3/24/20

This is the final report on your implementation. Paste at the beginning as shown below *as is*, except that for the project design+, respond to each comment—within each of their comments. Again, retain the gray parts. Keep in mind the evaluation criteria (at the end) and that they are somewhat altered for this assignment. For voluminous material, reference appendices (at the end). These will be read on an as-needed basis. Excluding appendices and figures, this response should not exceed 6 pages—without prior permission.

REPLACE WITH TITLE OF YOUR PROJECT

by replace with your name

# PROJECT PROPOSAL WITH YOUR RESPONSES

Please insert your Phase 1 here, including comments and your response within each comment.

As usual, please use this Word file template, follow (and retain) the instructions in gray text, and add your work where indicated. The purpose of this template is to assist you in completing a successful project. Keep in mind the evaluation matrix at the end as you do the work and use it to guide what you submit. Use no more than 6 pages of 12-point text excluding figures. You may include as many appendices as you wish for reference. Parts of these may be read as needed.

Your project should involve two technologies, at least one implemented, that approach the same application.

Aviral Srivastava

v 01-27-20

Project Proposal +

This assignment describes your proposed term project (though you may change it going forward). As discussed below, it should involve two ML approaches.

Please use this template (don’t change the headings—we all need them to navigate the growing document throughout the semester) and retain the gray text. Your phase 1 materials—in black 12-point Times New Roman, excluding the gray text, references, and figures—should not exceed 5 pages.

Sentiment Analysis of Demonetisation (and similar events) during the economic crisis in India.

## 1.1 SUMMARY DESCRIPTION

One-paragraph overall description of the purpose, inputs, and outputs for your proposed semester project. Do not go into details because section 1.3 below does this.

Many successful traders analyze sentiment to help them gain an advantage in their trading. Gaining investors’ trust without relying on governing bodies is important for the growth of the economy. India, a country that was once projected as the fastest growing economy by 2020 has fallen behind. Investors are not trusting the government and so, it is important to build up a real-time database for analyzing the current sentiment of the market(public). The target of this project would be demonetization and other such policies implemented recently by the government to help capitalists better understand the current mood. Thus, this project would comprise of a BlackBox that takes input various social media channels(Twitter, Web-based media and journal outlets, etc) and gives the output as follows:

{

entity: <This could be a person like Finance Minister, Infosys Chairman, etc>,

sentiment: <anger, disgust, fear, joy, or sadness>

time\_of\_the\_tweet: <>,

.

.

.

.

}

## 1.2 I/O EXAMPLES

At least two concrete examples of projected output for designated input. You will not be held to the details—it is just explanatory at this point. The examples should be entirely specific, for example, “given this input, A cat sitting on a table

Description automatically generated the output should be *A cat is in the image*” (it would *not* be answering this part to say “the application will recognize cats”).

Input: Tweets like “I love GST implementation #budget2020”; Articles like “Intolerant India” by The Economist

Output: A database comprising of all the entities from different sources (aforementioned). Format, as shown in the figure below where queries can be made on entities (Infosys, Reliance Industries, Red Chillies Entertainment) and result, would be the public sentiment on them: anger, disgust, fear, joy, sadness OR positive/negative.

Refined output: A s/w application that gives you public sentiment on a topic in a given time frame. This is important for many predictions such as elections as shown by Kulshrestha et al [1].

A close up of text on a whiteboard

Description automatically generated

## 1.3 REQUIREMENTS

High-level requirements statements, organized as follows. List your requirements into three approximately even categories using triage (first select the two extreme categories, *definite* and *nice-to-do*, then place the remainder in the middle category.

Requirements are *declarative* statements of the application’s intended functionality such as “the expected number of years to the graduate college shall appear on the console.” (A statement such as “First clean the data” is a procedure step, not a requirement.)

## 1.3.1 Definite (first priority)

 3 to 5 of these

## 1.3.2 Not sure yet (second priority)

3 to 5 of these

## 1.3.3 Nice-to-do (can dispense with if time does not allow; third priority)

3 to 5 of these

Definite (nice to-dos):

i) An API will be made available to query the current (time period definitive) sentiment of entities.

ii) An API will be made to identify entities and emotions (not sentiments) given a source of article/tweet.

Second priority:

i) Include the classification of entities

ii) Include classification of web-resource

ii) Include classification of web-resources: The resources for data which form the public opinion. Such resources are microblogging websites, news articles, etc.

Third priority:

1. Implement this in real-time (for eg, a tweet is tweeted and within 5 seconds(or so), the data is available to be queried.
2. Implement entity-based sentiment analysis
3. Implement real-time geographical sentiment analysis.

## 1.4 HOW SUCCESS WILL BE ASSESSED

Explain, as specifically as possible (quantification is ideal) how the success of the project should be assessed. For example, “the expected number of years to graduate college of 10 BU students, chosen at random, should be correct compared with what actually happened, to within 10%.” You will not necessarily be penalized if you do not attain this measure of success but you learn more if you compare actual accomplishments with a goal.

With Dr. Braude’s and Dr. Laura’s assessment of whether my model can classify, analyze the sentiment, remove fake or bought tweets and articles from analysis w.r.t the real-time tweets.

For eg: a tweet like: “I love PM Modi’s sheer brilliance” is tweeted 10000 times. In reality, if anyone of you would watch this, it’d be identified as fake or something circulated. Similarly, if Dr. Braude and Dr. Laura identify 50 tweets at random as positive/negative and/or happy, joy, sad, etc as their sentiment and emotions, and the same is reported by my software, then that’d be a success. The judges (Dr. Braude and Dr. Laura) can include more people to participate in this human (manual) testing.

**Refined:**

The verification will be done against IBM Watson NLU platform where same tweets will be sent, and the result will be verified with my software application.

