Mostafa Esmaeili, PhD

Suryateja Chalapati (U3699-1670)

ISM6905 – Text Analytics using Topic Modelling and Cosine Similarity on Kickstarter and Reddit Data

Independent Study Report

TABLE OF CONTENTS

1. INTRODUCTION | Overview

1.1. Project Overview

Summary: The topics for this independent research project is divided into two parts.

1. Kickstarter similarity analysis.
2. Reddit post content LDA topic modelling.

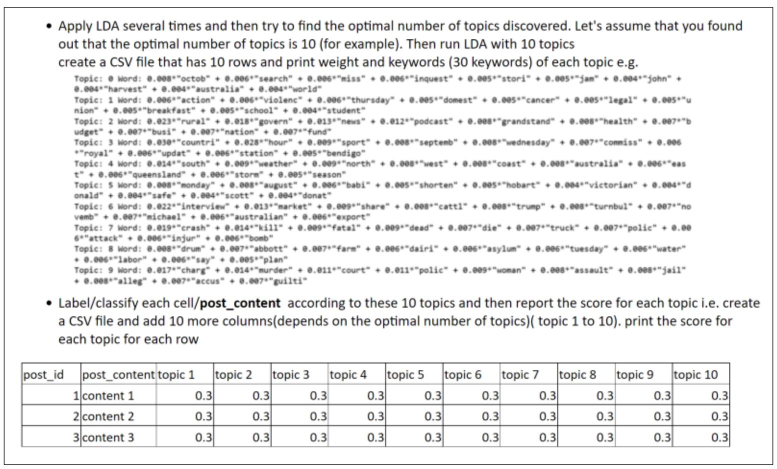
1.2. To-Do

My Tasks: These are projects with a lot of scope in predictive text analytics where I used topic modelling techniques like TF-IDF, LDA etc. to gain valuable insights from the data. My tasks for the projects were specifically related to gathering textual data and performing text analytics which is done in the following steps:

1. Extract data from the Kickstarter project page for ~3800 projects by web scraping and storing this data as a text corpus. Then perform similarity analysis on the corpus for each of the Story and Risk sections.
2. Perform Topic modelling on Reddit user content and build a Document Term Matrix.

I did the corpus extraction for both Story and Risk sections for Kickstarter project which is done by web scraping using python and wrote methods to process large amounts of text data for the Reddit project. Also involved with organizing and storing the data and performing sanity checks to make sure we have the corpus available as intended.

Technical Requirements Road Map – Reddit Project:



- Apply LDA several times and then try to find the optimal number of topics discovered. Let's assume that you found out that the optimal number of topics is 10 (for example). Then run LDA with 10 topics
  create a CSV file that has 10 rows and print weight and keywords (30 keywords) of each topic e.g.

```
Topic: 0 Word: 0.008*"octob" + 0.006*"search" + 0.006*"miss" + 0.006*"inquest" + 0.005*"stori" + 0.005*"jam" + 0.004*"john" +
0.004*"harvest" + 0.004*"australia" + 0.004*"world"
Topic: 1 Word: 0.006*"action" + 0.006*"violenc" + 0.006*"thursday" + 0.005*"domest" + 0.005*"cancer" + 0.005*"legal" + 0.005*"u
nion" + 0.005*"breakfast" + 0.005*"school" + 0.004*"student"
Topic: 2 Word: 0.023*"rural" + 0.018*"govern" + 0.013*"news" + 0.012*"podcast" + 0.008*"grandstand" + 0.008*"health" + 0.007*"b
udget" + 0.007*"busi" + 0.007*"nation" + 0.007*"fund"
Topic: 3 Word: 0.030*"countri" + 0.028*"hour" + 0.009*"sport" + 0.008*"septemb" + 0.008*"wednesday" + 0.007*"commiss" + 0.006
*"royal" + 0.006*"updat" + 0.006*"station" + 0.005*"bendigo"
Topic: 4 Word: 0.014*"south" + 0.009*"weather" + 0.009*"north" + 0.008*"west" + 0.008*"coast" + 0.008*"australia" + 0.006*"eas
t" + 0.006*"queensland" + 0.006*"storm" + 0.005*"season"
Topic: 5 Word: 0.008*"monday" + 0.008*"august" + 0.006*"babi" + 0.005*"shorten" + 0.005*"hobart" + 0.004*"victorian" + 0.004*"d
onald" + 0.004*"safe" + 0.004*"scott" + 0.004*"donat"
Topic: 6 Word: 0.022*"interview" + 0.013*"market" + 0.009*"share" + 0.008*"cattl" + 0.008*"trump" + 0.008*"turnbul" + 0.007*"no
vemb" + 0.007*"michael" + 0.006*"australian" + 0.006*"export"
Topic: 7 Word: 0.019*"crash" + 0.014*"kill" + 0.009*"fatal" + 0.009*"dead" + 0.007*"die" + 0.007*"truck" + 0.007*"polic" + 0.00
6*"attack" + 0.006*"injur" + 0.006*"bomb"
Topic: 8 Word: 0.008*"drum" + 0.007*"abbott" + 0.007*"farm" + 0.006*"dairi" + 0.006*"asylum" + 0.006*"tuesday" + 0.006*"water"
+ 0.006*"labor" + 0.006*"say" + 0.005*"plan"
Topic: 9 Word: 0.017*"charg" + 0.014*"murder" + 0.011*"court" + 0.011*"polic" + 0.009*"woman" + 0.008*"assault" + 0.008*"jail"
+ 0.008*"alleg" + 0.007*"accus" + 0.007*"guilti"
```

