**PROJECT REPORT ON**

***“News Portal based on news analysis and categorization”***

**Submitted**

as part of **ITE1008 Open Source Programming**

**by**

**Shivam kumar Verma**

**16BCB0015**

To



**School of Computer Science and Engineering**



**March 2019**

**Index**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Topic** | **Page No.** |
| **1** | Importance of idea & problem statement |  |
| **2** | Objectives |  |
| **3** | Software Requirements |  |
| **4** | Architecture |  |
| **5** | Development Details |  |
| **6** | Detailed Design |  |
| **7** | Implementation |  |
| **7.1.** | Pseudo Code |  |
| **8.** | Applicability category |  |
| **9.** | Conclusions |  |
| **10.** | References |  |

**Chapter 1**

**IMPORTANCE OF IDEA & PROBLEM STATEMENT**

**IMPORTANCE OF IDEA:**

* A Clustering based relavant news provider is a Idea where user can get relavant and Important News.
* The system allows public a fast and cheap way to get updated with the world.
* This News portal scrap data from hundreds of source online and collect top news from there and then go under analysis which helps user to track interested and similar news retrieval easily.
* The system developed can be used daily by all type of users and can play a major key role in IT sector where news plays a important role.
* Everyone wants to get up to date with the nearby activities so I facilitated the social theme where user can post their own news which can be easily viewed by nearby peoples.
* Fluid and easy to use UI developed on SPA framework (Angular). Which most news delivering website miss.
* Realtime data updates and live chat is provided as a bonus to gossip about the topic of interest which again most news website miss.

**PROBLEM STATEMENT**

**News analysis and a portal which uses the analysed data to provide better result to user.**

***Keywords:***

Unsupervised Learning, K-Means clustering, Latent Dirichlet Allocation, Text-preprocessing,

Natural Language processing, Title modeling, News Scraping.

**CHAPTER 2**

**OBJECTIVES**

|  |  |
| --- | --- |
| **1** | **To study various tools and techniques available for text analysis.** |
| **2** | **To develop a complete open source project which reflect its use in real world.** |

**CHAPTER 3**

**REQUIREMENTS**

4.1. Software Requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Item | Version | Spec | Vendor | Price | Description | Reference |
| 1 | Spyder  with Anaconda&Jupyter Notebook | 4 | An  Interactive  Python IDE | Open Source licensed under MIT | 0 | Used in performing scientific  programming in python. | [1] |
| 3 | Typescript | ~3.2.2 | A Superset of JavaScript  to facilitate oops | Licensed under  Apache  License 2.0 | 0 | Used in writing  OOPS scripts for  Angular | [3] |
| 4 | Angular | 7.0.0 | Frontend framework | Open Source licensed under MIT | 0 | Used to develop reactive frontend for web projects | [4] |
| 5 | NodeJs | 10.15.3 | Facilitate js compiling on backend | Open Source licensed under MIT | 0 | Used to develop backend and also  it has a package *npm* for  dependency management | [5] |

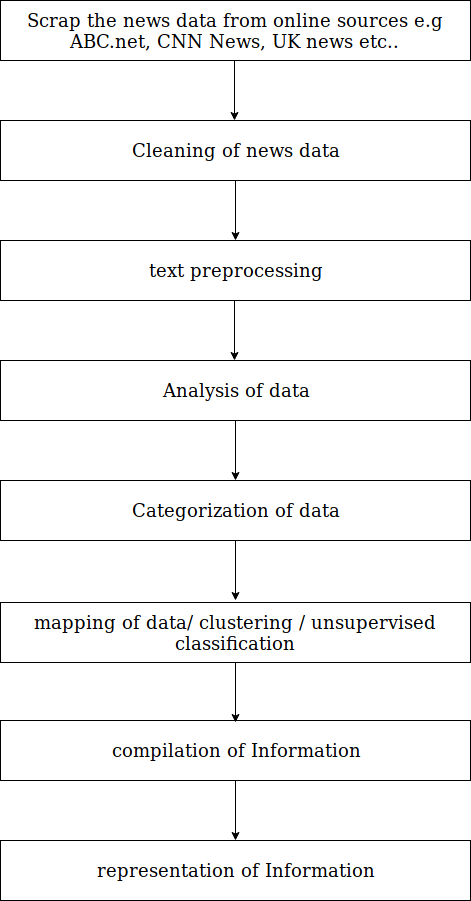
Table 1: Summary of Software Components

With the use of Open Source software, I aim at reducing the cost of the project.

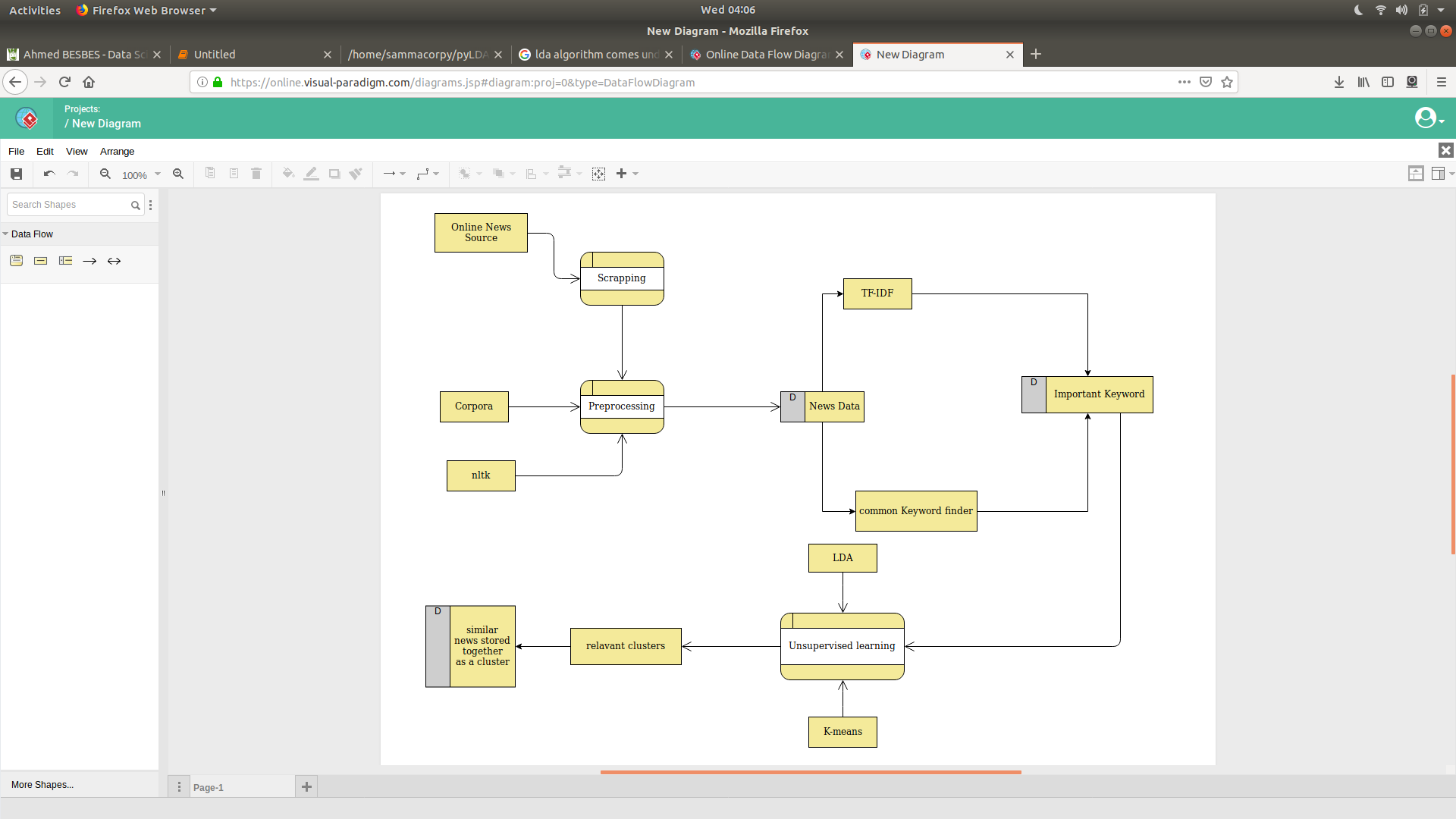
**CHAPTER 4**

**Architecture**

1. Task Flow Diagram



2. Data Flow Diagram



**CHAPTER 5**

**DEVELOPMENT DETAILS**

**6.1. Software Library Specifications:**

The model is developed is using Python 3. The following important libraries from the same will be used:

1. scikit-learn – The library is responsible for performing various Machine learning algorithm and models its a very good framework for ML who are a beginner in ML.
2. Matplotlib – The library is well known for its plotting functions. It would be useful in visualizing the way in which the car images are processed after different steps.
3. NLTK – Natural language processing libraries.
4. Pandas – This library responsible for Data representation, filtration and plotting of data
5. numpy – The library is known for its scientific and mathematical operations. It is a powerful tool in linear algebra, array processing, random number generation, etc.

**CHAPTER 5**

**Detailed Design**

1. **News Scrapping**

News gathering is done through online available API named newsapi.org from where you can  
gather all daily news data. It provide news data by collecting it from hundred different news providers.

1. **Data filtering**

Using pandas tool clean the data by removing junk and null defined data. Drop all the rows which have necessary filled missing and fill random or homogeneous/average data in less important field e.g timestamp.