“PM Modi unites India during COVID-19” – tweet 1

“PM Modi is a demagogue who just fakes unity, in reality, he is not taking any actionable measures.” – tweet 2

My s/w platform:

Tweet 1: {Topic: COVID-19, Entity: Narendra Modi, Sentiment: For/Positive}

Tweet 2: {Topic: COVID-19, Entity: Narendra Modi, Sentiment: Against/Negative}

IBM Watson NLU:

Tweet 1: {Keyword: COVID-19, Entity: Narendra Modi, Sentiment: Positive}

Tweet 2: {Keyword: COVID-19, Entity: Narendra Modi, Sentiment: Negative}

Both the outputs should match. For rigorous verification, I shall write a script which will match both the results and produce to you my software’s performance.

## 1.5 TECHNOLOGY EXPLANATION

Explain what *two* contrasting machine learning technologies you plan to use--and why you feel they apply. You are permitted to change these as your knowledge deepens. It is ideal if both technologies are implemented; second best if one of them is explained for this project but not implemented.

Sentiment Analysis is one example of Machine Learning and I may combine it with Natural Language Understanding. Sentiment Analysis is applied as sentiment needs to be drawn from the text (tweets and articles). Natural language understanding would be to draw what are the entities used, what are the emotions, and, perhaps what are the sentiments.

**Addition/Elaboration**

First, I am using Language Modeling using pre-trained datasets. I am doing this so as to understand the semantics of the English language. Once my language modeling is done, I proceed to use Model Transfer learning using ULMFiT to tune my data set for tweets and articles.

Thus, I am using language modelling to perform unsupervised pre-training. This follows a supervised learning step where I perform fine-tuning.

## 1.6 BEGINNING DESIGN

Provide the beginning of a design. Phase II will require a complete design but trying it for this phase (I) will allow you to show the notation you will use for your design. Try to use at least one figure, showing where inputs go, where outputs come from, and the ML nature of the elements in between. You can use figures from external sources but (a) acknowledge this here and in the list of references, and (b) tailor the figures to your particular application, don’t just copy it.

A close up of text on a whiteboard

Description automatically generated

## 1.7 DATA SOURCES

Explain whether or not your project requires data. If so, explain where you will obtain it (e.g., Kaggle). Be careful not to underestimate the effort required to collect large amounts of data yourself.

i. Twitter Public API

ii. Online media publications and blogs such as The Economist, Scoopwhoop, etc.

## 1.8 SCHEDULE

Rough schedule of design and implementation. (Maybe adjusted when subsequent assignments are posted)

February 25 - Complete 1 POC that analyzes tweets and presents the output in the aforementioned format.

March 3 - Complete end-to-end integration of Twitter -> Query APIs

March 10 - Integrate articles and figure out storage of such large texts

March 17 - Complete end-to-end integration of blogs and articles with public query APIs.

March 24 - Finish off the whole BlackBox with public repository available

March 31 - Discuss with Dr. Eric and Laura about the possibilities of research paper (at least a case study)

## 1.9 REFERENCES FOR PROPOSAL PHASE

Fill in below—and cite each of the following (e.g., “[2]“) within the text. References can include specific places in the notes and textbook. “Use of references” is a separate evaluation criterion—see below. Building on the work of others is encouraged as long as (1) you show that you understand it, and (2) you acknowledge it clearly (e.g., not simply list it among the references).

[1] Kulshrestha, Shah, Lu, “Politically Predictive Potential of Social Networks: Twitter and the Indian General Election 2014”

[2]

[3] …

## 1.7 Evaluation



# PROJECT DESIGN +

Please insert your Phase 2 here, as originally posted but including our comments. Please respond to every comment within each comment using the reply feature.

# Assignment 2 02/19/20

## 2.1Final Requirements

List your final requirements, numbering them in the form DiX and NiX where:

D/N means “Definite” / “Nice to do” (two categories now, not three)

i = 1, 2, 3, …

X=F when the goal is functional (a conventional requirement) – or – X=L and the goal is a learning goal of yours

For example:

D2F **Wine Suggestion**: The application shall suggest a wine with the dinner. (This is the second definite requirement.)

N4L **TPU Porting**: The application will be ported to Google’s TPU Console to assess the obstacles in doing so, and to measure the difference in execution compared with my GPU-enabled laptop. (This is the fourth nice-to-do requirement.)

There should be 4-8 items in the “Definite” list and at least 4 in the “Nice to do” list. You will reference these numbered requirements in phase 3 when you will be asked to show what the project accomplished.

### D1F Public Sentiment on a Topic

Given a topic and time frame, s/w will o/p the sentiment on that topic within that time frame.

Example:

Input: {Topic: Making COVID-19 testing kits, Time: 5h}

Output: {Favor}

If there were x tweets against making COVID-19 testing kits, and y for making more kits, and if y>x, the output will show “Favor”.

### D2F Enlist entities from a tweet(s)

Given a real time streaming of tweet(s), s/w will determine the entities mentioned in the tweet

In steps:

1. a stream of tweets is fed to the s/w application,
2. s/w application will digest them all to give results when processing over these tweets is completed, which won’t be instant.
3. s/w will show the entities involved in those tweets right away.

### D3F Sentiment on a Tweet

Given a tweet, s/w will output 2 results: topic that tweet is talking about AND corresponding sentiment.

Input: {“Delhi CM gave three months free ration to daily wage laborers amidst COVID-19 crisis.”}

Output: {Topics: Free ration amidst COVID-19 crisis, Sentiment: Favor}

The topic is identified by 24\*7 batch processing by my s/w which keeps on digesting tweets on the topics mentioned so far in order to identify them quickly and produce valuable outputs.

