the fine tuning of the original model, we will tweak some settings of the model and specify few hyperparameters as tags in the command which will play a role in the training process. Before starting the training, we have transferred the **train.csv, test.csv** and **dev.csv** from tsv_dir/clips to project directory for faster processing and retrieval. Some of the important directories created after the training process are:

- **fine_tuning_checkpoints**: This folder will save the training and validation checkpoints after every training and validation epoch. If we want to resume the training process at a later stage, these checkpoints are useful in these scenarios. These checkpoints are also used to make predictions on the test set.

- **finetune_export**: To export the model and predict on individual audio files, this directory saves the model as **output_graph.pb** file.

The below command is used to start the training process with above specifications and a learning rate of 0.0001 and 3 epochs.

```
$ python3 DeepSpeech.py
  --n_hidden 2048
  --initialize_from_frozen_model models/output_graph.pb
  --checkpoint_dir fine_tuning_checkpoints
  --epoch 3
  --train_files train.csv
  --dev_files dev.csv
  --test_files test.csv
  --learning_rate 0.0001
  --decoder_library_path new_native_client/libctc_decoder_with_kenlm.so
  --alphabet_config_path data/alphabet.txt
  --lm_binary_path data/lm/lm.binary
  --lm_trie_path data/lm/trie
  --export_dir finetune_export/
  --source_model_checkpoint_dir checkpoints
```

After the export, we can use the generated output_graph.pb file to transcribe an audio file.

3.3 Google Charts

Google Charts[15] provides a perfect way to visualize data on your website. From simple line charts to complex hierarchical tree maps, the chart gallery provides a large number of ready-to-use chart types.

The most common way to use Google Charts is with simple JavaScript that you embed in your web page. You load some Google Chart libraries, list the data to be charted, select options to customize your chart, and finally create a chart object with an id that you choose. Then, later in the web page, you create a div with that id to display the Google Chart.

Charts are rendered using HTML5/SVG technology to provide cross-browser compatibility (including VML for older IE versions) and cross platform portability to iPhones, iPads, and Android. Your users will never have to mess with plugins or

any software. If they have a web browser, they can see your charts. In order to use Google Charts in our application, we have to add the following tag in out User Interface, preferably inside the head tags:

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
</script>
```

3.4 Flask

Flask[16] is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

3.4.1 Setting up Flask

- **Download source code of the application**
- **Create & Activate the virtual environment**:

```
$ virtualenv venv
$ . venv/bin/activate
```

- **Install the requirements**

```
$ pip install -r requirements.txt
```

- **Set Environment Variables**

```
$ export FLASK_APP=application.py
$ export FLASK_DEBUG=1
```

- **Run the application**

```
$ python -m flask run
```

3.5 Selenium

3.5.1 Introduction

Selenium is a portable framework for testing web applications. Selenium automates browsers. The selenium package is used to automate web browser interaction using various programming languages like C, Groovy, Java, Perl, PHP, Python, Ruby and Scala. It is open-source software, released under the Apache 2.0 license: web developers can download and use it without charge.

3.5.2 Installation Steps

1. pip3 install -U selenium
2. Selenium requires a driver to interface with your choice of browser. Download the driver for your latest browser.
3. Once the driver is downloaded, save the file in the same directory where you will use the Selenium.

   Reference : Install Selenium and its driver

3.5.3 Basic Usage and Example :

- Example 1 :
  - open a new Firefox browser
  - load the page at the given URL

```
from selenium import webdriver

browser = webdriver.Firefox()
browser.get('http://seleniumhq.org/')
```

Figure 3.2: Example 1

- Example 2 :
  - open a new Firefox browser
  - the Yahoo homepage
  - search for “seleniumhq”
  - close the browser

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

browser = webdriver.Firefox()

browser.get('http://www.yahoo.com')
assert 'Yahoo' in browser.title

elem = browser.find_element_by_name('p')  # Find the search box
elem.send_keys('seleniumhq' + Keys.RETURN)

browser.quit()
```

Figure 3.3: Example 2

3.6 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. Keras makes experimentation easy. [1]

The key features of Keras are:

- **User Friendliness:** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity:** A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.
- **Easy extensibility:** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- **Work with Python:** No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

3.6.1 Prerequisites

- Python 3.6:
- Tensorflow: Tensorflow can be installed by the command:

```
$ pip install tensorflow
```

3.6.2 Installing Keras

There are two ways to install keras:

- Install Keras from PyPI (recommended):

```
$ sudo pip install keras
```

  If you are using a virtualenv, you may want to avoid using sudo:

```
$ pip install keras
```

- Alternatively: install Keras from the GitHub source: First, clone Keras using git:

```
$ git clone https://github.com/keras-team/keras.git
```

  Then, cd to the Keras folder and run the install command:

```
$ cd keras
$ sudo python setup.py install
```

3.6.3 Configuring Keras backend

By default, Keras will use TensorFlow as its tensor manipulation library.
To change the backend:

If you have run Keras at least once, you will find the Keras configuration file at:
$HOME/.keras/keras.json

The default configuration file looks like this:
{
”image_data_format”: ”channels_last”,
”epsilon”: 1e-07,
”floatx”: ”float32”,
”backend”: ”tensorflow”
}

Simply change the field backend to "theano", "tensorflow", or "cntk", and Keras will use the new configuration next time you run any Keras code.

You can also define the environment variable KERAS_BACKEND and this will override what is defined in your config file :

KERAS_BACKEND=tensorflow python -c "from keras import backend"
Using TensorFlow backend.

3.7 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.[2]

3.7.1 Installation

Matplotlib and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

```
$ python -m pip install -U matplotlib
```

3.8 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python[3]

3.8.1 Installation

pandas and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

```
$ python -m pip install -U pandas
```

3.9 Sci-kit Learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.[4]

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

3.9.1 Prerequisites

- Numpy:

```
$ pip install numpy
```

- Matplotlib (Described above)
- Scipy

```
$ pip install scipy
```

- Pandas (Described above)

3.9.2 Installation

```
$ pip install scikit-learn
```

3.10 NLTK

The Natural Language Toolkit (NLTK) is a Python package for natural language processing. NLTK requires Python 2.7, 3.5, 3.6, or 3.7. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. [5]

3.10.1 Installation

```
$ pip install nltk
```

Chapter 4

Outline

The project's aim is achieved by building a pipeline which takes the inputs from all data paths and finds out the number of operational and non-operational queries in the input. The pipeline's components are

1. **Selenium Web Scraper:** The scraper is used for extracting youtube chats from the given youtube link. This process takes time as the process has to wait so that initial ads are completed and the video has to be seeked till end to find all the chats.

2. **Question Classifier:** The Question Classifier is used to segregate questions from other forms of sentences. It is a Recurrent Neural Network contaning a sing LSTM layer.

3. **Operational Classifier:** The input from previous component is a list of all questions in the input. This is then passed to this operational classifier. We use a Naive Bayes classifier for the purpose. Naive Bayes is used due to small input size and also the fact that operational and non-operational questions are not semantically very different, thus there may be problem reading into the semantic difference.

The completed pipeline is integrated into a flask environment which can be hosted on any server. The UI (described later) takes in link/chat file and provides a graphical analysis. This will help better distribution of A-view's resources as the analysis provides the much needed data required to find out which location needs more resources.