1. **Pre-Text processing**

After a text is obtained, we start with text normalization. Text normalization includes:

1. removing punctuations, accent marks and other diacritics
2. removing white spaces
3. expanding abbreviations
4. removing stop words, sparse terms, and particular words
5. text canonicalization

#code for preprocessing of text data.

def \_removeNonAscii(s):

return "".join(i for i in s if ord(i)<128)

def clean\_text(text):

text = text.lower()

text = re.sub(r"what's", "what is ", text)

text = text.replace('(ap)', '')

text = re.sub(r"\'s", " is ", text)

text = re.sub(r"\'ve", " have ", text)

text = re.sub(r"can't", "cannot ", text)

text = re.sub(r"n't", " not ", text)

text = re.sub(r"i'm", "i am ", text)

text = re.sub(r"\'re", " are ", text)

text = re.sub(r"\'d", " would ", text)

text = re.sub(r"\'ll", " will ", text)

text = re.sub(r'\W+', ' ', text)

text = re.sub(r'\s+', ' ', text)

text = re.sub(r"\\", "", text)

text = re.sub(r"\'", "", text)

text = re.sub(r"\"", "", text)

text = re.sub('[^a-zA-Z ?!]+', '', text)

text = \_removeNonAscii(text)

text = text.strip()

return text

def tokenizer(text):

text = clean\_text(text)

tokens = [word\_tokenize(sent) for sent in sent\_tokenize(text)]

tokens = list(reduce(lambda x,y: x+y, tokens))

tokens = list(filter(lambda token: token not in (stop\_words + list(punctuation)) , tokens))

return tokens

1. **Important token / keywords filtering**

Use Counter to find most used tokens in that News data but it may cause problem due to common words like people, Year, Sunday.

So to remove this problem use TF-IDF Term frequency and inverse document frequency to find most important tokens from the doc.

1. **Dimension decompostion**

t-distributed stochastic neighbor embedding and singular value decomposition is used as a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

1. **Clustering**

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. I used two clustering algorithm first Kmeans and second LDA

1. **Compiling of information**

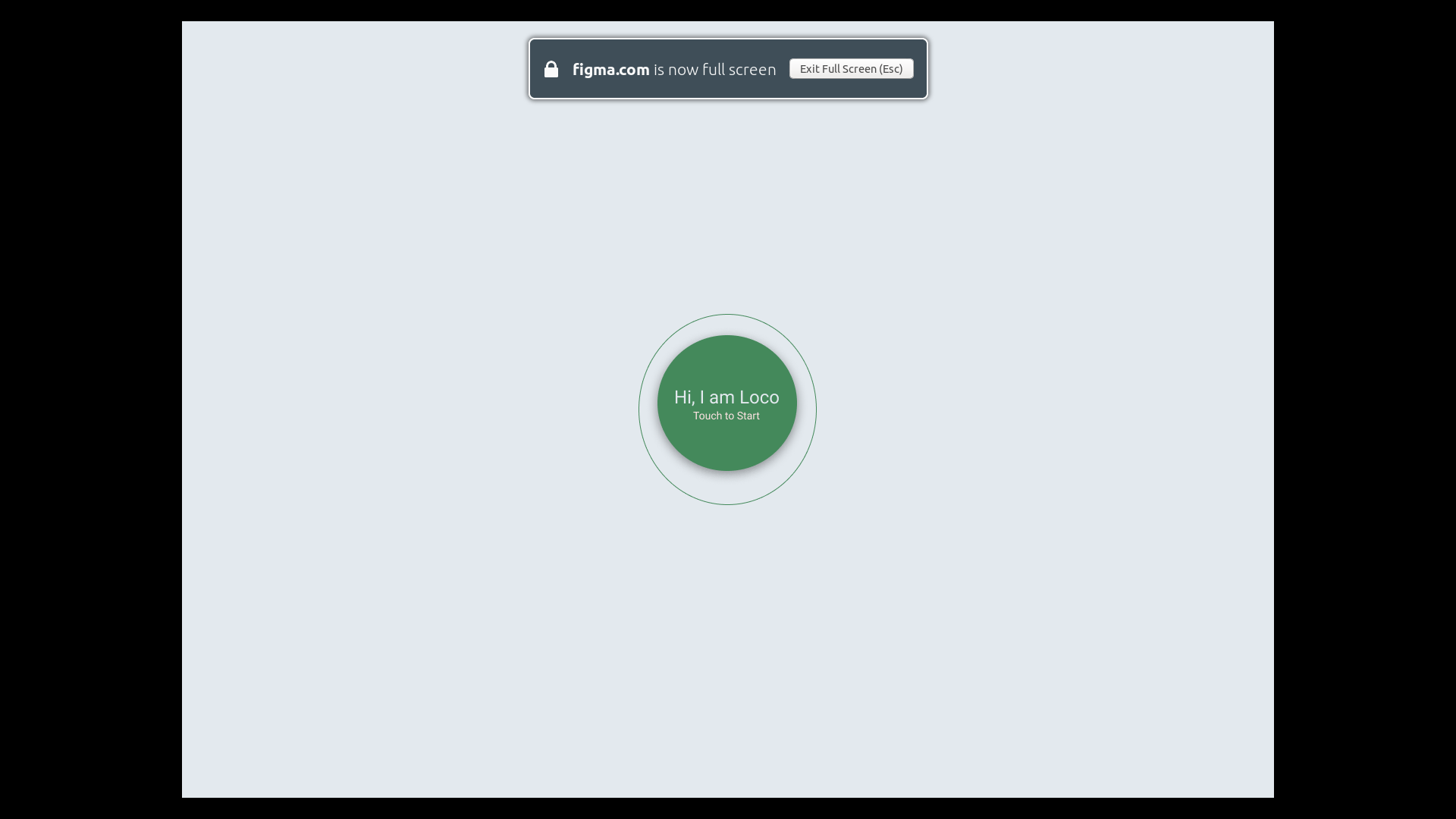
All clusters are filtered and all news in each cluster are tagged with specific ID so that user can easily find one whole cluster from any of its child news.

1. **presentation of information**

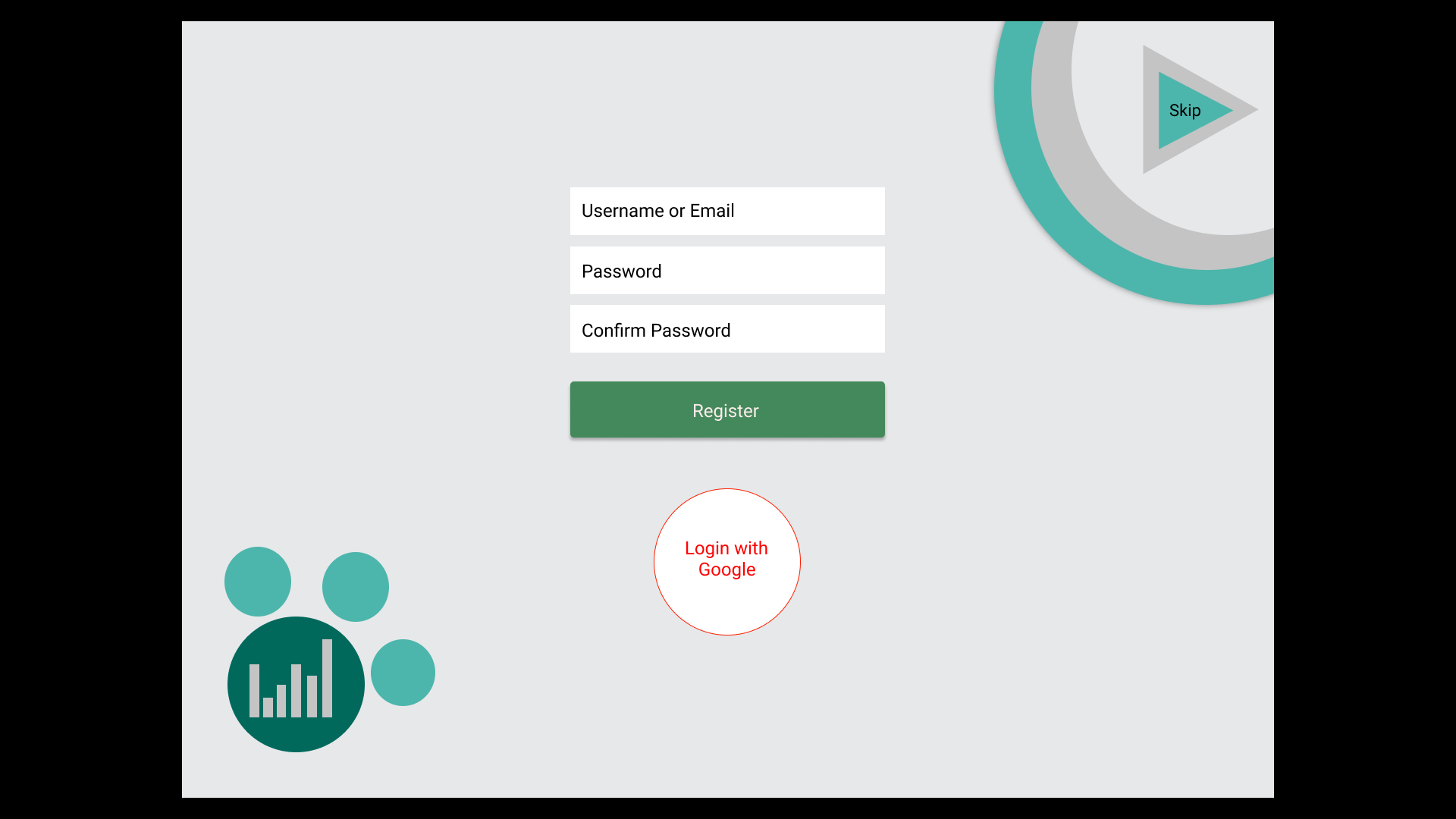
Developed a UI for News portal where we present our information. I call that app ‘Loco’