### D4F Public sentiment for Entities

Given a list of entities, s/w will output the overall sentiment pertaining to those entities in a default but configurable time frame.

Input: {“Entities: [Kejriwal, Modi]}

Output: {Kejriwal: For, Modi: Favor}

### D1L Learning accuracy with less data.

Learn to produce good results (result quality verified against IBM Watson NLU) with less data [1]. [Using ULMFiT and Fast AI.]

### D2L Learning streaming techniques in ML

Learn to apply gradual freezing: in order to save weights from getting destroyed so that they can be used further out and reused.[2]

### D3L Analyzing different optimizers to improve learning rate.

Learn to use different optimizers among Adam, Adagrad, RMSProp to optimize learning so that I can reach the stable point with less computation and make sure I achieve the best possible accuracy.

### N1F Pub-Sub model for live sentiment analysis

Given an API, create a publisher-subscriber model [3] for enabling end-users gather public sentiment in a “live” manner, real-time.

### N2F Entity sentiment analysis

Given a topic and time frame, the s/w should be able to enlist all the entities involved in those topics and highlight the sentiment corresponding to those entities.

### N3F Trends in a geographical area

Given an area and a trend (for/against), the s/w will output the tweets and topics counted in that trend.

### N3F Identify fake tweets

There are cyber armies of political parties [] and so, it is of utmost important to identify fake tweets and fake articles somehow in order to produce better results.

### N1L To perform sentiment analysis as fast as IBM Watson NLU

IBM Watson NLU is amongst the fastest NLU platform [4] with its ever-growing database. I intend to build a platform which might not be as diverse (as I am aiming only for economic topics or political topics in general) but at least as fast in yielding outputs as Watson is.

### N2L Pub-Sub Machine Learning models

Almost all ML models are about trial and error [5]. I intend to learn about machine learning models suitable for pub-sub applications.

### N3L Case study of Transfer-flow machine learning models in real-time analysis

Transfer-flow models are new in the world of ML, not much have been experimented when it comes to such models in real-time data applications, as per my research and observations. I intend to not only use them but also to draw out certain observations when it comes to real-time data – on the data velocity, variety and computation capacity.

## 2.2 Design and Theory Describe the design of your proposed system. Use annotated diagrams. Explain the ML technologies and theory behind your design. This is particularly true of technology not specifically covered in the notes. Explain how the two technologies will interface or compare. The reader should understand how the pieces are going to fit together. Show this at a high level, as well as providing as much relevant detail as you can. Include at least one (meaningful) figure, as in the figure, for example (you may edit it for your purposes if you like).

**End-to-end picture**: The s/w will scrape Twitter and web articles (both in batches and in streams) using Tweepy [6] (Twitter client in Python) and will use a pub-sub model to stream the output data (sentiment and entities). The pub-sub feature makes it available as an on-demand tool for the end user. The s/w will continuously train itself with tweets and articles in batches and will always be available to be queried for the data.

**Deep diving into ML design**: Overall, this project is broadly a classification problem [7] as we are taking public sentiments (tweets, articles, etc.) on certain topics (economy related) and classifying them (positive/negative OR against/supportive). The Twitter data set is considered as a noisy dataset [8], and it being an untagged data gives us a new challenge: performing classification on untagged data. Due to this, we have to use transfer learning model as it is suitable for less tagged data [9]. Transfer model first learns the language for which Wiki data set is used (which is a huge data set). So, Wiki Text 103 is used to train our model and make it (the model) learn semantics of the English language.

Steps:

1. Pre-training: Build a language model by training on Wiki-text 103.
2. Target Language Model: Feed our model with tweets so that it learns Twitter data set, learn the semantics of it. We do this step so as to make our model customized to our needs (Twitter data) instead of general English language.
3. Target Classifier: After making our model learn the semantics in step 2, we will use classifier model to make it learn the classification. We will use the Twitter data set along with IBM Watson NLU’s sentiment analysis to train our model. This is a significant step as IBM Watson NLU will be used to verify the results of our approach. IBM Watson’s NLU has been heavily adapted by YC funded startups [10] so it can be treated as a benchmark.
4. Improving accuracy of Target LM.
   1. **Freezing**: Freeze all the layers so that trained weights do not get lost. I will use GRU as it is less computationally intensive than LSTM [11] otherwise will switch to LSTM. Both of them provide with the freezing abilities.
   2. **Scheduling Learning Rate**: We will schedule the learning rate to reach the optimized result so that we can get the best stable state of our learning *i.e.* accuracy.
   3. **Discriminative Fine Tuning**: We will optimize learning rate in different layers within an iteration so that we can reach optimization across them in a smaller number of epochs.
5. Improving accuracy of Target Classifier:
   1. **Concat Pooling:** We cannot only consider input(data) from the last hidden state as the input set consists of thousands of words and a lot of information might get lost. Thus, any hidden state *h* at a time *t* will be concatenated with the maximum and average pool of the hidden states currently in the memory (GPU) as suggested in [12].
   2. **Linear Decoder:** ReLu and Softmax (shown below) together form a linear decoder to classify whether the input is in favor or against.
   3. **Gradual Unfreezing:** We want to unfreeze when we want to make our model to train on our target.

**Variable Architecture**

The variables in architecture are as follows:

1. N: number of tokens
2. D: Embedding Dimension
3. H: Hidden activations per layer

A screenshot of a cell phone

Description automatically generated

## 2.3 Tools

Describe the tool(s) you will use, or explain why you will build from scratch. Explain your choice. Show that you understand how the tools work.