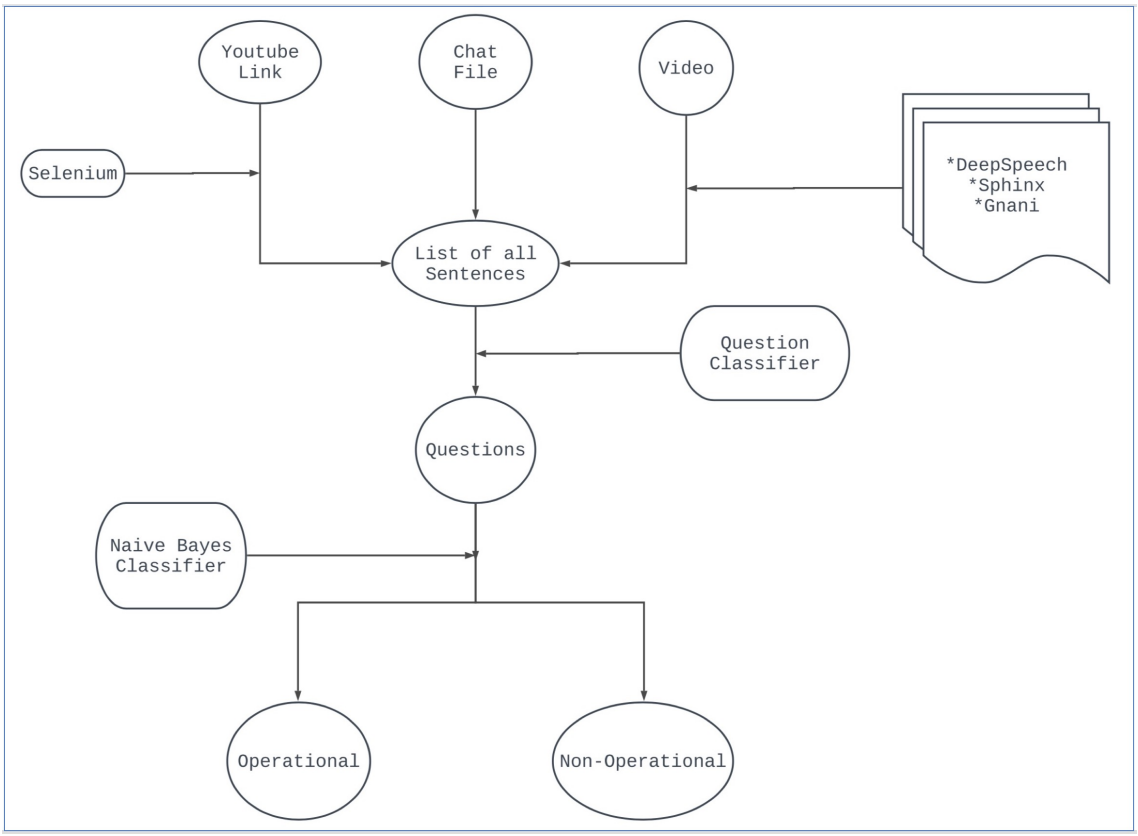


Figure 4.1: Flow Diagram

Chapter 5

Data Pathways

The three input channels are video of lecture, a-view chats and youtube link to the live streamed video. We extract the sentences from all these inputs. For video, we use selenium to scrape the sentences. The stt engine for video input currently used is google STT for the purpose of demonstration as no open source speech to text engine exists which gives a reasonable accuracy. Although any STT engine can be plugged in without any hassle. Since the A-VIEW chats are already structured, we just read them from a file.

Now, the extracted sentences are passed through a question classifier which assigns each sentence a probability which corresponds to the probability of that sentence being a question. The sentences with a probability higher than a fixed threshold are retained.

These sentences are then passed through a Naive Bayes Classifier which predicts if a question is operational or non-operational. We define operational as those queries which relate to working of communication channel between student and teacher.
Eg: - The audio is not working, Please tilt your camera, etc.

The pipeline takes input from three channels:

- **Youtube Live Chats:** This is the link to the video streamed during the lecture. We use this to extract the live chats and classify them.

- **A-view Chat file:** This contains the queries and answers on the a-view chat platform.

- **.mp4 Video:** This channel is yet to be implemented due to lack of open source speech to text engines. A speech to text engine requires huge amounts of data and a complex neural network which takes a lot of time to train. But the integration of the stt engine will be fairly simple. .

Chapter 6

Selenium web scraper

Selenium automates browsers.
The **"selenium"** package is used to automate web browser interaction from Python.

Selenium is a portable framework for testing web applications. Selenium provides a playback (formerly also recording) tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then be run against most modern web browsers. Selenium deploys on Windows, Linux, and macOS platforms. It is open-source software, released under the Apache 2.0 license: web developers can download and use it without charge.

6.1 Components of Selenium :

6.1.1 Selenium IDE

It allows for recording, editing, and debugging of functional tests. Scripts may be automatically recorded and edited manually providing autocompletion support and the ability to move commands around quickly. Scripts are recorded in Selenese, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser (click the link, select an option), and for retrieving data from the resulting pages.

6.1.2 Selenium client API

As an alternative to writing tests in Selenese, tests can also be written in various programming languages. These tests then communicate with Selenium by calling methods in the Selenium Client API. Selenium currently provides client APIs for

Java, C, Ruby, JavaScript, R and Python.

6.1.3 Selenium WebDriver

Selenium WebDriver accepts commands (sent in Selenese, or via a Client API) and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results.

6.1.4 Selenium Remote Control

Selenium Remote Control (RC) is a server, written in Java, that accepts commands for the browser via HTTP. RC makes it possible to write automated tests for a web application in any programming language, which allows for better integration of Selenium in existing unit test frameworks.

6.1.5 Selenium Grid

Selenium Grid is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub. Tests contact the hub to obtain access to browser instances. The hub has a list of servers that provide access to browser instances (WebDriver nodes), and lets tests use these instances.

6.2 Which part of Selenium is to be used?

6.2.1 Selenium Web Driver

If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Then you want to use Selenium WebDriver; a collection of language specific bindings to drive a browser – the way it is meant to be driven.
Selenium WebDriver is the successor of Selenium Remote Control which has been officially deprecated. The Selenium Server (used by both WebDriver and Remote Control) now also includes built-in grid capabilities.

6.2.2 Selenium IDE

If you want to

- create quick bug reproduction scripts
- create scripts to aid in automation-aided exploratory testing

Then you want to use Selenium IDE; a Chrome and Firefox add-on that will do simple record-and-playback of interactions with the browser.

Follow the link below for step by step installation and use of selenium :
https://pypi.org/project/selenium/

Chapter 7

Video Transcript Extraction

7.1 Introduction

The common way to recognize speech is the following: Take a waveform, split it at utterances by silences and then try to recognize what's being said in each utterance. This is done by taking all possible combinations of words and trying to match them with the audio. The best matching combination is then chosen.

7.2 Video to Audio Conversion with required configuration

7.2.1 Required Configuration

Before using any API for Speech to Text the video is required to be converted to audio with given properties:

- .wav file
- Sampling Rate:16000 Hz
- Mono Channel

7.2.2 ffmpeg

It is a very fast video and audio converter that also uses a high-quality polyphase filter to convert from one sampling rate to another as well as for resizing the video and changing other properties of video as well as audio.[19]

7.2.3 Pydub

This is a Python module that allows us to change the mp3 or .wav file to the required configuration as required by the google api for the purpose of our project.[21]

7.2.3.1 Dependencies

In order to open and save the non-wav files - like mp3, we would need ffmpeg which needs to be installed beforehand.

7.2.4 moviepy

This Python module allows us to edit videos and permits basic operations,(like cuts, concatenations, title insertions),video composting, video processing, and can read and write the most common video formats.[20]

7.2.4.1 Dependencies

MoviePy depends on the Python modules Numpy, imageio, Decorator, and tqdm, which will be automatically installed during MoviePy's installation.It can work on Windows/Mac/Linux, with Python 2.7+ and 3. It also depends on ffmpeg for video reading and writing.