**UI Design of the Loco:**

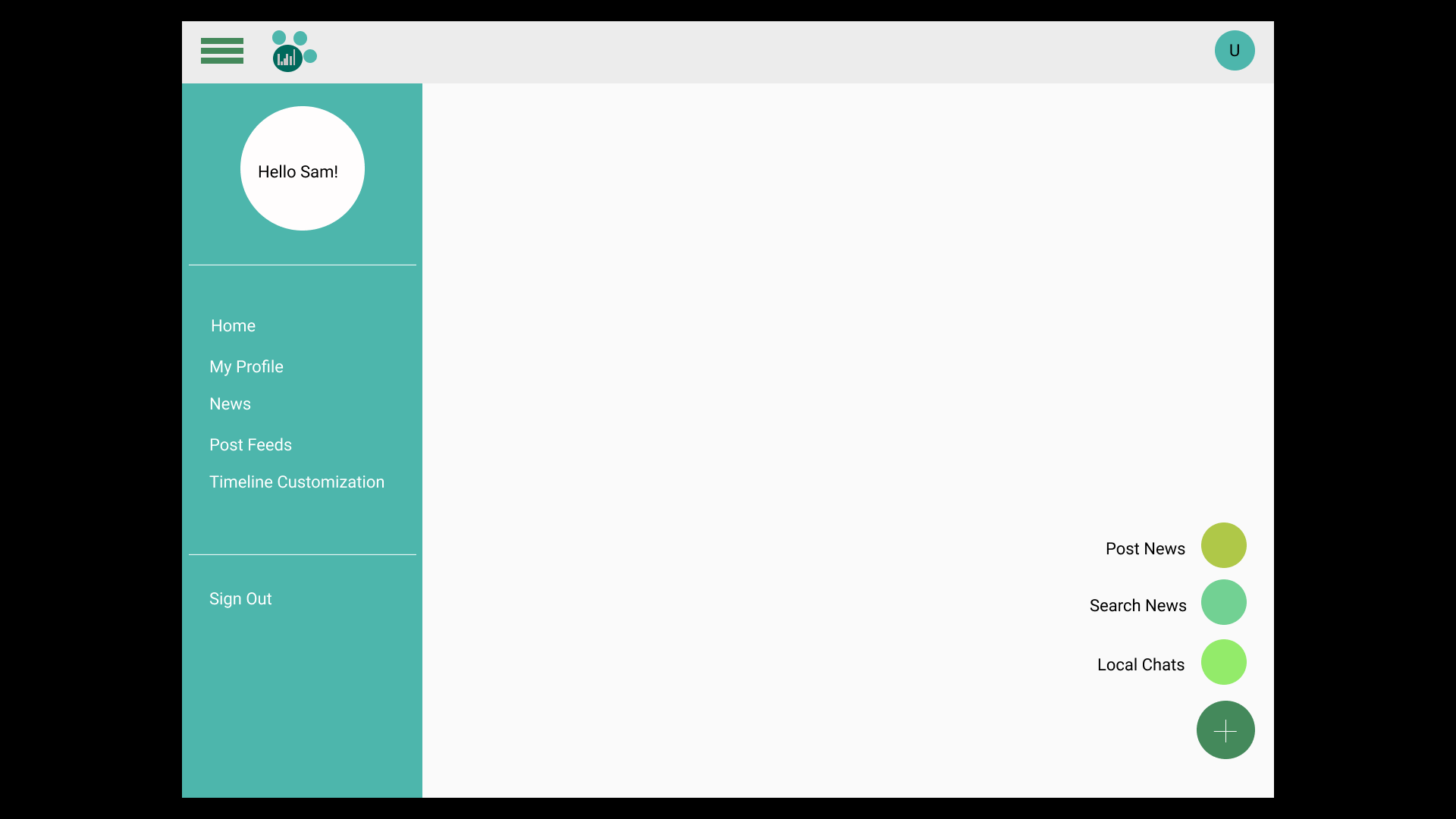
**First Landing Page:**

****

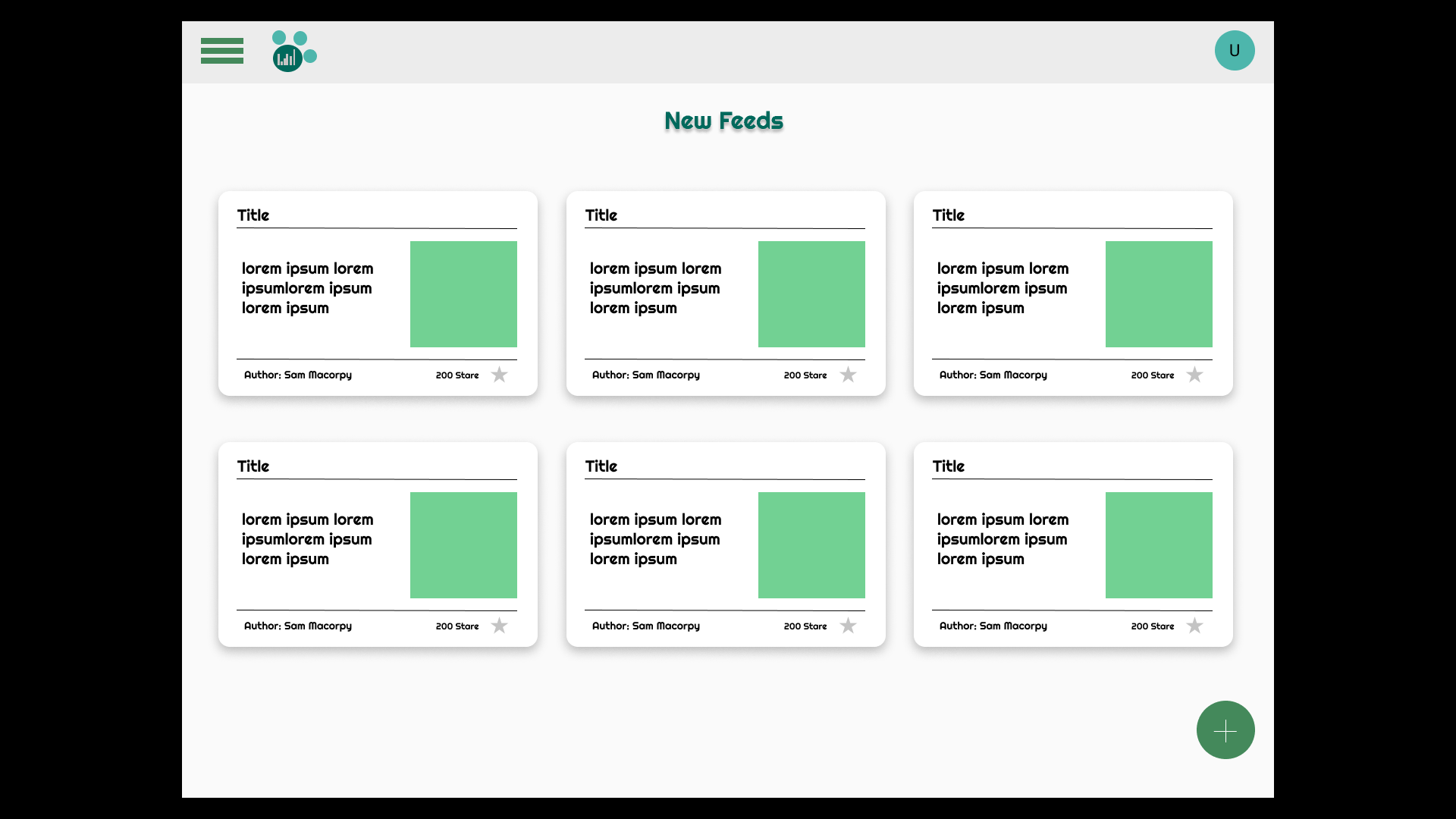
**Signup/Login Page:**

****

**Navigation Tools (side nav + Quick nav)**

****

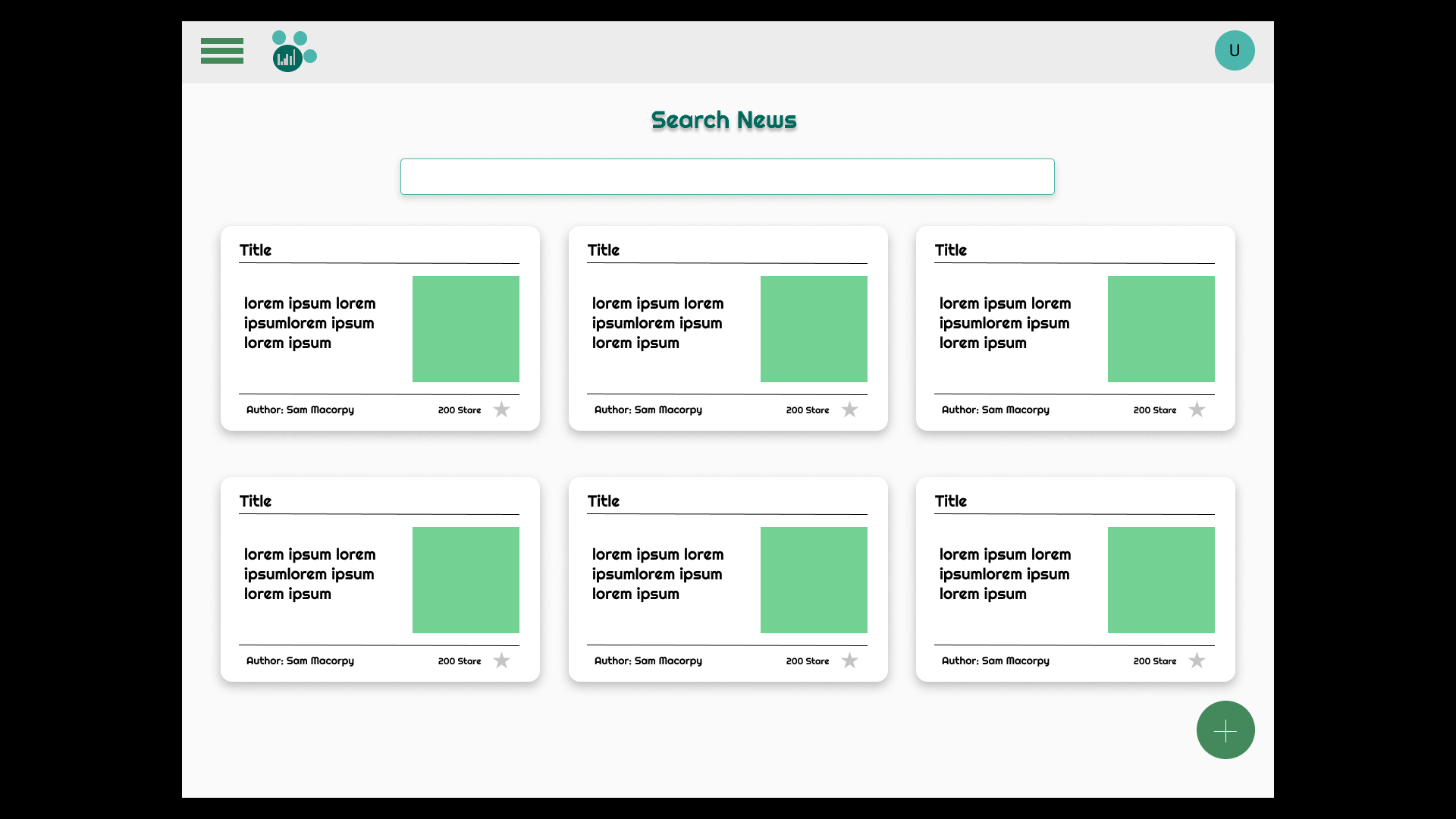
**Home Page (New Feeds | Timeline)**



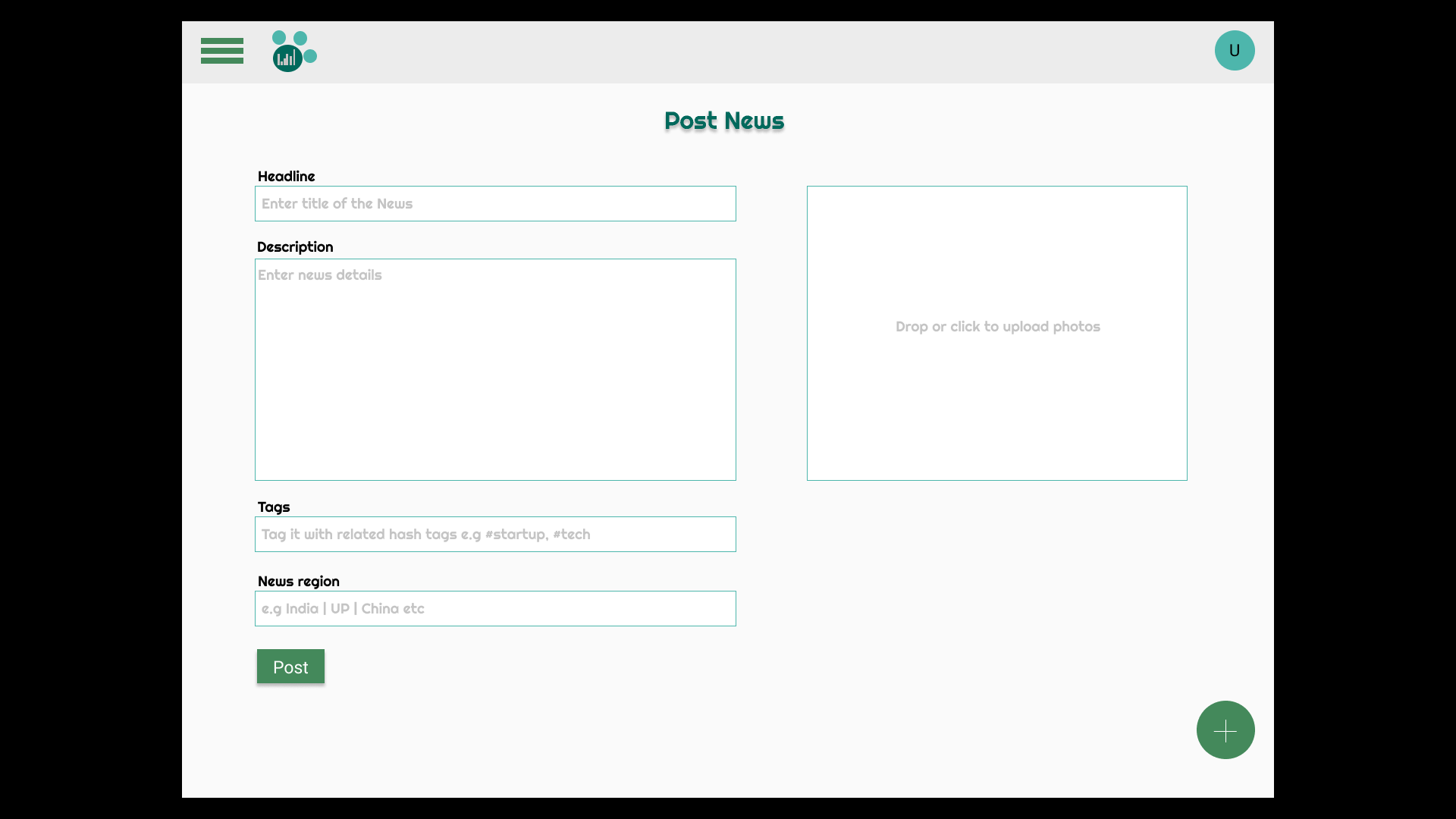