Fast.ai module

Pytorch

Keras

Python

Kafka (pub-sub usage)

GPU for training language model (if possible, not used yet)

## 2.4 Implementation Fragments

Show enough *parts* of an implementation—or a simplified form of it—to convince the reader (and yourself) that you will have the implementation of the definite requirements completed on time. These can be experimental or exploratory in nature but the purpose is to have something up and running at all times. Your choices can coordinate with section 2.4 below. Cut and paste at most 1 1/2 pages of commented code below. Explain what part of the application they each relate to.

In section 2.2, from the subsection “Talking ML”, I am presenting the snippets of the first two steps (until “Target Language Model”). This section is not yet customized for the software that I am developing but it assures me that I will be able to use the language training model [13].

This is the ML model being implemented: <https://drive.google.com/open?id=1E33PD-57HC6L2BeHMNubkY4phyPz-VHP>.

Following code snippets are for extracting data from Twitter (ignore the errors), taken reliably from Tweepy examples:

import re

import tweepy

from tweepy import OAuthHandler

from textblob import TextBlob

class TwitterClient(object):

'''

Generic Twitter Class for sentiment analysis.

'''

def clean\_tweet(self, tweet):

'''

Utility function to clean tweet text by removing links, special characters

using simple regex statements.

'''

return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])

|(\w+:\/\/\S+)", " ", tweet).split())

def get\_tweets(self, query, count = 10):

'''

Main function to fetch tweets and parse them.

'''

# empty list to store parsed tweets

tweets = []

try:

# call twitter api to fetch tweets

fetched\_tweets = self.api.search(q = query, count = count)

# parsing tweets one by one

for tweet in fetched\_tweets:

# empty dictionary to store required params of a tweet

parsed\_tweet = {}

# saving text of tweet

parsed\_tweet['text'] = tweet.text

# saving sentiment of tweet

parsed\_tweet['sentiment'] = self.get\_tweet\_sentiment(tweet.text)

# appending parsed tweet to tweets list

if tweet.retweet\_count > 0:

# if tweet has retweets, ensure that it is appended only once

if parsed\_tweet not in tweets:

tweets.append(parsed\_tweet)

else:

tweets.append(parsed\_tweet)

# return parsed tweets

return tweets

except tweepy.TweepError as e:

# print error (if any)

print("Error : " + str(e))

Following code snippet is for streaming Tweets:

# -\*- coding: utf-8 -\*-

import logging

import re

import configparser

from kafka import KafkaProducer

from kafka.errors import KafkaError

import tweepy

# Logging Config

logger = logging.getLogger('streaming')

logger.setLevel(logging.DEBUG)

fh = logging.FileHandler('streaming.log')

fh.setLevel(logging.DEBUG)

ch = logging.StreamHandler()

ch.setLevel(logging.DEBUG)

formatter = logging.Formatter(

'%(asctime)s - %(name)s - %(levelname)s - %(message)s')

fh.setFormatter(formatter)

ch.setFormatter(formatter)

logger.addHandler(fh)

logger.addHandler(ch)

####

def check\_for\_http\_and\_https(text):

httpsCheck = 'https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+'

httpCheck = 'http?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+'

if(re.findall(httpCheck, text) or re.findall(httpsCheck, text)):

return True

else:

return False

def convert\_hashtags\_dict\_to\_list(hashtag\_dict):

hashtag\_list = []

for key in hashtag\_dict:

hashtag\_list.append(hashtag\_dict[key])

return hashtag\_list

class StreamListener(tweepy.StreamListener):

'''Streaming Class'''

def \_\_init\_\_(self, api, kafka\_topic):

self.api = api

self.kafka\_topic = kafka\_topic

self.producer = KafkaProducer(

bootstrap\_servers=['localhost:9092'], retries=5)

self.tweet = {}

self.idSelf = 0

def on\_status(self, status):

if status.retweeted:

return

tweetText = status.text.encode('utf8')

if (check\_for\_http\_and\_https(tweetText)):

return

try:

self.idSelf += 1

self.tweet["userProfileImageURL"] = status.user.\_json[

"profile\_image\_url\_https"]

self.tweet["tweet"] = tweetText

self.tweet["id"] = status.id

self.tweet["sequence"] = self.idSelf

self.tweet["timestamp"] = status.created\_at

logger.info("Tweet sequence: %d", self.idSelf)

self.producer.send(self.kafka\_topic, bytes(

str(self.tweet).encode('utf8')))

except Exception as e:

logging.info("Error inside status")

logging.error(e)

finally:

self.producer.flush()

def on\_error(self, status\_code):

logging.error(status\_code)

return True

if \_\_name\_\_ == '\_\_main\_\_':

# Setup logging

config = configparser.ConfigParser()

config.read('config.txt')

# Read Twitter Credentials

consumer\_key = config['TWITTER']['consumerKey']

consumer\_secret = config['TWITTER']['consumerSecret']

access\_key = config['TWITTER']['accessToken']

access\_secret = config['TWITTER']['accessTokenSecret']

# Auth Object

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_key, access\_secret)

api = tweepy.API(auth)

logger.info("Twitter API Auth successful")

# Streaming Config

hash\_tags\_dict = dict(config['HASHTAGS'].items())

collection = config['FIXTURE']['collection']

hash\_tags\_to\_track = convert\_hashtags\_dict\_to\_list(hash\_tags\_dict)

logger.info("Tracking %s", hash\_tags\_to\_track)

# Finally Streaming

while True:

try:

stream = tweepy.Stream(

auth=api.auth, listener=StreamListener(api=api, kafka\_topic=collection))

logger.info("Connection Made")

stream.filter(languages=["en"], track=hash\_tags\_to\_track)

except Exception as e:

logger.error(e)

Remaining:

1. Target Classifier
2. Improving accuracy of Target LM
3. Improving accuracy of Target Classifier

2.5 Risk Retirement  
Identify and prioritize the 4 top risks in carrying out the project. Try as best you can to retire the top three by the time you submit this, by means of experiments, prototypes, or work-arounds. Explain how you did this. Explain how you will retire the remaining risks in advance. (P.S. Consult with Eric or the course TA if and when you need assistance!)