7.3 Audio to Text Conversion

In many modern speech recognition systems, neural networks are used to simplify the speech signal using techniques for feature transformation and dimensionality reduction before HMM recognition.The analysis of different models for speech to text conversion that were tried with their advantages and shortcomings are described below.

7.3.1 Gnani API

It is an API that can be used to convert speech to text. This works pretty well on Indian accents[17]. It supports various languages:

- Hindi
- Indian English
- Tamil
- Telugu
- Gujarati
- Kannada

In order to use this API, we need to register ourselves on their site in order to generate the API key.

7.3.1.1 Requirements

- Audio Format Supported - wav
- Audio Sampling Rate - 16kHz
- Number of Channels - 1
- Encoding Format - pcm16

7.3.1.2 Restrictions of Gnani

Only 1000 grpc requests per user are allowed beyond which the user would be blocked. Due to this API not being open source, we could not use it for the purpose of our project despite of it giving considerably good accuracy for Indian accent English.

7.3.1.3 Results of tests performed on Gnani

ORIGINAL SPEECH	GNANI VERSION
very few engineering colleges provide a financial courses which are very important financial literacy is something that everyone must have so i want to know how important are these courses and if there should be any weightage that should be given to any of these courses.	very few engineering colleges provide a financial courses which are very important financial literacy is something that everyone must have so i dont know how important are these courses and a if the ssbci anemometers thats be given to these causes
Actually i wanted to know how many years would it take for me to become a professional kathak dancer as i am interested in this field and want to pursue it as my career but i dont know how much time would it take me to become perfect.	i wanted to know how many years would become a professional kathak dancer as i am interested in the scene and want to pursue it as my career but i dont know how much time would it take me to become pope

Figure 7.1: Results of test performed on Gnani

7.3.2 Sphinx

Sphinx[18] is an open source tool developed at Carnegie Mellon University, for speech recognition and speech to text conversion.The speech decoders come with acoustic models and sample applications. The available resources include in addition software for acoustic model training, Language model compilation and a public domain pronunciation dictionary (cmudict). Sphinx encompasses a number of software systems like:

Pocket Sphinx A version of Sphinx that can be used in embedded systems (e.g., based on an ARM processor). PocketSphinx is under active development and incorporates features such as fixed-point arithmetic and efficient algorithms for GMM computation. The project was cloned from github which is in active development.

7.3.2.1 Dependencies

Dependencies required for installing Pocket sphinx :

- gcc
- Automake
- Autoconf
- Libtool
- Bison
- Swig
- Python-dev
- Libpulse-dev

7.3.2.2 Requirements for Sphinx

Wave file needs to have the following configurations:

- mono channel
- normalized
- 16000 sample rate

7.3.2.3 Outcome:

Sphinx works really well with American English and the accuracy was close to 100%.But for Indian accent it does not provide good results.

7.3.2.4 Results of Sample tests

Comparison of sphinx for Indian accent and American accent The original version was taken using Google Api in python which gives the best results.

7.3.2.5 Adaptation of the model

Besides models, CMUSphinx provides some approaches for adaptation which should suffice for most cases when more accuracy is required. Adaptation is known to work well when you are using different recording environments (close-distance or far microphone or telephone channel), or when a slightly different accent (UK English or Indian English) or even another language is present. Adaptation, for example, works well if you need to quickly add support for some new language just by mapping a phoneset of an acoustic model to a target phoneset with the dictionary. Phoneset is like a dictionary which describes how a word is actually pronounced in different languages or accents.
Some examples are: These were taken from phonetic dictionary for Indian accent

- cause k ah z
- a_priori ey p r ay ao r ay
- abandon ax b ae n d ax n
- Abdomen ae b d ax m ax n
- photography f ax t oh g r ax f iy
- physics f ih z ih k

7.3.3 Google API using SpeechRecognition library

We first tried using the Google Speech Recognition API using the SpeechRecognition library. The SpeechRecognition library can be used for several popular speech APIs as a wrapper class. It contains a Recognizer class whose instance is used to recognize speech when audio is passed to it as a source. It provides a variety of functionalities for recognizing speech. The APIs that can be used with this library along with the corresponding methods of the Recognizer class that are available for them are listed below:

- Microsoft Bing Speech can be used using the recognize_bing() method.
- Google Web Speech API is supported by using the recognize_google() method.
- Google Cloud Speech can be used with the help of recognize_google_cloud()
- Houndify by SoundHound can be used to recognize speech by using the recognize_houndify() method.
- For IBM Speech to Text , recognize_ibm() method is available.
- CMU Sphinx is supported using the recognize_sphinx() method
- Wit.ai can be used with the recognize_wit() method.

The SpeechRecognition library contains a default API key hard-coded into it which is supported by Google Web Speech API. We preferred this option as this is convenient to use while the other six APIs require an API key authentication or user authentication.

7.3.3.1 Outcome:

The results of Google API were pretty good and quite accurate.

7.3.4 DeepSpeech

DeepSpeech is an open source Speech-To-Text engine, using a model trained by machine learning techniques based on Baidu's Deep Speech research paper. In order to make the implementation easier, it uses Google's Tensor-Flow. A variation of the paper named Deep Speech 1 forms the basis of the current model. As Baidu says, their work is really game-changing as long as they've implemented a couple of cool algorithms into their Deep Speech newest paper. They mention that their architecture is significantly simpler than other traditional speech-to-text systems. The most valuable they've done is that their newest functions or deep-learning based algorithms can perceive and filter the white noise effects, such as ground noise, reverberation, or speaker variation. In their words, Deep Speech "directly learns a function that is robust to such effects".

Additionally, they claim that the DeepSpeech doesn't use phenome dictionaries. On the contrary, according to the latest paper, the key to their approach is an optimized Recurrent Neural Network, that uses GPUs for parallelized training and effective generation. And, GPU-optimized training supports training on a huge amount of data, as well.

Furthermore, on a new noisy speech recognition dataset Baidu Deep Speech achieved 19.1% word prediction error, that is almost 10.4 whole percent less than average prediction error on the global market.

As DeepSpeech has been trained for foreign English accents, the results for audios recorded in Indian English accent were not pretty good. As an attempt to improve the performance of the model we attempted to try Transfer Learning by using Indian English accent data.

7.3.5 Transfer Learning:

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems

and from the huge jumps in a skill that they provide on related problems.[22]

One common approach for Transfer Learning is **Pre-trained Model Approach.**

- **Select the Source Model:** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
- **Reuse Model:** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
- **Tune Model:** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

In order to save time or get better performance, transfer learning can be used as optimization or shortcut.

Lisa Torrey and Jude Shavlik in their chapter on their book on Transfer Learning[**?**] describe three possible benefits to look for when using transfer learning:

1. **Higher start:** The initial skill (before refining the model) on the source model is higher than it otherwise would be.
2. **Higher slope:** The rate of improvement of skill during the training of the source model is steeper than it otherwise would be.
3. **Higher asymptote:** The converged skill of the trained model is better than it otherwise would be.