**Detailed News:**



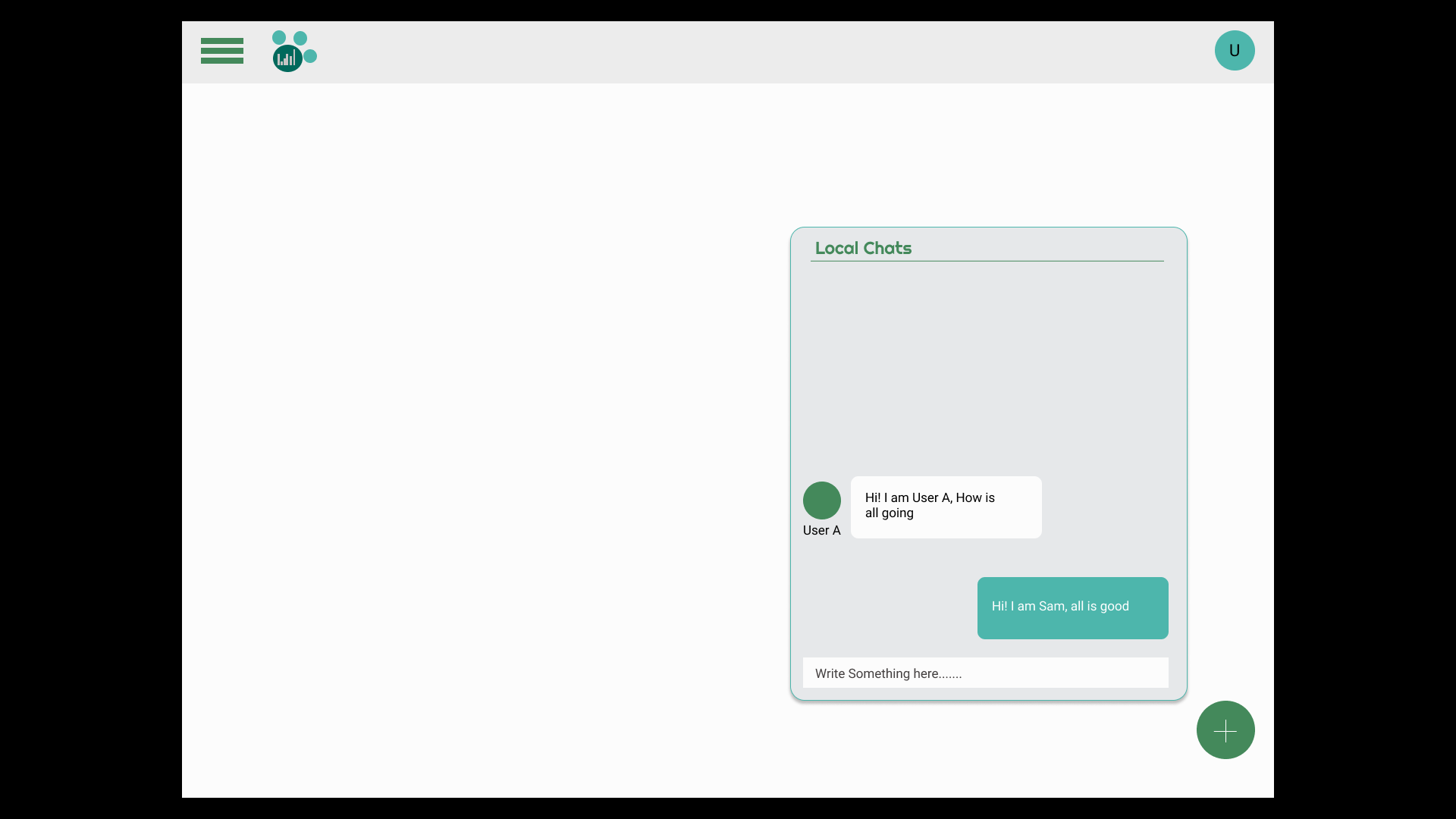
**Search feeds page:**



**Post News:**



**Local Chats:**

****

**Tools and Techs USED:**

**Frontend**

* Angular
* Bootstrap
* Material design
* HTML, CSS & TypeScript
* Angularfire5 SDK
* Angular Material SDK