- Label/classify each cell/**post_content** according to these 10 topics and then report the score for each topic i.e. create a CSV file and add 10 more columns(depends on the optimal number of topics)(topic 1 to 10). print the score for each topic for each row

post_id	post_content	topic 1	topic 2	topic 3	topic 4	topic 5	topic 6	topic 7	topic 8	topic 9	topic 10
1	content 1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
2	content 2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
3	content 3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Fig. 1: Flow Chart

1.3. Project Hierarchy

The project hierarchy as follows:

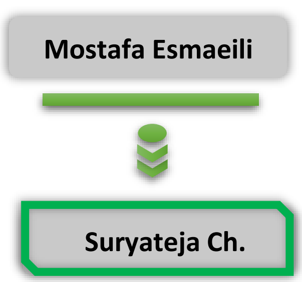


Fig. 2: Hierarchy

1.4. Literature Review

The principal goal of our literature review is to identify which libraries provide the best opportunity for similarity analysis and topic modelling, there are lot of packages that can help with text analytics like Genism, NLTK etc. For the Kickstarter project, I started with an API search to extract textual data and read up on prior work to figure out the best way to extract the text sections from the Kickstarter page where the dataset consisted of around 3800 rows of URL's. For the Reddit project, I looked through a lot of prior research on how to pre-process the data and build a text feature set for the analysis. The dataset consists of about 40k rows of post content comments, and I need to build a topic modelled matrix based on the top 10 words.

2. PROGRESS | To-Do

2.1. Kickstarter Project - Data Extraction

This script extracts Story and Risk sections from the Kickstarter page for every 200 entries. The code was scheduled on a cluster and the request query was randomized to not overwhelm the server.

```
In [4]: def kickScraper(url_list):
    slugs = []

    #extract slugs from url
    for url in url_list:
        slugs.append(re.search('/projects/(.*)\?', url).group(1))

    s = requests.Session()
    r = s.get("https://www.kickstarter.com")
    soup = BeautifulSoup(r.text, 'html.parser')
    xcsrf = soup.find("meta", {"name": "csrf-token"})["content"]

    query = """
    query Campaign($slug: String!) {
      project(slug: $slug) {
        risks
        story(assetWidth: 680)
      }
    }"""

    text = pd.DataFrame(columns=['url_T', 'story_url_T', 'risk_url_T'])

    for slug in slugs:

        max_attempts = 10
        attempts = 0

        while attempts < max_attempts:
            # If not rate limited, break out of while loop and continue with the rest of the code
            if r.status_code != 429:
                break

            # If rate limited, wait and try again
            time.sleep((2 ** attempts) + random.random())
            attempts = attempts + 1

        r = s.post("https://www.kickstarter.com/graph",
            headers= {
                "x-csrf-token": xcsrf
            },
            json = {
                "operationName":"Campaign",
                "variables":{
                    "slug": slug
                },
                "query": query
            })

        max_attempts1 = 20
        attempts1 = 0
        while attempts1 < max_attempts1:
            # If not rate limited, break out of while loop and continue with the rest of the code
            if r.status_code != 429:
                break

            # If rate limited, wait and try again
            time.sleep((2 ** attempts1) + random.random())
            attempts1 = attempts1 + 1

        result = r.json()

        story_html = result["data"]["project"]["story"]
        soup = BeautifulSoup(story_html, 'html.parser')
        story_row = ''
        for i in soup.find_all('p'):
            story_row += i.get_text().replace("\xa0", "").replace("\n", "")

        risk_row = ''
        risk_row += result["data"]["project"]["risks"].replace("\r", "").replace("\n", "")

        inText = {'url_T': slug, 'story_url_T': story_row, 'risk_url_T': risk_row}
        text = text.append(inText, ignore_index=True)

    return text
```