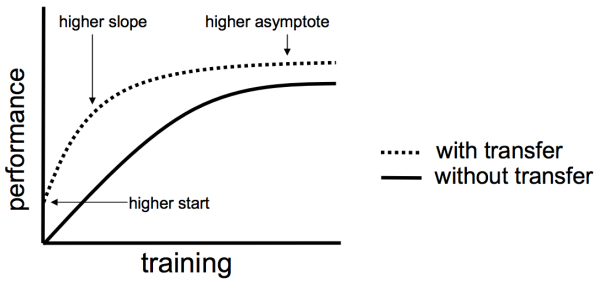


Figure 7.2: Learning Curve for Transfer Learning V/S Traditional Learning

We tried this technique of transfer learning on the DeepSpeech model due to the lack of time and training dataset. For training the entire model again a large amount of time and the dataset of the desired Indian English accent is required and to gain better results in a short span of time we used the dataset from the following source for applying transfer learning on the Mozilla DeepSpeech model:

7.3.6 About Dataset for Transfer Learning on DeepSpeech:

Link: https://voice.mozilla.org/en/datasets **(Common Voice Dataset)**

It is an open source, multi-language dataset of voices that anyone can use to train speech-enabled applications. Common Voice's multi-language dataset is already the largest publicly available voice dataset of its kind. Each entry in the dataset consists of a unique MP3 and corresponding text file. Many of the 2,454 recorded hours in the dataset also include demographic metadata like age, sex, and accent that can help train the accuracy of speech recognition engines. The dataset currently consists of 1,965 validated hours in 29 languages, but we're always adding more voices and languages. The Common Voice dataset complements Mozilla's open source voice recognition engine Deep Speech. However, for our desired purpose we required only the English accent voices which comprised 4% of the entire dataset. This included audios recorded by people from India and South Asia (India, Pakistan, Sri Lanka).

7.3.7 Methodology adopted for Transfer Learning

The official repository of DeepSpeech Open Source Project consists of several branches offering a plethora of tweaks in the original model setting. One of those branches, namely "transfer-learning2" offers the functionality of implementing the technique of Transfer Learning as discussed in the sections above.

In the original code of the branch, there were two methods in which we tried training the model by:

- Dropping the last two layers
- Dropping the last layer

Dropping the last two layers: Since the original model contains 6 layers. In this first trial, we tried fine tuning the model by training the last two layers of the pre-trained model. By training in this manner, we received results with a Word Error Rate (WER) of 0.621 and 66.40 % Loss. The reasons for this bad accuracy can be attributed to the scarcity of data and the computational power of GPUs.

The training took 36 hours upon 16,000 audio files with 3 epochs. Following results were received on testing using the test.csv file.

src is the source sentence in the audio file.
res is the result received from the fine-tuned model.

Test on test.csv - WER: 0.621259, CER: 0.423441, loss: 66.408829

WER: 1.400000, CER: 26.000000, loss: 106.340515
- src: "valgrind detected multiple memory leaks"

- res: "well in the late but a polar"

WER: 1.166667, CER: 23.000000, loss: 75.858665
- src: "worryingly the radio signal became weaker"
- res: "who are you there using all became meg"

WER: 1.000000, CER: 10.000000, loss: 36.138115
- src: "do you understand me"
- res: "to you and a tiny"

WER: 1.000000, CER: 16.000000, loss: 45.777679
- src: "however did you manage that"
- res: "we were human at"

WER: 1.000000, CER: 17.000000, loss: 49.965107
- src: "it's either all or nothing"
- res: "it is the out"

WER: 1.000000, CER: 10.000000, loss: 52.934818
- src: "soon ripe soon rot"
- res: "so i stood"

WER: 1.000000, CER: 17.000000, loss: 53.830944
- src: "he'll be ready on time"
- res: "the rather"

WER: 1.000000, CER: 15.000000, loss: 60.061100
- src: "handle that with kid gloves"
- res: "hanta wo to close"

WER: 1.000000, CER: 28.000000, loss: 62.839684
- src: "procrastination is the thief of time"
- res: "or outing ainsi"

WER: 1.000000, CER: 25.000000, loss: 64.615280
- src: "the cursor blinked expectantly"
- res: "a beeline"

Dropping the last layer: As opposed to the first approach, In this second trial, we tried fine tuning the model by training just the last layer of the pre-trained model. By training in this way, we received results with a Word Error Rate (WER) of 0.625 and 60.40 % Loss. Clearly, we can see the improvement in the loss percentage. Again, to improve the loss and the WER, we need lots of data along with the significant computational ability to train the model.

The training again took 36 hours upon 16000 audio files with 3 epochs. Following results were received on testing using the test.csv file.

Test on test.csv - WER: 0.625387, CER: 0.420284, loss: 60.096409

WER: 1.250000, CER: 10.000000, loss: 28.218716
- src: "do you understand me"
- res: "to you and to turn my"

WER: 1.166667, CER: 27.000000, loss: 64.553398
- src: "worryingly the radio signal became weaker"
- res: "what are you there uilleam we go"

WER: 1.000000, CER: 10.000000, loss: 28.748739
- src: "over neck and head headlong"
- res: "we left and he had long"

WER: 1.000000, CER: 12.000000, loss: 34.951168
- src: "what has become of him"
- res: "at a becoming"

WER: 1.000000, CER: 14.000000, loss: 36.344097
- src: "he'll be ready on time"
- res: "the retentive"

WER: 1.000000, CER: 13.000000, loss: 39.816517
- src: "it's either all or nothing"
- res: "it is there all out"

WER: 1.000000, CER: 21.000000, loss: 42.525055
- src: "and you couldn't release him"
- res: "adoniram"

WER: 1.000000, CER: 12.000000, loss: 48.656555
- src: "soon ripe soon rot"
- res: "sorry to a"

WER: 1.000000, CER: 17.000000, loss: 50.275631
- src: "handle that with kid gloves"
- res: "another would to close"

WER: 1.000000, CER: 14.000000, loss: 52.107201
- src: "many hands make light work"
- res: "may an may live"

Hence, as per our results with the two approaches, we can say that the second approach of fine-tuning the last layer was better in terms of loss and can be relied upon when a significant amount of data and high computational power is available to train the model.

Chapter 8

Question Classifier

This model assigns a number to each sentence which corresponds to the probability of that sentence being a question and then retains the sentences with probability beyond a certain threshold.

8.1 Explanation:

The classifier has the following components:

1. Cleaning
2. Splitting
3. Tokenizing
4. Glove vectors
5. Model structure
6. Training
7. Plotting results
8. Testing
9. Predicting

• 8.1.1 Cleaning:

There are no traditional questions vs non-questions datasets available. So we will have to make the dataset.

For the questions, we use the dataset https://www.kaggle.com/quora/question-pairs-dataset. [6] The dataset contains pairs of questions which are similar. Therefore we take only one column. For non-questions, we need sentences which are declarative. So we take both positive and negative sentences from the following link and and shuffle it[7]: https://www.kaggle.com/chaitanyarahalkar/positive-and-negative-sentences.

We store the questions in a list named questions_data and non_questions in another called non_questions_data

Now we have questions_data in a list and non_questions_data in another. We have to add labels to them.
1 - question
0 - Not a question

• 8.1.2 Splitting

We split the dataset into training and testing with a 9:1 ratio. Then the splits are checked to have correct shape.

• 8.1.3 Tokenizing

Tokenizer class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf. We deliberately exclude '?' from filters as they could be very important in predicting the sentence as a question.

The split datasets are tokenized using Tokenizer class. Also, we fix maximum length of vectors as 44. This is not too big which implies sparse vectors and hence wastage of space and computation. Neither is this number too small lose meaning of word. We use pad_sequences to limit or expand all word vectors to 44.

The labels need to be one hot encoded. By one hot encoding we mean, one value of vector hot at an instant.

An example could be:
red = (1,0,0)
blue = (0,1,0)

green = (0,0,1)

We also need to get validation dataset. The validation dataset is different from the test dataset that is also held back from the training of the model, but is instead used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models. We do this by splitting train datastet again into 9:1 pieces.

• 8.1.4 Glove Vectors

According to the project's official documentation:[8] "GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space."

These vectors enclose semantic meaning as they are trained with context words. We convert our tokenized inputs to glove vectors using an embedding layer. Thus, we set training as false. These vectors are available in different dimensions. If dimension is very less, we might not be able to represent the semantic meaning completely while higher dimensions tend to be sparse. The dimension 100 fits right for this purpose.