1. Insufficient entities in IBM Watson NLU

There was a risk of entities going unidentified in tweets as none of the systems state they know certain entities and don’t know certain others. This **risk is resolved** as I experimented with more than 10k tweets and **got entities in all of them** using Watson.

1. Insufficient data to train model for outputting public sentiment

There was risk of not having enough data. Now, users will mostly query data to get the current scenario as the mood 9 months ago can easily be known. I learned about LSTM in lecture and proceeded with it to end up using ULTMFiTs with LSTM. So, less data is not an issue as I learned training my model the semantics by Wiki data. This **risk is resolved.**

1. Target Classifier  
   I learned how to train the semantics but was unsure of how to classify my data and use it to analyze sentiments of incoming tweets. As every new tweet is a challenge for my s/w to solve but also a new data point for later tweets to analyze. I have retired this risk by introducing IBM Watson NLU’s data as the benchmark and so, as the training data for my model. This **risk is resolved.**
2. Improving accuracy of Target Language Model: **done at Proof of concept level, so, not completely resolved.**
3. Improving accuracy for Target Classifier: **done at Proof of concept level, so, not completely resolved.**

## 2.6 Schedule

Explain in outline the steps you intend to take to carry out the project. Show the completion of the stages. Include a schedule, as detailed as can be reasonably foreseen.

March 24 – complete Proof of Concept (90% done)

March 31 – complete classification and de-risk unknowns

April 7 – produce results and write a case study

April 14 – discuss the possibilities of case study/research paper with Dr Eric and Laura.

## 2.7 References

## List the references to materials of all kinds that you used in developing this, including the notes, the Web and any books. Each should occur in the body of the text at the appropriate place. Example (within the preceding sections): Jones [5] recommends running the genetic algorithm for at least 10 generations. Note that there is a separate evaluation criteria for references—25% of your grade—because we want you to gather information widely.

[1] Jeremy Howard and Sebastian Ruder, “Introducing state of the art text classification with universal language models”

[2] Howard and Ruder, “Universal Language Model Fine-tuning for Text Classification”

[3] <https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern>

[4] <https://researcher.watson.ibm.com/researcher/view_group_pubs.php?grp=147>

[5] Dr Eric in ML lectures, MC.AI(<https://mc.ai/learning-through-trial-and-error/>)

[6] <http://docs.tweepy.org/en/v3.8.0/>

[7] Sebastian Ruder, “Semi Supervised Learning…text classification”. (<https://ruder.io/semi-supervised/>)

[8] Joshi, *et al,* “Twitter Sentiment Analysis System”, International Journal of Computer Applications, 2018

[9] Ding et al. “Empirical Study and Improvement on Deep Transfer Learning for Human Activity Recognition”, (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6339185/>, “target uses less unlabeled data”)

[10] “YC funded startup Trenity uses IBM Watson NLU”, available here(<https://developer.ibm.com/blogs/trenity-uses-watson-nlu-to-build-a-sports-engagement-platform/>)

[11] Anand et al. “Approximate Computing for Long Short Term Memory (LSTM) Neural Networks”, IEEE.

[12] ULMFiT RNN Classifier (<https://forums.fast.ai/t/ulmfit-get-rnn-classifier-concat-pooling/25102>)

[13] SemEval Stance Task Classification (<http://alt.qcri.org/semeval2016/task6/>)

[14] India Misinformation Election, The Atlantic (<https://www.theatlantic.com/international/archive/2019/04/india-misinformation-election-fake-news/586123/>)

## 2.8 Evaluation of Design Phase

Keep in mind that good and excellent responses require more than satisfying a minimum.



# IMPLEMENTATION

Excluding figures and appendix, this should be at most 5 pages of 12-point type.

## 3.1 Summary

In a paragraph or two, summarize the outcome of your project (a) functionally and (b) learning-wise but avoid duplication with Section 3.2 below. (Reminder: observe leave these gray sections in your paper.)

We have built the software named [Assange](https://github.com/kebab-mai-haddi/assange). Assange delivers you sentiment analysis on any topic that is being discussed by people and lets you decide independently what the public perception is. You can use Assange to make data repositories on different topics or you can use the existing topics from the source code repository.

While building Assange, I learned creating data pipelines specifically for Machine Learning tasks. Besides engineering, I learned Machine Learning techniques such as using transfer learning models, learning to improve accuracy by freezing and most importantly, determining whether to build something myself or use something off-the-shelf.

## 3.2 Report on Requirements

Describe the extent to which you accomplished each definite requirement "DiX" (X = F or L) as well as any other fulfilled requirements. For each, include 1-3 sentences regarding degree of accomplishment, and screenshot of input and output that supports this. Your effectiveness depends a lot on how much you demonstrate that you learned. Limit: 2 pages of 12-point Times New Roman.

### D1F Public Sentiment on a Topic

Given a topic, Assange should output the public sentiment (positive, negative, or neutral),

This requirement is completed.

A picture containing food, drawing

Description automatically generated

### D2F Enlist entities from a tweet(s) on a topic

Given a topic, enlist all the entities (public figures) from that topic

This requirement is completed.

Note: Initially, the though was to just list entities and list sentiment for those entities in a topic. I have merged these two requirements (D2F and D4F) as one. Now, users will be able to get all the entities mentioned in a topic along with their sentiment.

A screenshot of a cell phone

Description automatically generated

### D3F Sentiment on a Tweet

Given a topic, enlist sentiment on each tweet.

This requirement is completed. Now, the user will get a csv file for a topic that will have tweets and their corresponding sentiment.