**Backend**

* Django Python Framework
* Firebase
* Firestore

**Tools**

* VS Code
* Chrome Development Tools
* **Figma UI designing Tool**

**Chapter 6**

**Implementation Code:**

**Scrapping news:**

#!/usr/bin/env python3

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

"""

Created on Fri Mar 1 11:58:33 2019

@author: sammacorpy

"""

import requests

from bs4 import BeautifulSoup

import pandas as pd

from datetime import datetime

from tqdm import tqdm, tqdm\_notebook

from functools import reduce

def getSources():

source\_url = 'https://newsapi.org/v1/sources?language=en'

response = requests.get(source\_url).json()

sources = []

for source in response['sources']:

sources.append(source['id'])

return sources

def mapping():

d = {}

response = requests.get('https://newsapi.org/v1/sources?language=en')

response = response.json()

for s in response['sources']:

d[s['id']] = s['category']

return d

def category(source, m):

try:

return m[source]

except:

return 'NC'

def getDailyNews():

sources = getSources()

key = '6b25a55c0566498a83bd13852b7331ec'

url = 'https://newsapi.org/v1/articles?source={0}&sortBy={1}&apiKey={2}'

responses = []

for i, source in tqdm\_notebook(enumerate(sources), total=len(sources)):

try:

u = url.format(source, 'top', key)

except:

u = url.format(source, 'latest', key)

response = requests.get(u)

r = response.json()

try:

for article in r['articles']:

article['source'] = source

responses.append(r)

except:

print('Rate limit exceeded ... please wait and retry in 6 hours')

return None

articles = list(map(lambda r: r['articles'], responses))

articles = list(reduce(lambda x,y: x+y, articles))

news = pd.DataFrame(articles)

news = news.dropna()

news = news.drop\_duplicates()

news.reset\_index(inplace=True, drop=True)

d = mapping()

news['category'] = news['source'].map(lambda s: category(s, d))

news['scraping\_date'] = datetime.now()

try:

aux = pd.read\_csv('./data/news.csv')

aux = aux.append(news)

aux = aux.drop\_duplicates('url')

aux.reset\_index(inplace=True, drop=True)

aux.to\_csv('./data/news.csv', encoding='utf-8', index=False)

except:

news.to\_csv('./data/news.csv', index=False, encoding='utf-8')

print('Done')

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

getDailyNews()

**News analysis code:**

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

"""

Spyder Editor

This is a temporary script file.

"""

import pandas as pd

pd.options.display.max\_columns = 200

pd.options.mode.chained\_assignment = None

from nltk.tokenize import word\_tokenize, sent\_tokenize

from nltk.corpus import stopwords

stop = set(stopwords.words('english'))

from string import punctuation

from functools import reduce

from collections import Counter

import re

import numpy as np

from tqdm import tqdm\_notebook

tqdm\_notebook().pandas()

data = pd.read\_csv('./data/news.csv')

print(data.shape)

print(data.head(3))

data.category.value\_counts(normalize=True).plot(kind='bar', grid=True, figsize=(16, 9))

data = data.drop\_duplicates('description')

data = data[~data['description'].isnull()]

print(data.shape)

data = data[(data.description.map(len) > 140) & (data.description.map(len) <= 300)]

data.reset\_index(inplace=True, drop=True)

print(data.shape)

data.description.map(len).hist(figsize=(15, 5), bins=100)

data = data.sample(200, random\_state=42,replace =True)

data.reset\_index(inplace=True, drop=True)

data.head(2)

stop\_words = []

f = open('./data/stopwords.txt', 'r')

for l in f.readlines():

stop\_words.append(l.replace('\n', ''))

additional\_stop\_words = ['t', 'will']

stop\_words += additional\_stop\_words

print(len(stop\_words))

def \_removeNonAscii(s):

return "".join(i for i in s if ord(i)<128)

def clean\_text(text):

text = text.lower()

text = re.sub(r"what's", "what is ", text)

text = text.replace('(ap)', '')

text = re.sub(r"\'s", " is ", text)

text = re.sub(r"\'ve", " have ", text)

text = re.sub(r"can't", "cannot ", text)

text = re.sub(r"n't", " not ", text)

text = re.sub(r"i'm", "i am ", text)

text = re.sub(r"\'re", " are ", text)

text = re.sub(r"\'d", " would ", text)

text = re.sub(r"\'ll", " will ", text)

text = re.sub(r'\W+', ' ', text)

text = re.sub(r'\s+', ' ', text)

text = re.sub(r"\\", "", text)

text = re.sub(r"\'", "", text)

text = re.sub(r"\"", "", text)

text = re.sub('[^a-zA-Z ?!]+', '', text)

text = \_removeNonAscii(text)

text = text.strip()

return text

def tokenizer(text):

text = clean\_text(text)

tokens = [word\_tokenize(sent) for sent in sent\_tokenize(text)]

tokens = list(reduce(lambda x,y: x+y, tokens))

tokens = list(filter(lambda token: token not in (stop\_words + list(punctuation)) , tokens))

return tokens

data['description'] = data['description'].map(lambda d: bytes.decode(d.encode('utf-8')))

data['tokens'] = data['description'].progress\_map(lambda d: tokenizer(d))

for descripition, tokens in zip(data['description'].head(5), data['tokens'].head(5)):

print('description:', descripition)

print('tokens:', tokens)

print()

def keywords(category):

tokens = data[data['category'] == category]['tokens']

alltokens = []

for token\_list in tokens:

alltokens += token\_list

counter = Counter(alltokens)

return counter.most\_common(10)

for category in set(data['category']):

print('category :', category)

print('top 10 keywords:', keywords(category))

print('---')

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min\_df=5, analyzer='word', ngram\_range=(1, 2), stop\_words='english')

vz = vectorizer.fit\_transform(list(data['tokens'].map(lambda tokens: ' '.join(tokens))))

vz.shape

tfidf = dict(zip(vectorizer.get\_feature\_names(), vectorizer.idf\_))

tfidf = pd.DataFrame(columns=['tfidf']).from\_dict(dict(tfidf), orient='index')

tfidf.columns = ['tfidf']

tfidf.tfidf.hist(bins=25, figsize=(15,7))

from wordcloud import WordCloud

import matplotlib.pyplot as plt

def plot\_word\_cloud(terms):

text = terms.index

text = ' '.join(list(text))

# lower max\_font\_size

wordcloud = WordCloud(max\_font\_size=40).generate(text)

plt.figure(figsize=(25, 25))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()

plot\_word\_cloud(tfidf.sort\_values(by=['tfidf'], ascending=True).head(40))

plot\_word\_cloud(tfidf.sort\_values(by=['tfidf'], ascending=False).head(40))

from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n\_components=50, random\_state=0)

svd\_tfidf = svd.fit\_transform(vz)

svd\_tfidf.shape

run = False

if run:

# run this (takes times)

from sklearn.manifold import TSNE

tsne\_model = TSNE(n\_components=2, verbose=1, random\_state=0, n\_iter=500)

tsne\_tfidf = tsne\_model.fit\_transform(svd\_tfidf)

print(tsne\_tfidf.shape)

tsne\_tfidf\_df = pd.DataFrame(tsne\_tfidf)

tsne\_tfidf\_df.columns = ['x', 'y']

tsne\_tfidf\_df['category'] = data['category']

tsne\_tfidf\_df['description'] = data['description']

tsne\_tfidf\_df.to\_csv('./data/tsne\_tfidf.csv', encoding='utf-8', index=False)

else:

# or import the dataset directly

tsne\_tfidf\_df = pd.read\_csv('./data/tsne\_tfidf.csv')

groups = tsne\_tfidf\_df.groupby('category')

fig, ax = plt.subplots(figsize=(15, 10))

ax.margins(0.05) # Optional, just adds 5% padding to the autoscaling

for name, group in groups:

ax.plot(group.x, group.y, marker='o', linestyle='', label=name)

ax.legend()

plt.show()

import bokeh.plotting as bp

from bokeh.models import HoverTool, BoxSelectTool

from bokeh.plotting import figure, show, output\_notebook, reset\_output

from bokeh.palettes import d3

import bokeh.models as bmo

from bokeh.io import save, output\_file

output\_notebook()

plot\_tfidf = bp.figure(plot\_width=700, plot\_height=600, title="tf-idf clustering of the news",

tools="pan,wheel\_zoom,box\_zoom,reset,hover,previewsave",

x\_axis\_type=None, y\_axis\_type=None, min\_border=1)

palette = d3['Category10'][len(tsne\_tfidf\_df['category'].unique())]

color\_map = bmo.CategoricalColorMapper(factors=tsne\_tfidf\_df['category'].map(str).unique(), palette=palette)

plot\_tfidf.scatter(x='x', y='y', color={'field': 'category', 'transform': color\_map},

legend='category', source=tsne\_tfidf\_df)

hover = plot\_tfidf.select(dict(type=HoverTool))

hover.tooltips={"description": "@description", "category":"@category"}

show(plot\_tfidf)

vz.shape

import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning)

from sklearn.cluster import MiniBatchKMeans, KMeans

from sklearn.metrics import silhouette\_score

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import Normalizer

distorsions = []

sil\_scores = []

k\_max = 80

for k in tqdm\_notebook(range(2, k\_max)):

kmeans\_model = MiniBatchKMeans(n\_clusters=k, init='k-means++', n\_init=1, random\_state=42,

init\_size=1000, verbose=False, max\_iter=1000)

kmeans\_model.fit(vz)

sil\_score = silhouette\_score(vz, kmeans\_model.labels\_)

sil\_scores.append(sil\_score)

distorsions.append(kmeans\_model.inertia\_)

f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(15, 10))

ax1.plot(range(2, k\_max), distorsions)

ax1.set\_title('Distorsion vs num of clusters')

ax1.grid(True)

ax2.plot(range(2, k\_max), sil\_scores)

ax2.set\_title('Silhouette score vs num of clusters')

ax2.grid(True)

num\_clusters = 40

kmeans\_model = MiniBatchKMeans(n\_clusters=num\_clusters, init='k-means++', n\_init=1, random\_state=42,

init\_size=1000, batch\_size=1000, verbose=False, max\_iter=1000, )

kmeans = kmeans\_model.fit(vz)

kmeans\_clusters = kmeans.predict(vz)

kmeans\_distances = kmeans.transform(vz)

for (i, desc),category in zip(enumerate(data.description),data['category']):

if(i < 25):

print("Cluster " + str(kmeans\_clusters[i]) + ": " + desc +

"(distance: " + str(kmeans\_distances[i][kmeans\_clusters[i]]) + ")")

print('category: ',category)

print('---')

sorted\_centroids = kmeans.cluster\_centers\_.argsort()[:, ::-1]

terms = vectorizer.get\_feature\_names()

all\_keywords = []

for i in range(num\_clusters):

topic\_keywords = []

for j in sorted\_centroids[i, :10]:

topic\_keywords.append(terms[j])

all\_keywords.append(topic\_keywords)

keywords\_df = pd.DataFrame(index=['topic\_{0}'.format(i) for i in range(num\_clusters)],

columns=['keyword\_{0}'.format(i) for i in range(10)],

data=all\_keywords)

keywords\_df

warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')

import gensim

import gensim.corpora as corpora

from gensim import matutils

from gensim.models import CoherenceModel

aux = data.copy()

bigram = gensim.models.Phrases(aux['tokens'], min\_count=5, threshold=100)

bigram\_mod = gensim.models.phrases.Phraser(bigram)

aux['tokens\_bigram'] = aux['tokens'].progress\_map(lambda tokens: bigram\_mod[tokens])

id2word = corpora.Dictionary(aux['tokens\_bigram'])

texts = aux['tokens\_bigram'].values

corpus = [id2word.doc2bow(text) for text in texts]

def LDA\_model(num\_topics, passes=1):

return gensim.models.ldamodel.LdaModel(corpus=tqdm\_notebook(corpus, leave=False),

id2word=id2word,num\_topics=num\_topics, random\_state=100, eval\_every=10,chunksize=2000, passes=passes, per\_word\_topics=True)

def compute\_coherence(model):

coherence = CoherenceModel(model=model,

texts=aux['tokens\_bigram'].values,

dictionary=id2word, coherence='c\_v')

return coherence.get\_coherence()

def display\_topics(model):

topics = model.show\_topics(num\_topics=model.num\_topics, formatted=False, num\_words=10)

topics = map(lambda c: map(lambda cc: cc[0], c[1]), topics)

df = pd.DataFrame(topics)

df.index = ['topic\_{0}'.format(i) for i in range(model.num\_topics)]

df.columns = ['keyword\_{0}'.format(i) for i in range(1, 10+1)]

return df

def explore\_models(df, rg=range(5, 25)):

id2word = corpora.Dictionary(df['tokens\_bigram'])

texts = df['tokens\_bigram'].values

corpus = [id2word.doc2bow(text) for text in texts]

models = []

coherences = []

for num\_topics in tqdm\_notebook(rg, leave=False):

lda\_model = LDA\_model(num\_topics, passes=5)

models.append(lda\_model)

coherence = compute\_coherence(lda\_model)

coherences.append(coherence)

fig = plt.figure(figsize=(15, 5))

plt.title('Choosing the optimal number of topics')

plt.xlabel('Number of topics')

plt.ylabel('Coherence')

plt.grid(True)

plt.plot(rg, coherences)

return coherences, models

best\_model = LDA\_model(num\_topics=20, passes=10)

def get\_document\_topic\_matrix(corpus, num\_topics=best\_model.num\_topics):

matrix = []

for row in tqdm\_notebook(corpus):

output = np.zeros(num\_topics)

doc\_proba = best\_model[row][0]

for doc, proba in doc\_proba:

output[doc] = proba

matrix.append(output)

matrix = np.array(matrix)

return matrix

matrix = get\_document\_topic\_matrix(corpus)

doc\_topic = best\_model.get\_document\_topics(corpus)

lda\_keys = []

for i, desc in enumerate(data['description']):

lda\_keys.append(np.argmax(matrix[i, :]))

tsne\_model = TSNE(n\_components=2, verbose=1, random\_state=0, n\_iter=500)

tsne\_lda = tsne\_model.fit\_transform(matrix)

lda\_df = pd.DataFrame(tsne\_lda, columns=['x', 'y'])

lda\_df['topic'] = lda\_keys

lda\_df['topic'] = lda\_df['topic'].map(str)

lda\_df['description'] = data['description']

lda\_df['category'] = data['category']

lda\_df.to\_csv('./data/tsne\_lda.csv', index=False, encoding='utf-8')

reset\_output()

output\_notebook()

plot\_lda = bp.figure(plot\_width=700, plot\_height=600, title="KMeans clustering of the news",

tools="pan,wheel\_zoom,box\_zoom,reset,hover,previewsave",

x\_axis\_type=None, y\_axis\_type=None, min\_border=1)

palette = d3['Category20'][20] + d3['Category20b'][20]

color\_map = bmo.CategoricalColorMapper(factors=lda\_df['topic'].unique(), palette=palette)

plot\_lda.scatter('x', 'y', source=lda\_df,

color={'field': 'topic', 'transform': color\_map},

legend='topic')

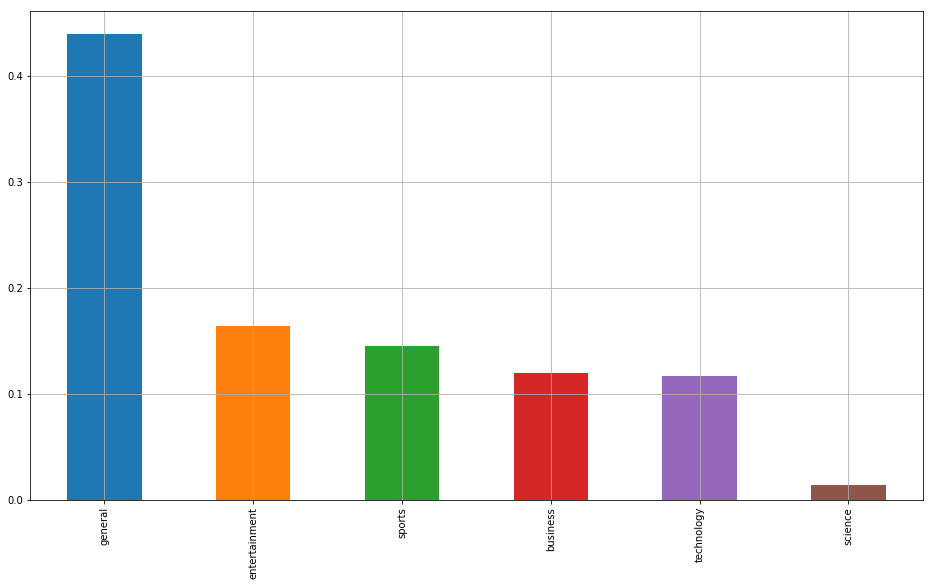
hover = plot\_lda.select(dict(type=HoverTool))

hover.tooltips={"description": "@description", "topic": "@topic", "category": "@category"}