The glove file is saved in 'glove.twitter.27B' where .27B refers that the model is trained on approximate .27 Billion words.

We use these vectors to create and embedding dictionary. This will contain each word and its corresponding vector from the create vocabulary. Since this dictionary will be huge , we use pickle to save the object as a binary file.

• 8.1.5 Model Structure

The first layer is embedding layer as explained. Then we use an LSTM layer, as we want to use the context in sequence to learn. Then we use a dense layer with 2 neurons. This is our output layer with softmax as activation function. The function used in softmax layer is:

The curve:

Clearly the output of softmax is between 0 and 1 which in our model is the probability that input sentence is a question.

For clarification, this is the model's declaration:[9]

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

Figure 8.1: Softmax Equation

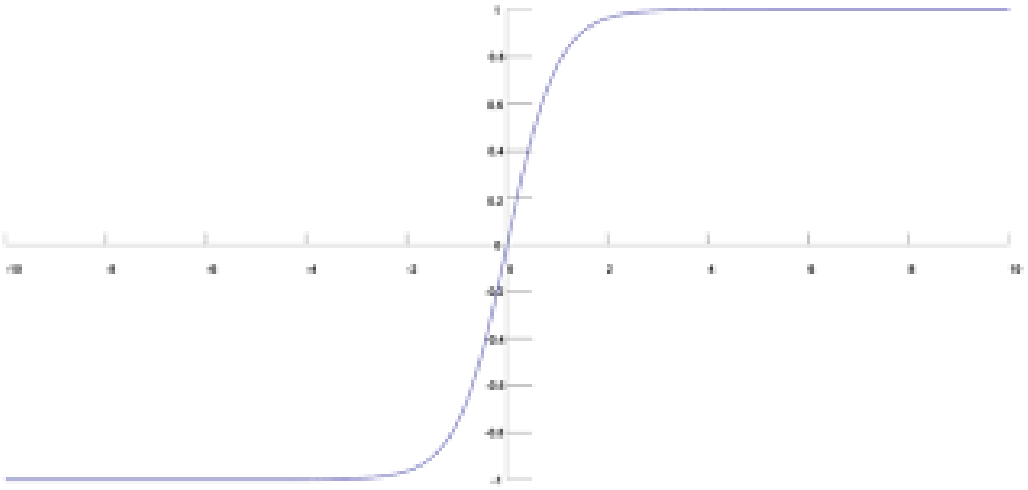


Figure 8.2: Softmax Curve

```
glove_model = models.Sequential()
glove_model.add(layers.Embedding(NB_WORDS, GLOVE_DIM,input_length=MAX_LEN))
glove_model.add(layers.LSTM(200, dropout_U = 0.2, dropout_W = 0.2))
glove_model.add(layers.Dense(2, activation='softmax'))
glove_model.summary()

glove_model.layers[0].set_weights([emb_matrix])
glove_model.layers[0].trainable = False
 #we are using pretrained weights. No need to train the initial layer
```

- ### 8.1.6 Training

Define the epochs and batch size:

```
NB_START_EPOCHS = 10 # Number of epochs we usually start to train with
```

```
# Size of the batches used in the mini-batch gradient descent
BATCH_SIZE = 512
```

Define the cost function and optimizer

```
glove_model.compile(optimizer='rmsprop'
                , loss='categorical_crossentropy'
                , metrics=['accuracy'])
#defining the model's training configs
```

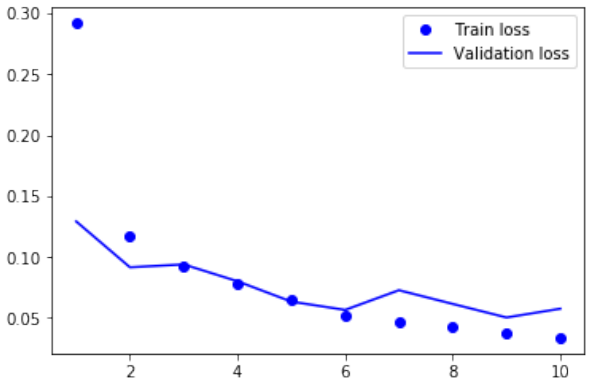
Then we train!

```
glove_history = glove_model.fit(X_train_emb    , y_train_emb
                      , epochs=NB_START_EPOCHS
                      , batch_size=BATCH_SIZE
                      , validation_data=(X_valid_emb, y_valid_emb)
                      , verbose=1)

                    #training the model

glove_history.history['acc'][-1]
```

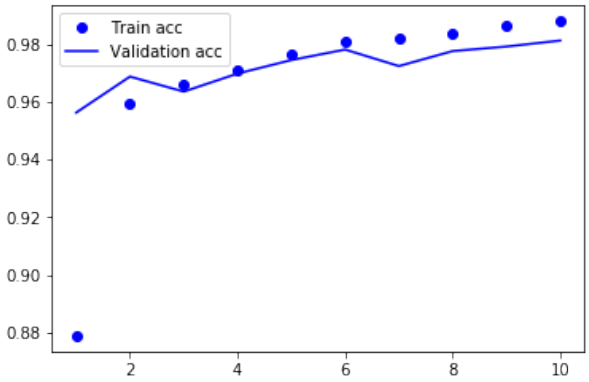
• 8.1.7 Plotting results

We define custom functions to plot loss and accuracy

Graph of Loss vs Epochs:



Graph of Accuracy vs Epochs



- ### 8.1.8 Testing

Testing can be done by in-build evaluate function

```
glove_model.fit(X_train_seq_trunc
              , y_train_oh
              , epochs=3
              , batch_size=BATCH_SIZE
              , verbose=0)
glove_results = glove_model.evaluate(X_test_seq_trunc,y_test_oh)
print('/n')
print('Test accuracy of word glove model: {0:.2f}%'.format
(glove_results[1]*100))
```

- ### 8.1.9 Predicting

We have to define a custom function to return probability when passed a sentence.

```
def predict_if_question(sentence):
    sentence_nstop = [sentence]
    sentence_nstop = np.array(sentence_nstop)
    sentence_seq = tk.texts_to_sequences(sentence_nstop)
    #print(sentence_seq)
    sentence_seq_trunc = pad_sequences(sentence_seq, maxlen=MAX_LEN)
    return glove_model.predict(sentence_seq_trunc)\\
```

This function just preprocesses sentence and predicts the probability

Some Results:

predict_if_question("explain about telecom industry")

array([[0.58534485, 0.41465515]], dtype=float32)

predict_if_question("i was explaining about telecom industry")

array([[0.81078756, 0.18921246]], dtype=float32)

As can be seen by just adding "I was", there is a huge change in probability. This is a clear sign our model has trained well.

We use this prediction function on the input sentences and pass only those with probability greater than a threshold.

Chapter 9

Naive Bayes Classifier: Operational vs Non-Operational

9.1 Introduction

Bayes theorem is one of the earliest probabilistic inference algorithms developed by Reverend Bayes (which he used to try and infer the existence of God no less and still performs extremely well for certain use cases.

It's best to understand this theorem using an example. Let's say you are a member of the Secret Service and you have been deployed to protect the Democratic presidential nominee during one of his/her campaign speeches. Being a public event that is open to all, your job is not easy and you have to be on the constant lookout for threats. So one place to start is to put a certain threat-factor for each person. So based on the features of an individual, like the age, sex, and other smaller factors like is the person carrying a bag?, does the person look nervous? etc. you can make a judgement call as to if that person is viable threat.

If an individual ticks all the boxes up to a level where it crosses a threshold of doubt in your mind, you can take action and remove that person from the vicinity. The Bayes theorem works in the same way as we are computing the probability of an event(a person being a threat) based on the probabilities of certain related events(age, sex, presence of bag or not, nervousness etc. of the person).