A close up of a logo

Description automatically generated

A close up of a piece of paper

Description automatically generated

### D1L Learning accuracy with less data

Learn to produce good results (result quality verified against IBM Watson NLU) with less data [1]. [Using ULMFiT and Fast AI.]

I have been able to achieve accuracy > 50% with less than 5k tweets on a topic. I’d state that this requirement has been completed.

A screenshot of a cell phone

Description automatically generated

### D2L Learning streaming techniques in ML

Learn to apply gradual freezing: in order to save weights from getting destroyed so that they can be used further out and reused.[2]

This requirement is completed

Proof: [A three minute video](https://www.loom.com/share/86d86faeb9484257bc62732ec47fe864) demonstrating streaming machine learning analysis.

Video is included as I failed to capture streaming in screenshots.

Complete video: [This video](https://www.loom.com/share/634dae64f46f40698b07385a16d5d33f)

### D3L Analyzing different optimizers to improve learning rate.

Learn to use different optimizers among Adam, Adagrad, RMSProp to optimize learning so that I can reach the stable point with less computation and make sure I achieve the best possible accuracy.  
This requirement is completed and I ended up using Adam, ASGD[15] for optimizing learning curve. For 1k tweets (1k rows in a csv file), there are two epochs used with an average accuracy of more than 50%. In the image below, notice the two *epochs* and 66% accuracy.  
A screenshot of a cell phone

Description automatically generated

### N1F Source code build up from scratch

The source code for Assange should be i)open-source ii)build is easy such that anyone can build it up from scratch

Completed.

A screenshot of a cell phone

Description automatically generated

Complete proof available here: <https://github.com/kebab-mai-haddi/assange>

### N2F Decouple Learning-Streaming-Analysis

I wanted Assange to be trained on one-topic only *i.e.* training on one topic should not be influenced by another topics’ training. Assange is decoupled in topic-wise analysis. Users could train it on a topic like “COVID-19” alone in order to achieve more accuracy. Users need not to feed tweets other than COVID-19 in order to train Assange. This requirement is useful in topics which are scarcely available and high accuracy (90%) is desired.  
Completed.  
No screenshot could demonstrate this so [video](https://www.loom.com/share/634dae64f46f40698b07385a16d5d33f) is needed:

### N1L Pub-Sub Machine Learning models

Almost all ML models are about trial and error [5]. I intend to learn about machine learning models suitable for pub-sub applications.

Completed.

The [video](https://www.loom.com/share/634dae64f46f40698b07385a16d5d33f) demonstrates how this is a pub-sub model.

## 3.3 Report on Design

Describe the design that you finally used. Indicate how, where, and why it differed from your planned design. Describe its advantages and its shortcomings. Include a description of how the technologies you explored (not the tools—those are described below) leveraged each other. Include at least one diagram. Limit: 2 pages of 12-point Times New Roman.

**End-to-end picture**: The s/w will scrape Twitter in streams using Tweepy [6] and will use a pub-sub model to stream the output data (sentiment and entities). The pub-sub feature makes it available as an on-demand tool for the end user. The s/w will train itself on user’s command and will always be available to be queried for the data.

**Deep diving into ML design**: Overall, this project is broadly a classification problem [7] as we are taking public sentiments (tweets, articles, etc.) on certain topics (economy related) and classifying them (positive/negative/neutral). The Twitter data set is considered as a noisy dataset [8], and it being an untagged data gave us a new challenge: performing classification on untagged data. Due to this, we used transfer learning model as it is suitable for less tagged data [9]. Transfer model first learns the language for which Wiki data set is used (which is a huge data set). So, Wiki Text 103 is used to train our model and make it (the model) learn semantics of the English language. After which we stream data from IBM Watson NLU tool in order to perform training. We decided to let go of content that does not contribute to any sentiment analysis (such as hyperlinks) or performing analysis on them was a whole new project (such as images).

Steps:

Pre-training: Build a language model by training on Wiki-text 103.

Target Language Model: Feed our model with tweets so that it learns Twitter data set, learn the semantics of it. We do this step so as to make our model customized to our needs (Twitter data) instead of general English language.

Target Classifier: After making our model learn the semantics in step 2, we will use classifier model to make it learn the classification. We will use the Twitter data set along with IBM Watson NLU’s sentiment analysis to train our model. This is a significant step as IBM Watson NLU will be used to verify the results of our approach. IBM Watson’s NLU has been heavily adapted by YC funded startups [10] so it can be treated as a benchmark.

Improving accuracy of Target LM.

**Freezing**: Freeze all the layers so that trained weights do not get lost. I will use GRU as it is less computationally intensive than LSTM [11] otherwise will switch to LSTM. Both of them provide with the freezing abilities.

**Scheduling Learning Rate**: We will schedule the learning rate (by increasing the proportion of training data) to reach the optimized result so that we can get the best stable state of our learning *i.e.* accuracy. This is also configurable by the end user.

**Discriminative Fine Tuning**: We will optimize learning rate in different layers within an iteration so that we can reach optimization across them in a smaller number of epochs.

Improving accuracy of Target Classifier: [All this was done under the hood by *learner* module of Fast-AI]

**Concat Pooling:** We cannot only consider input(data) from the last hidden state as the input set consists of thousands of words and a lot of information might get lost. Thus, any hidden state *h* at a time *t* will be concatenated with the maximum and average pool of the hidden states currently in the memory (GPU) as suggested in [12].

**Linear Decoder:** ReLu and Softmax (shown below) together form a linear decoder to classify whether the input is in favor or against.