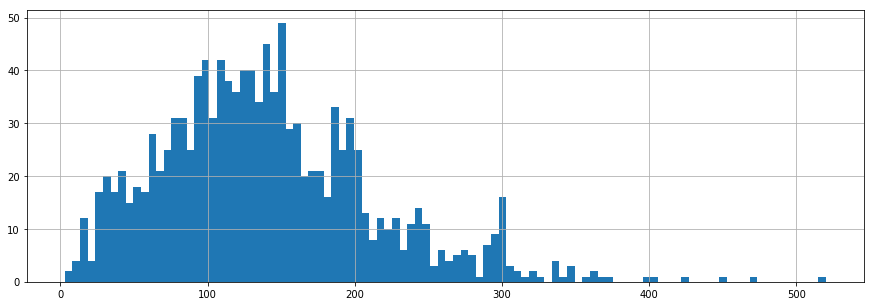
show(plot\_lda)

Output:

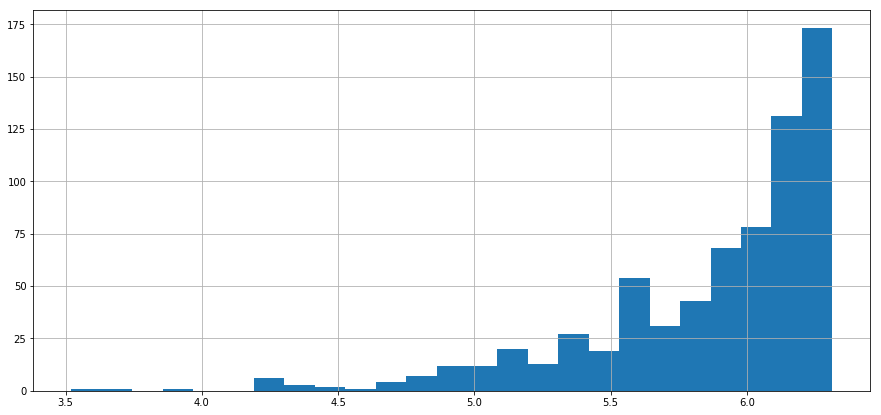
1. News data category plotting



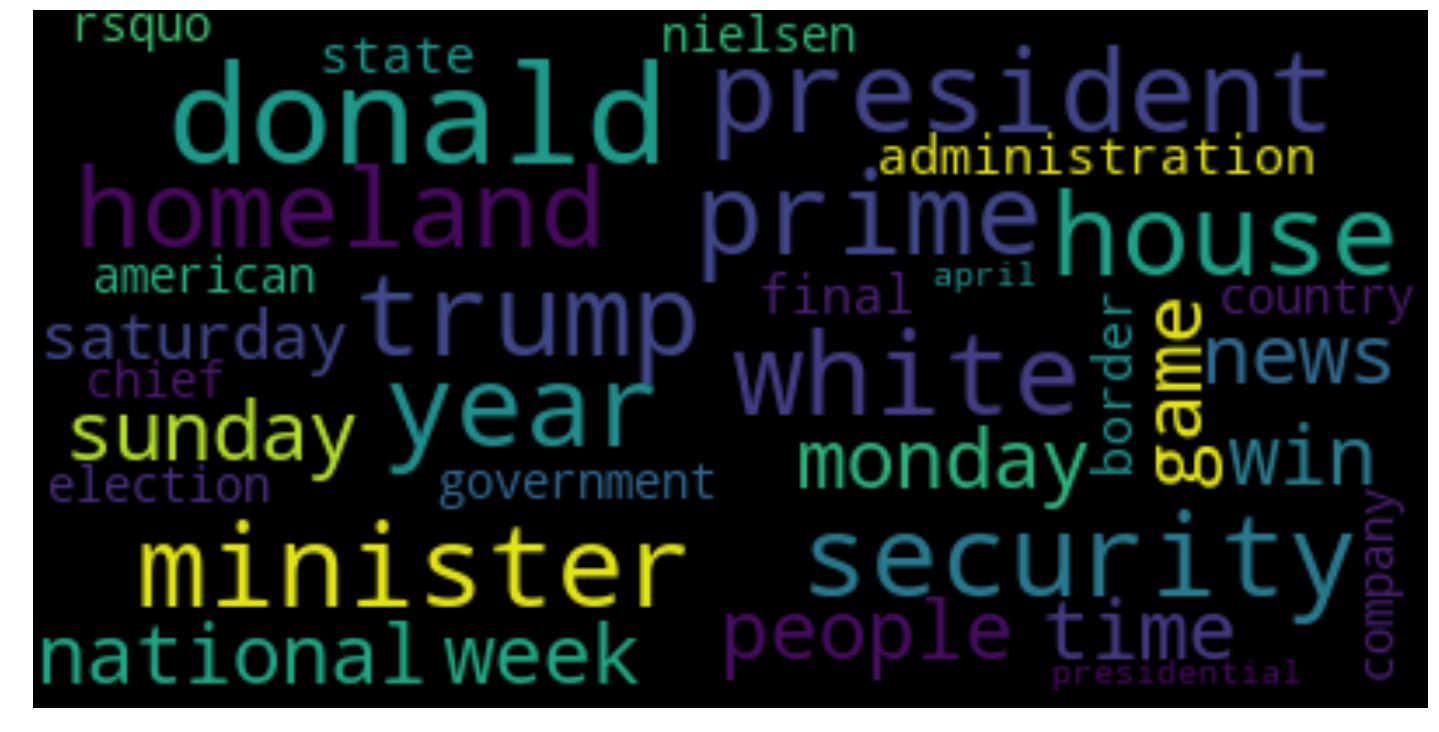
**2. Tokens frequency in each news**

****

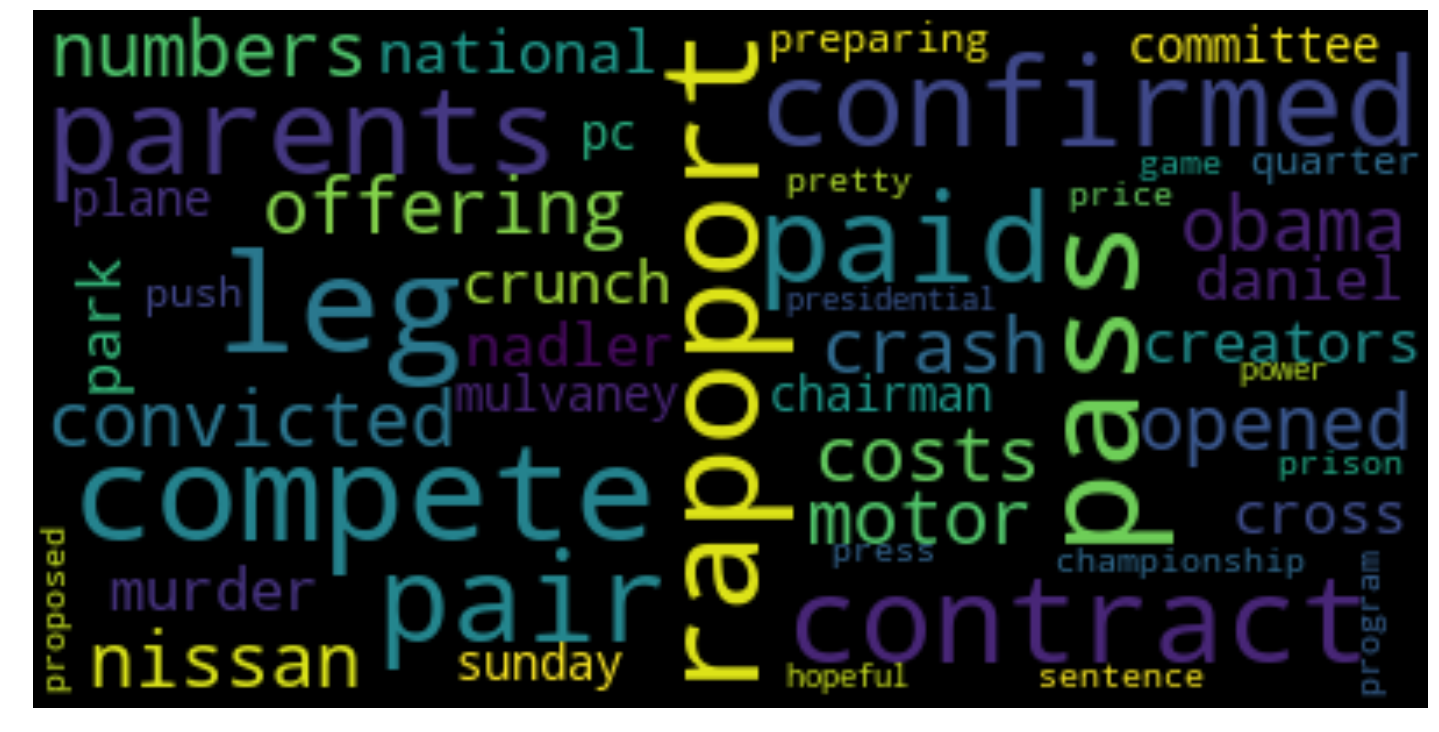
**3. TF-IDF plotting**

****

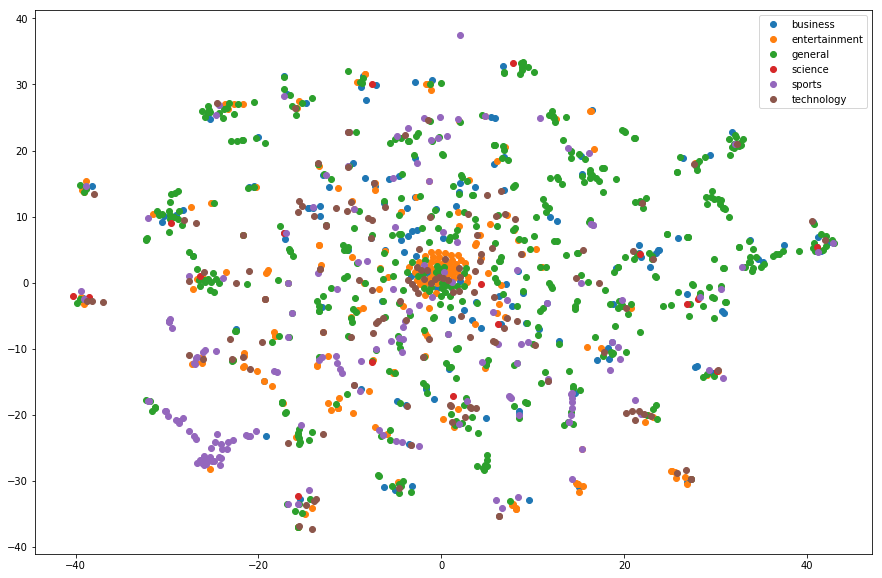
**4. less important words**

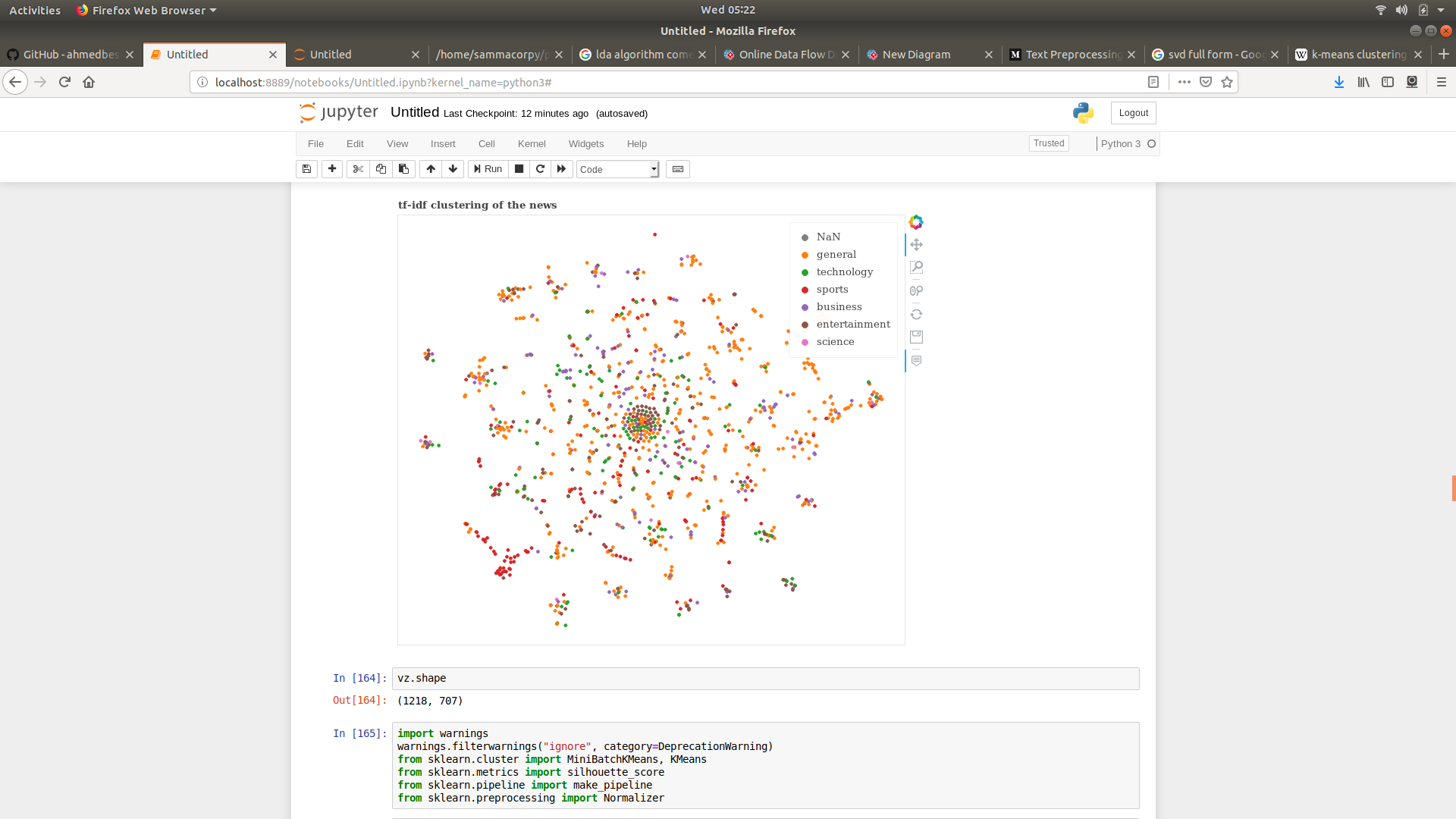
****

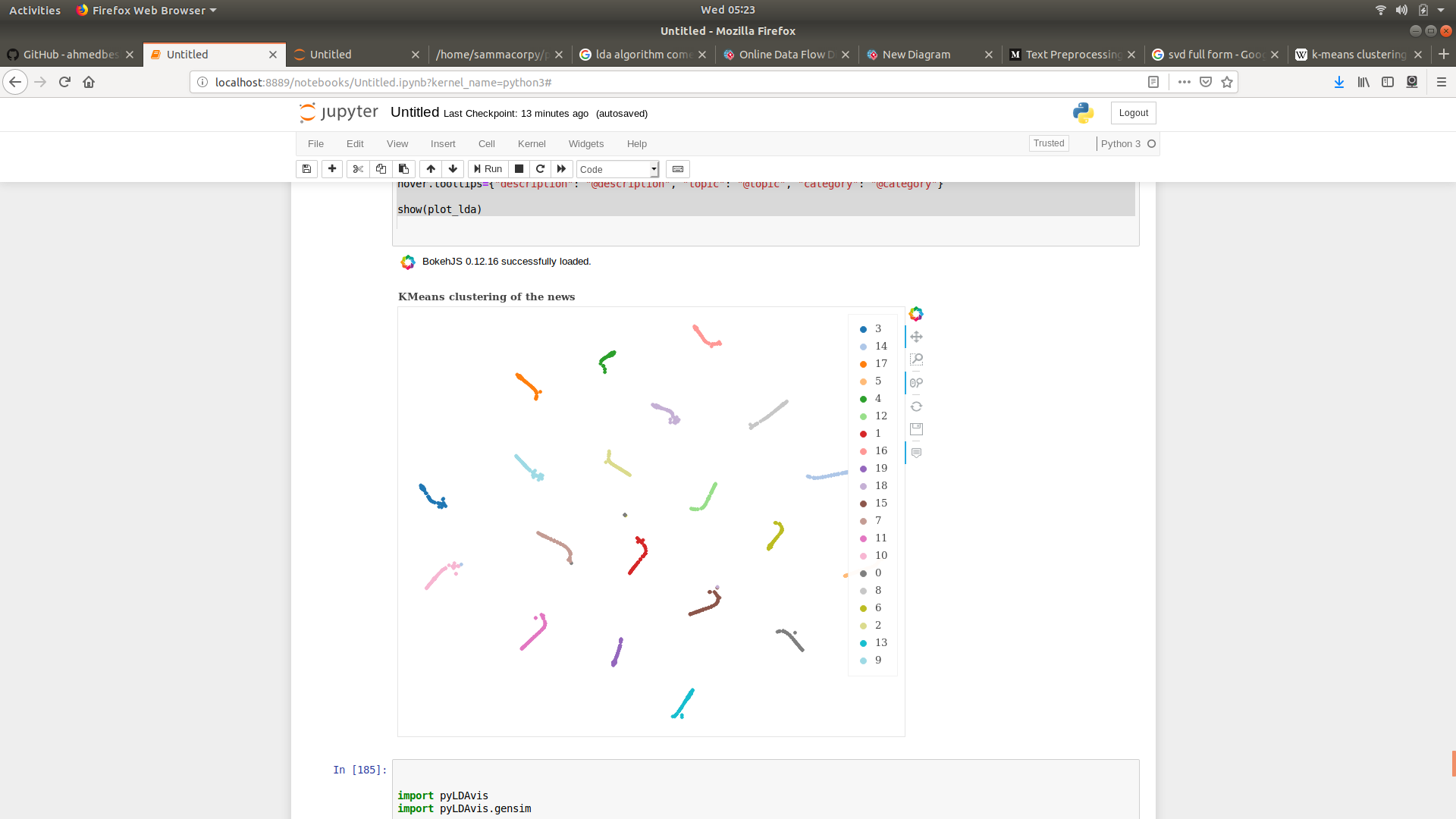
**5. more Important words**

****

**6. Clustering based on above important token words in news doc.**

****

**7.**



8.

**Git hub repository link :**

**https://github.com/sammacorpy/News-analysis**

My project is licensed under MIT open source license on github.

**Outcome:**  The outcome of project was a full fledged web portal with all the cool features:

1. cluster based searching of news.

2. offline news reading.

3. news posting work even in no internet connection (using background synchronization).

4. Chat with locals without sending friend request.

5. mordern UI design.

6. Decoupled routing.

7. Quick nav like in Iphones.

8. can Share news on facebook and linkedIn.

Link to hosted Website

[https://ainews-42800.firebaseapp.com](https://ainews-42800.firebaseapp.com/)