One thing to consider is the independence of these features amongst each other. For example if a child looks nervous at the event then the likelihood of that person being a threat is not as much as say if it was a grown man who was nervous. To break this down a bit further, here there are two features we are considering, age AND nervousness. Say we look at these features individually, we could design a model that flags ALL persons that are nervous as potential threats. However, it is likely that we will have a lot of false positives as there is a strong chance that minors present at the event will be nervous. Hence by considering the age of a person along with the 'nervousness' feature we would definitely get a more accu-

rate result as to who are potential threats and who aren't.

This is the 'Naive' bit of the theorem where it considers each feature to be independent of each other which may not always be the case and hence that can affect the final judgement.

In short, the Bayes theorem calculates the probability of a certain event happening(in our case, a message being spam) based on the joint probabilistic distributions of certain other events(in our case, the appearance of certain words in a message).

9.2 Bayes Theorem understanding from scratch

Now that we have our dataset in the format that we need, we can move onto the next portion of our mission which is the algorithm we will use to make our predictions to classify a message as spam or not spam. Remember that at the start of the mission we briefly discussed the Bayes theorem but now we shall go into a little more detail. In layman's terms, the Bayes theorem calculates the probability of an event occurring, based on certain other probabilities that are related to the event in question. It is composed of a prior(the probability that we are aware of or that is given to us) and the posterior(the probabilities we are looking to compute using the priors).

Let us implement the Bayes Theorem from scratch using a simple example. Let's say we are trying to find the odds of an individual having diabetes, given that he or she was tested for it and got a positive result.
In the medical field, such probabilities play a very important role as it usually deals with life and death situations.

We assume the following:

- 'P(D)' is the probability of a person having Diabetes. It's value is '0.01' or in other words, 1% of the general population has diabetes(Disclaimer: these values are assumptions and are not reflective of any medical study).

- 'P(Pos)' is the probability of getting a positive test result.

- 'P(Neg)' is the probability of getting a negative test result.

- 'P(Pos|D)' is the probability of getting a positive result on a test done for detecting diabetes, given that you have diabetes. This has a value '0.9'. In other words the test is correct 90% of the time. This is also called the Sensitivity or True Positive Rate.

- 'P(Neg|$\tilde{D}$)' is the probability of getting a negative result on a test done for detecting diabetes, given that you do not have diabetes. This also has a value of '0.9' and is therefore correct, 90% of the time. This is also called the Specificity or True Negative Rate.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figure 9.1: Conditional Probability

- **'P(A)'** is the prior probability of A occurring independently. In our example this is 'P(D)'. This value is given to us.
- **'P(B)'** is the prior probability of B occurring independently; i.e. 'P(Pos)'.
- **'P(A|B)'** is the posterior probability that A occurs given B. In our example this is 'P(D|Pos)'. That is, The probability of an individual having diabetes, given that, that individual got a positive test result. This is the value that we are looking to calculate.
- **'P(B|A)'** is the likelihood probability of B occurring, given A. In our example this is 'P(Pos|D)'. This value is given to us.

9.3 Naive Bayes understanding from scratch

Now that you have understood the ins and outs of Bayes Theorem, we will extend it to consider cases where we have more than feature.
Let's say that we have two political parties' candidates, 'Jill Stein' of the Green Party and 'Gary Johnson' of the Libertarian Party and we have the probabilities of each of these candidates saying the words 'freedom', 'immigration' and 'environment' when they give a speech:

- Probability that Jill Stein says 'freedom': 0.1 ——— 'P(F—J)'
- Probability that Jill Stein says 'immigration': 0.1 —— 'P(I—J)'
- Probability that Jill Stein says 'environment': 0.8 —— 'P(E—J)'
- Probability that Gary Johnson says 'freedom': 0.7 ——- 'P(F—G)'
- Probability that Gary Johnson says 'immigration': 0.2 — 'P(I—G)'
- Probability that Gary Johnson says 'environment': 0.1 — 'P(E—G)'

And let us also assume that the probability of Jill Stein giving a speech, 'P(J)' is '0.5' and the same for Gary Johnson, 'P(G) = 0.5'.

Given this, what if we had to find the probabilities of Jill Stein saying the words 'freedom' and 'immigration'? This is where the Naive Bayes Theorem comes into play as we are considering two features, 'freedom' and 'immigration'.

Now we are at a place where we can define the formula for the Naive Bayes' theorem:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Figure 9.2: Bayes Theorem

Here, 'y' is the class variable or in our case the name of the candidate and 'x1' through 'xn' are the feature vectors or in our case the individual words. The theorem makes the assumption that each of the feature vectors or words ('xi') are independent of each other.

To break this down, we have to compute the following posterior probabilities:

P(J|F,I) : Probability of Jill Stein saying the words Freedom and Immigration.

Using the formula and our knowledge of Bayes' theorem, we can compute this as follows: 'P(J|F,I)' = '(P(J) * P(F|J) * P(I|J)) / P(F,I)'. Here 'P(F,I)' is the probability of the words 'freedom' and 'immigration' being said in a speech.

'P(G|F,I)': Probability of Gary Johnson saying the words Freedom and Immigration.

Using the formula, we can compute this as follows: 'P(G|F,I)' = '(P(G) * P(F|G) * P(I|G)) / P(F,I)'

9.4 Understanding our dataset

The dataset consists of various fields which are not of importance while dealing with this classification problem. However, there are two columns which we are concerned about for this task. They are:

- **sentence**: This column consists of sentences which can be questions or general messages.
- **label**: This column consists of two values, 'N' which signifies that the sentence is not operational, and 'O' which signifies that the sentence is operational.

9.5 Implementation using Python

9.5.1 Training and testing sets

Splits our dataset into training and testing set so that we can ensure no data leakage and hence apply training and testing process on separate datasets.

- **"X_train"** is our training data i.e. **"sentence"** column.
- **"y_train"** is our training labels i.e. **"label"** column.
- **"X_test"** is our testing data from **"sentence"** column.
- **"y_test"** is our testing labels from **"label"** column.

9.5.2 Naive Bayes implementation using scikit-learn

Scikit-learn has several Naive Bayes implementations that we can use and so we do not have to do the math from scratch. We will be using sklearn **'sklearn.naive_bayes'** method to make predictions on our dataset.

Specifically, we will be using the Multinomial Naive Bayes implementation. This particular classifier is suitable for classification with discrete features (such as in our case, word counts for text classification). It takes in integer word counts as its input. On the other hand Gaussian Naive Bayes is better suited for continuous data as it assumes that the input data has a Gaussian (Normal) distribution.

9.5.3 Types of Naive Bayes:

Naive Bayes can be implemented using the scikit-learn(a Python library). This library offers three variations of this model:

- **Gaussian Naive Bayes:** It is used for classification when the features used for prediction take up continuous value and are not discrete and thus these values are assumed to be sampled from Gaussian distribution.
- **Multinomial Naive Bayes:** It is used in the case when the predictors take discrete values. For Example, it is widely used for classifying a document whether it belongs to a particular category,e.g. sports, politics, technology,

etc. The features that the classifier uses is the frequency of words in the document.

- **Bernoulli Naive Bayes:** It is a type of Multinomial Naive Bayes in which the features that we use to predict the target variable are boolean variables i.e. may take up a value yes or no, for example, if a word appears in a text or not. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

9.5.4 CountVectorizer (One-Hot Encoding):

One of the most basic ways we can numerically represent words is through One-Hot Encoding method. CountVectorizer creates a vector that has as many dimensions as your corpora has unique words. Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else. In short we can say that CountVectorizer implements both tokenization and occurence counting in single class.