**Gradual Unfreezing:** We want to unfreeze when we want to make our model to train on our target.

**Variable Architecture**

The variables in architecture are as follows:

N: number of tokens

D: Embedding Dimension

H: Hidden activations per layer

A screenshot of a cell phone

Description automatically generated

**Differences from the planned design:**

We decided to include articles as well in our analysis but later realized that it falls outside our scope. Assange is built to plot public sentiment analysis and not the analysis of journalists or blog writers. So, scraping from articles and blogs have been removed from our design.

We also decided to stop batch-processing of tweets and instead opted only for streaming as it is provided easily by the Twitter’s APIs and batch-processing didn’t add any value to our analysis objective.

**Advantages of this design:**

**Highly configurable and customizable**: Assange is highly configurable as it provides freedom to opt for topics, learning rate (by limiting the input data). It is idempotent such that if streaming tweets is broken, source data is not lost. This is important if performing some customization over Assange.

**Fast:** For a dataset of 1k, it takes only a couple of minutes to learn everything from the start and start producing results.

**Extendible:** Assange is highly granular (users can drill down to topics, tweets, entities mentioned in each tweet). This opens up the possibility of building something using Assange,

**Lightweight:** Assange does not need a heavy processing unit. We used 2 CPU cores for all the work, and it was smooth.

**Shortcomings of this design:**

1. **Unpredictable training dataset**
   1. As Assange supports any range of topics, the training data set is unpredictable in length and so the risk of overfitting and underfitting exists.
2. **Undefined Accuracy Range**
   1. There are some topics where training data might contain way too many neutral (or positive or negative ) sentiment tweets than positive and negative ones and in such cases, Assange is treated more for neutral than the rest two.

## 3.4 Tools

Describe the machine learning tool(s) that you used. Show samples. Describe their advantages and their shortcomings. (List other tools separately.) Limit: 1 page of 12-point Times New Roman.

**Fast-ai**: for simplifying training and increasing accuracy of neural nets. Quick, reliable, useful for less tagged data. However, it breaks too often with new updates as it is a bleeding-edge library.

**NLTK**: for working with human language data, used specially for its parsing, tokenization, and classification of English language. It provides an extensive functionality of APIs but is not much configurable. It takes time that agile development is hindered.

**Corpus**: for dealing with *stopwords* such as ‘a’, ‘an’, ‘the’. It draws the semantics precisely but sometimes, Tweets present serious attention to these stop words which are ignored by Corpus. There should have been some configuration provided for us like, not to ignore articles in capital letters (‘THE’) such that we can draw emphasis and complete sentences.

**SKLearn**: for classification of tweets after determining their language semantics. It is highly customizable as many options are provided for testing and training the dataset. I could not find any drawback of using SKLearn specifically. It is a very mature library.

## 3.5 Contrast between approaches

You were to include two approaches to your problem, and implement at least one. Contrast the two approaches as they specifically relate to your project.

The two approaches mentioned at the beginning were: i) Sentiment Analysis ii) Natural Language Understanding.

Sentiment Analysis was intended to be used with the purpose of categorizing tweets, classifying words and then stating the sentiment. I thought that this would not include Natural Language Understanding and just the labelled training data would suffice.

As I expanded my project, it became the other way around - I had to include more NLU and then perform sentiment analysis using the labelled data.

On the whole, I used both the technologies but the NLU became more important and sentiment analysis (after performing NLU) became a small but significant part.

## 3.6 What did *not* work well

Explain the most important aspects of your project that fell short of your plans or desires.

**The speed could not be as fast as IBM Watson.**

I wanted Assange to be faster than Watson, but it fell short. I realized that in order to be as fast as Watson, I would not only have to be efficient in ML model but also invest time and capital in compute resources.

**Build a streaming model**

I started with the aim of delivering the public sentiment analysis for a topic and a little part of it was to provide sentiment analysis on tweets that have topics on which Assange was never trained for. I won’t say I fell short of it but I de-railed from this idea as I realized that public sentiment is not analyzed over one real time tweet, but on a certain significant number of tweets.

**Identifying Fake Tweets**

It was my desire to identify fake tweets (repetitive tweets from cyber armies of political parties) but I could not do that as it turned out to be a whole different project and an active research area – something that even Twitter and Facebook are struggling for [16].

## 3.7 What *did* work well

In paragraph form, explain the most important aspects of your project that met or exceeded your plans or desires.

Assange exceeded my expectations in speed and accuracy. I did aim for a high accuracy but had no expectations given the project and my Machine Learning experience. More than these two, I really liked how Assange turned out to be a less intense in computing.

## 3.8 Presentation Materials

Prepare a presentation for the last class based on the above. Include a short demonstration. This should last no more than 15 minutes. Include in this report copies of the slides, two per page.

Demo: <https://www.loom.com/share/634dae64f46f40698b07385a16d5d33f>

Skip to 5:00 in order to start right at the demo part, before that time, there is an explanation of the tool.

Presentation: <https://docs.google.com/presentation/d/1Kf1m_bh6BBOVzE6Rke_Sb74Zm3gg8wLNo9j62W6yMCo/edit?usp=sharing>

## 3.9 Sample Source

Supply up to 1 page of key excerpts from your source code—or what comes closest to “source code.” Limit: 2 pages of 12-point Times New Roman. Include an explanation of where the excerpts fit in your implementation.

The key-excerpt for this project would be the machine learning model used and implemented. Rest all are significant but non-ML or less-ML lines of code.