Example:
Text: **"to be or not to be"**
Vocabulary: **'be': 0, 'not': 1, 'or': 2, 'to': 3**
Transformed vector: **[2, 1, 1, 2]**
One-Hot vector for: **"to"** and **"be"**
[[0, 0, 0, 1],
[1, 0, 0, 0]]

9.5.5 TfidfTransformer:

The classifiers and learning algorithms can not directly process the text documents in their original form, as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Therefore, during the preprocessing step, the texts are converted to a more manageable representation.

One common approach for extracting features from text is to use the bag of words model: a model where for each document the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored. In our case, we calculated the Tf-Idf term for each term in the dataset.

This is a common term weighting scheme in information retrieval, that has also found good use in document classification. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of

the training corpus.

9.5.6 Multinomial Naive Bayes Topic Classification Model Sample Results:

Sample Input: I am facing issues with audio connectivity
Sample Output: **Operational**

Sample Input: How to determine symptoms of fever
Sample Output: **Non Operational**

Sample Input: Am I visible?
Sample Output: **Operational**

Sample Input: How to improve this design?
Sample Output: **Non Operational**

Sample Input: Mic not set properly
Sample Output: **Operational**

9.5.7 Evaluating Classifier's Performance

Confusion Matrix: The confusion matrix is used to evaluate the quality of the output of a classifier on the test data set. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

9.5.8 Classification of YouTube live Chats : Model Evaluation

We have also used the youtube live chats to test our model and classify the operational and non-operation sentences.

RAKE - NLTK: To Extract Keywords From The Youtube Chat Data

RAKE short for Rapid Automatic Keyword Extraction algorithm, is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text.

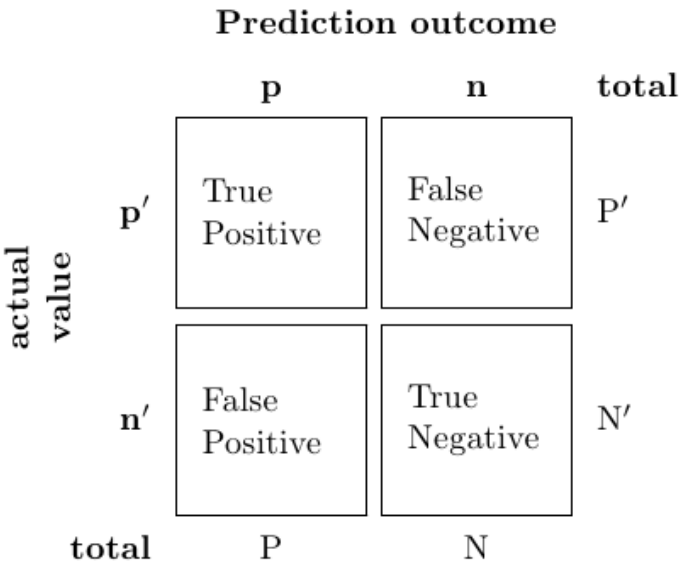


Figure 9.3: Confusion Matrix

	precision	recall	f1-score	support
Operational	0.87	0.90	0.89	477
Non-Operational	0.86	0.82	0.84	351
accuracy			0.87	828
macro avg	0.87	0.86	0.86	828
weighted avg	0.87	0.87	0.87	828

Figure 9.4: Classification Report

9.6 Implementation in R

9.6.1 Installation of Packages

- E1071: Functions required for naive bayes model
- Tm: Provides text mining framework
- Dplyr: for data manipulation
- Caret: has several functions that attempt to streamline the model building and evaluation process, as well as feature selection and other techniques.

9.6.2 Data Set

We download the file to our working directory in R/RStudio and read it as a dataframe object 'dat'.

```
>dat <- read.csv(file="chat.csv",head=FALSE,sep=",",stringsAsFactors=F)
```

The third column consists of Operational label 'O' and non-operational label 'N'. The second column consists of chat messages.

Randomize the dataset:

```
> set.seed(100)
rows<-sample(nrow(dat))
dat1<-dat[rows,]
dat1$label<-as.factor(dat1$label)
```

9.6.3 Tokenization

In this approach, we represent each word in a document as a token (or feature) and each document as a vector of features. In addition, for simplicity, we disregard the word order and focus only on the number of occurrences of each word i.e., we represent each document as a multi-set 'bag' of words.

Prepare a corpus of all the documents in the dataframe.

```
> corpus<-Corpus(VectorSource(dat1$sentence))
```

9.6.4 Data Cleaning

We clean up the corpus by eliminating numbers, punctuation, white space, and by converting to lower case. In addition, we discard common stop words such as "his", "our", "hadn't", couldn't", etc. We use the tm_map() function from the 'tm' package to this end.

```
corpus.clean <- corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords(kind="en")) %>%
  tm_map(stripWhitespace)
```

9.6.5 Document Term Matrix

We represent the bag of words tokens with a document term matrix (DTM). The rows of the DTM correspond to documents in the collection, columns correspond to terms, and its elements are the term frequencies. We use a built-in function from the 'tm' package to create the DTM.

```
dtm<-DocumentTermMatrix(corpus.clean)
```

9.6.6 Partitioning the Data

We create 75:25 partitions of the dataframe, corpus and document term matrix for training and testing purposes.

```
split<-round(nrow(dat1)*.75)

dat1.train<-dat1[1:split,]
dat1.test<-dat1[(split+1):nrow(dat1),]

dtm.train<-dtm[1:split,]
dtm.test<-dtm[(split+1):nrow(dat1),]

corpus.clean.train<-corpus.clean[1:split]
corpus.clean.test<-corpus.clean[(split+1):nrow(dat1)]
```

9.6.7 Feature Selection

We reduce the number of features by ignoring words which appear in less than five. To do this, we use 'findFreqTerms' function to identify the frequent words, we then restrict the DTM to use only the frequent words using the 'dictionary' option.

```
fivefreq<-findFreqTerms(dtm.train,5)
dtm.train.nb<-DocumentTermMatrix(corpus.clean.train,control=list
(dictionary=fivefreq))
dtm.test.nb<-DocumentTermMatrix(corpus.clean.test,control=list
(dictionary=fivefreq))
```

9.6.8 Model

The Naive Bayes text classification algorithm is essentially an application of Bayes theorem (with a strong independence assumption) to documents and classes.

```
classifier <- naiveBayes(trainNB, dat1.train$label, laplace = 1)
 pred <- predict(classifier, newdata=testNB)
 conf.mat <- caret::confusionMatrix(pred, dat1.test$label)
```

Confusion Matrix and Statistics:
(P-Prediction,R-Reference)

* R
P N O
N 448 84
O 35 260

9.6.9 Accuracy

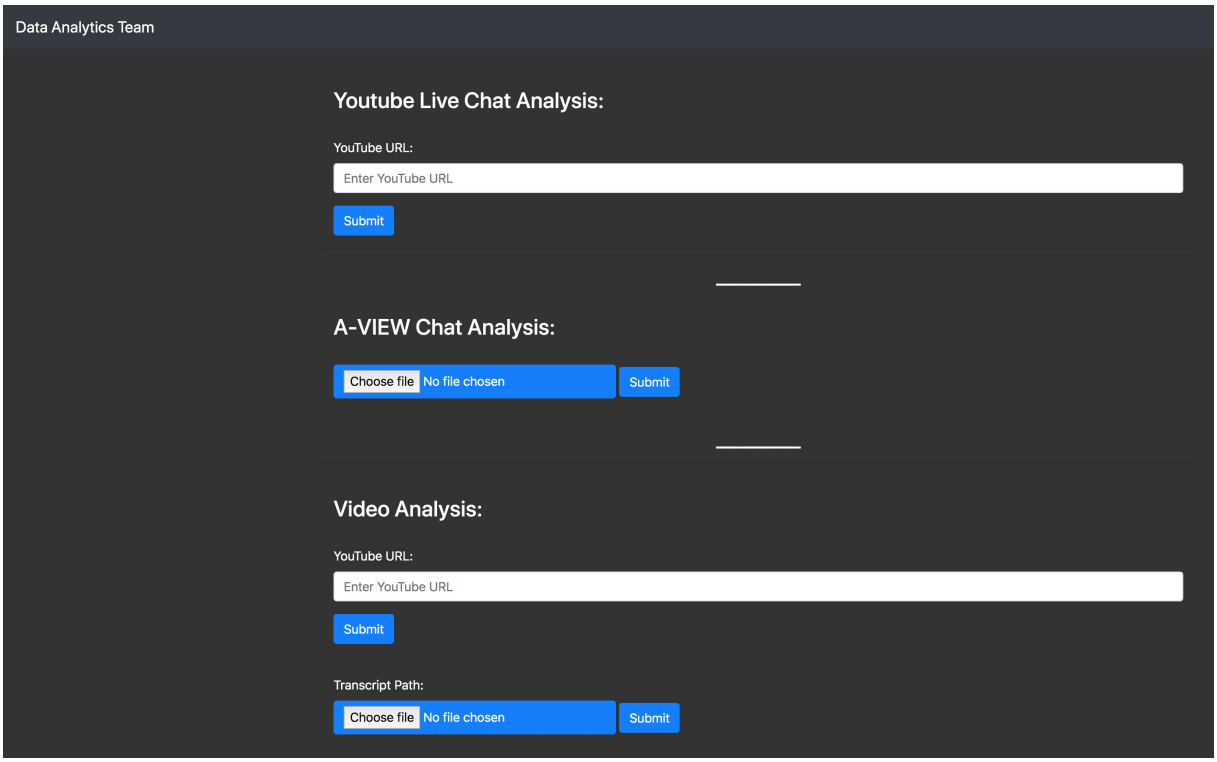
The accuracy for this model turns out to be 86%. We see that despite many simplifying assumptions, the Naive Bayes algorithm does reasonably well at predicting the correct sentiment classes.

Chapter 10

User Interface

The user interface is hosted on the local machine's port number 5000 after running

```
$ python application.py
```



The user interface has 3 major parts.

1. **Youtube chat analysis:** The user can paste the youtube link and the analysis will be performed on the live chats of the video. Note that, the video should have been live streamed before or else there are no chats to be analysed, thus resulting in no result. This will take a huge amount of time as the process needs to wait for the ads to be complete. Also, the process needs to be seeked till end so that no chats are missed.

2. **A-view chat analysis:** The user can upload a csv file of a-view chats and perform the analysis on the uploaded file. This does not need any scraping, therefore the processing is quite fast.

3. **Youtube Video Analysis:** This has two parts. One part converts video to audio and the other converts transcript to a pie graph representing results. The channel from audio to transcript should be done by an stt engine integrated by the user.

Chapter 11

Bottleneck

The pipeline for data analytics of A-view chat logs and video chats was successfully built. The only bottleneck is the absence of an open-source speech to text engine which gives satisfactory accuracy after accent transfer to Indian English. Indian English accent is required as target audience here includes Indian Professors and students. Open-source softwares which could be re-trained on Indian data and used:

- DeepSpeech
- Sphinx

The project has been developed keeping in mind substitution of a speech to text engine. Thus, integration of a speech to text engine will be seamless.

Chapter 12

Conclusions and Future Scope

Data Analytics enables organizations to analyse a mix of structured, semi-structured and unstructured data in search of valuable business information and insights. We tried to pre-train the DeepSpeech model for Indian accent but the results are not up to the expectations. So we have used output transcripts through propriety softwares for the preforming our tasks. Since the transcript didn't have any punctuation, so we used DeepSegment to provide proper punctuation to the sentences. The next part was data pre-processing using various NLP techniques to mould the data for our model. Redefined data is feed to a model which segregates questions from non-questions sentences. All the questions are collected and passed through a model which separates operational questions from the general question. We have developed a system which will help to analyse the audio and video problems faced by remote centres while interacting with the Instructor. This will help the local team and the Instructors to take necessary actions before the start of new session in future.

The future work in this area would be :

1. We can give a unique username name to different remote centres which can be used to uniquely identity which remote centre is having more problems.
2. DeepSpeech can be replaced by a new open source model which give appropriate text results.
3. Highest number of questions are asked from a particular remote centres.
4. According to different types of question asked, particular sessions can be arranged in future.
5. Log files must be saved instead of being overwritten, so that better analysis can be performed.

Chapter 13

Results

We have been able to identify which data inputs have more number of questions and whether those questions are related to Operational or Non-Operational problems. Although the input pathways are different but our models would be able to perform well on all the three pathways.

- **Question Classifier:** Sentences are passed as input to this classifier and their segregation into Questions and Non-Questions is performed by this classifier. This classifier was developed using the concept of Neural Networks and GloVe vectors and gives an accuracy greater than 90%.
- **Naive Bayes Classifier:** Sentences are passed as input to this classifier to segregate Operational and Non-Operational questions. The accuracy of our Naive Bayes model to perform this segregation is close to 86%.
- **Pipeline:** The complete pipeline was built and tested. Inputs arrive from three different channels are passed through the question classifier and then the Naive Bayes classifier to yield the operational and non-operational queries.

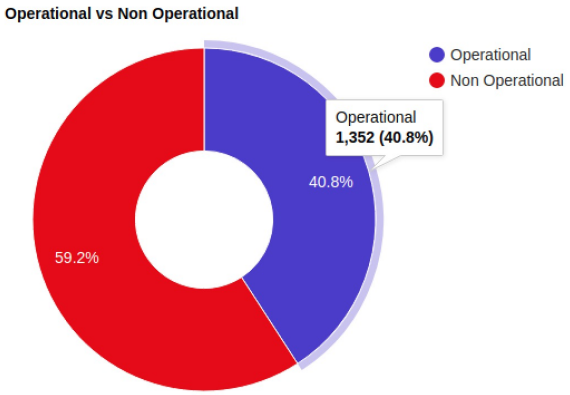


Figure 13.1: Sample Output of Donut Chart

References

[1] Keras Documentation
http://keras.io/

[2] Matplotlib Documentation
https://matplotlib.org/

[3] pandas Documentation
https://pandas.pydata.org/

[4] Scikit Learn Documentation
https://scikit-learn.org/

[5] NLTK Documentation
https://www.nltk.org/

[6] Question Pairs Dataset
https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[7] Positive and Negative Sentences Dataset
https://www.kaggle.com/chaitanyarahalkar/positive-and-negative-sentences

[8] GloVe Vectors
https://nlp.stanford.edu/pubs/glove.pdf

[9] GloVe Model
https://towardsdatascience.com/word-embeddings-for-sentiment-analysis-65f42ea5d2

[10] Baidu Deep Learning Research Paper
https://arxiv.org/abs/1412.5567

[11] Python3.6
https://www.python.org/downloads/release/python-360/

[12] Git - Large File Storage (LFS)
https://git-lfs.github.com/

[13] Deep Speech Repository
https://github.com/mozilla/DeepSpeech

[14] Common Voice Dataset
https://voice.mozilla.org/en/datasets

[15] Google Charts
https://developers.google.com/chart/

[16] Flask
http://flask.pocoo.org/

[17] Gnani-API
https://gnani-ai.github.io/API-service/

[18] CMU-Sphinx
https://cmusphinx.github.io/wiki/tutorial/
https://en.wikipedia.org/wiki/CMU_Sphinx

[19] https://ffmpeg.org/

[20] https://zulko.github.io/moviepy/

[21] https://github.com/jiaaro/pydub

[22] https://machinelearningmastery.com/transfer-learning-for-deep-learning