ML:

import configparser

import io

import os

from functools import partial

from hashlib import sha1

import fastai

import nltk

import numpy as np

import pandas as pd

from fastai import \*

from fastai.text import \*

from nltk.corpus import stopwords

from pymongo import MongoClient

from sklearn.datasets import fetch\_20newsgroups

from sklearn.model\_selection import train\_test\_split

file\_path = input('Please enter path of the file.\n')

df = pd.read\_csv(

filepath\_or\_buffer=file\_path,

header=None

)

# CATEGORISING THE SENTIMENT TABLE

# NEUTRAL -> 0 | NEGATIVE -> 1 | POSITIVE -> 2

df[1] = df[1].astype('category')

df[1] = df[1].cat.codes

# NUMBER OF SAMPLES FOR EACH CATEGORY

print(df[1].value\_counts())

#df = pd.DataFrame({'label':dataset.target, 'text':dataset.data})

df = pd.DataFrame({'label': df[1], 'text': df[0]})

print(df.shape)

df['text'] = df['text'].str.replace("[^a-zA-Z]", " ")

print(df.head(5))

nltk.download('stopwords')

stop\_words = stopwords.words('english')

# tokenization

tokenized\_doc = df['text'].apply(lambda x: x.split())

# remove stop-words

tokenized\_doc = tokenized\_doc.apply(

lambda x: [item for item in x if item not in stop\_words])

# de-tokenization

detokenized\_doc = []

for i in range(len(df)):

t = ' '.join(tokenized\_doc[i])

detokenized\_doc.append(t)

df['text'] = detokenized\_doc

df\_trn, df\_val = train\_test\_split(

df, stratify=df['label'], test\_size=0.15, random\_state=12)

print(df\_trn.shape, df\_val.shape)

# Language model data

data\_lm = TextLMDataBunch.from\_df(train\_df=df\_trn, valid\_df=df\_val, path="")

# Classifier model data

data\_clas = TextClasDataBunch.from\_df(

path="", train\_df=df\_trn, valid\_df=df\_val, vocab=data\_lm.train\_ds.vocab, bs=32)

learn = language\_model\_learner(data\_lm, arch=AWD\_LSTM, drop\_mult=0.7)

# train the learner object with learning rate = 1e-2

learn.fit\_one\_cycle(1, 1e-2)

learn.save\_encoder('ft\_enc')

learn = text\_classifier\_learner(data\_clas, arch=AWD\_LSTM, drop\_mult=0.7)

learn.load\_encoder('ft\_enc')

learn.fit\_one\_cycle(1, 1e-2)

# get predictions

preds, targets = learn.get\_preds()

predictions = np.argmax(preds, axis=1)

pd.crosstab(predictions, targets)

config = configparser.ConfigParser()

config.read('config.txt')

hash\_tags\_dict = dict(config['HASHTAGS'].items())

for key in hash\_tags\_dict.keys():

source\_collection = hash\_tags\_dict[key]

client = MongoClient()

db = client.phase3

file\_path = os.path.basename(file\_path)

collection\_name = '{}'.format(file\_path.replace('.', '\_'))

for i in range(1, len(df\_val)+1):

tweet = df\_val[i-1:i]['text'].array[0]

neutral = preds[i-1:i][0][0]

negative = preds[i-1:i][0][1]

positive = preds[i-1:i][0][2]

if neutral > positive and neutral > negative:

sentiment = 'neutral'

elif negative > neutral and negative > positive:

sentiment = 'negative'

else:

sentiment = 'positive'

db[collection\_name].insert\_one(

{

"tweet": tweet,

"sentiment": sentiment

}

)

if i % 100 == 0:

print(i)

Rest of the source code can be found at: <https://github.com/kebab-mai-haddi/assange>

## 3.10 Source

Attach source code (or what comes closest to it) and input where possible. You may refer the reader to github for source if you prefer.

<https://github.com/kebab-mai-haddi/assange>

## References

Show that you used a wide variety of resources by listing them below and clearly indicating in the body above where you used them.

[1] Jeremy Howard and Sebastian Ruder, “Introducing state of the art text classification with universal language models”

[2] Howard and Ruder, “Universal Language Model Fine-tuning for Text Classification”

[3] <https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern>

[4] <https://researcher.watson.ibm.com/researcher/view_group_pubs.php?grp=147>

[5] Dr Eric in ML lectures, MC.AI(<https://mc.ai/learning-through-trial-and-error/>)

[6] <http://docs.tweepy.org/en/v3.8.0/>

[7] Sebastian Ruder, “Semi Supervised Learning…text classification”. (<https://ruder.io/semi-supervised/>)

[8] Joshi, *et al,* “Twitter Sentiment Analysis System”, International Journal of Computer Applications, 2018

[9] Ding et al. “Empirical Study and Improvement on Deep Transfer Learning for Human Activity Recognition”, (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6339185/>, “target uses less unlabeled data”)

[10] “YC funded startup Trenity uses IBM Watson NLU”, available here(<https://developer.ibm.com/blogs/trenity-uses-watson-nlu-to-build-a-sports-engagement-platform/>)

[11] Anand et al. “Approximate Computing for Long Short Term Memory (LSTM) Neural Networks”, IEEE.

[12] ULMFiT RNN Classifier (<https://forums.fast.ai/t/ulmfit-get-rnn-classifier-concat-pooling/25102>)

[13] SemEval Stance Task Classification (<http://alt.qcri.org/semeval2016/task6/>)

[14] India Misinformation Election, The Atlantic (<https://www.theatlantic.com/international/archive/2019/04/india-misinformation-election-fake-news/586123/>)

[15] “Regularizing and Optimizing LSTM Language Models” by *Stephen Merity, Nitish Shirish Keskar, Richard Socher*

[16] <https://www.wired.com/story/twitter-abusive-apps-machine-learning/>, <https://www.facebook.com/facebookmedia/blog/working-to-stop-misinformation-and-false-news>

# Appendices

If necessary, supply one or more appendices with material that you want to make available. Appendices will be read on an as-needed basis only. They should be referred to in the body of the paper.

## Appendix 1: title

your content here

## Appendix 2: title

your content here

## Appendix 3: title

your content here

# Evaluation