Fig. 3: Kickstarter Scraper

2.2. Kickstarter Project - Analysis

This script extracts the feature set from the data and performs the similarity analysis on the text section requiring a similarity score between 0-1 (Higher the better).

```
In [4]: # Import IGN's awesome_cossim_topn module
        from sparse_dot_topn import awesome_cossim_topn

        # The arguments for awesome_cossim_topn are as follows:
        ### 1. Our TF-IDF matrix
        ### 2. Our TF-IDF matrix transposed (allowing us to build a pairwise cosine matrix)
        ### 3. A top_n filter, which allows us to filter the number of matches returned, which isn't useful for our purposes
        ### 4. This is our similarity threshold. Only values over 0.8 will be returned
        cosine_matrix = awesome_cossim_topn(
          tfidf_matrix,
          tfidf_matrix.transpose(),
          vals.size,
          0.8
        )

In [5]: import re
        import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sparse_dot_topn import awesome_cossim_topn

        # Import your data to a Pandas.DataFrame
        df = pd.read_csv('Final.csv')

        # Instaniate our lookup hash table
        group_lookup = {}


        # Write a function for cleaning strings and returning an array of ngrams
        def ngrams_analyzer(string):
            string = re.sub(r'[,-./]', r'', string)
            ngrams = zip(*[string[i:] for i in range(5)])  # N-Gram length is 5
            return [''.join(ngram) for ngram in ngrams]


        def find_group(row, col):
            # If either the row or the col string have already been given
            # a group, return that group. Otherwise return none
            if row in group_lookup:
                return group_lookup[row]
            elif col in group_lookup:
                return group_lookup[col]
            else:
                return None


        def add_vals_to_lookup(group, row, col):
            # Once we know the group name, set it as the value
            # for both strings in the group_lookup
            group_lookup[row] = group
            group_lookup[col] = group


        def add_pair_to_lookup(row, col):
            # in this function we'll add both the row and the col to the lookup
            group = find_group(row, col)  # first, see if one has already been added
            if group is not None:
                # if we already know the group, make sure both row and col are in lookup
                add_vals_to_lookup(group, row, col)
            else:
                # if we get here, we need to add a new group.
                # The name is arbitrary, so just make it the row
                add_vals_to_lookup(row, row, col)


        # Construct your vectorizer for building the TF-IDF matrix
        vectorizer = TfidfVectorizer(analyzer=ngrams_analyzer)

        # Grab the column you'd like to group, filter out duplicate values
        # and make sure the values are Unicode
        vals = df['story_urlA'].unique().astype('U')

        # Build the matrix!!!
        tfidf_matrix = vectorizer.fit_transform(vals)

        cosine_matrix = awesome_cossim_topn(tfidf_matrix, tfidf_matrix.transpose(), vals.size, 0.8)
```

Fig. 4: Kickstarter Analysis

This output is a .csv file that was generated for both the Story and Risk sections of the data with similarity scores as two separate columns.

	urla	urlA	story_urlA	risk_urlA	urlb	urlB	story_urlB	risk_urlB	cs_simStory	cs_simRisk
0	https://www.kickstarter.co	921574158/500mics-dis	Many independent album	Risks involving a goa	https://www.kickstarte	921574158/500mics-pa	500MICS by Independe	Risks involving a goal su	0.119311752	0.938971068
1	https://www.kickstarter.co	1117100067/spores-a-h	I have been making movie	As with any project t	https://www.kickstarte	1117100067/one-freak	What can you do with a	The film is funded as of	0.164597705	0.131558703
2	https://www.kickstarter.co	delwin/enceladuss-first-a	To hear more of my music (including previews a		https://www.kickstarte	delwin/capyacs-full-len	In May of last year, we t	This baby is coming out	0.148285713	0
3	https://www.kickstarter.co	1486860504/single-mur	Being a single mother of tv	Traveling always con	https://www.kickstarte	1486860504/brussels-s	Brussels Smiles : 100 da	This project will take life	0.090010287	0.110818328
4	https://www.kickstarter.co	luckycharmsss/the-myst	YOU GUYS I DID A THING A	Finding a machine fo	https://www.kickstarte	luckycharmsss/the-dum	I have copy/pasted my	if i break both my arms	0.137583471	0
5	https://www.kickstarter.co	655348447/city-coordin	Hello everyone,Northshire	All of posters are alre	https://www.kickstarte	655348447/20-oldscho	Northshire is happy to a	All of posters are alread	0.193849529	1
6	https://www.kickstarter.co	43425144/closed-systen	Go to my website to read a	It may not work. This	https://www.kickstarte	43425144/spirituality-fi	The video above is one	There are no real risks,	0.066831888	0.16495722
7	https://www.kickstarter.co	2130842041/need-coffe	I started roasting coffee la	One of the biggest ch	https://www.kickstarte	2130842041/coffee-snc	To raise awareness of m	There is no risk or challe	0.195837363	0.099449032
8	https://www.kickstarter.co	1968016606/eat-the-mc	On a whim, chucking it all (	I am not expecting a	https://www.kickstarte	1968016606/through-h	My daughters love art, v	The only risks are the fu	0.242510183	0.181071492
9	https://www.kickstarter.co	jo1/my-pig-penyata	I like to read about people	I may not make a pro	https://www.kickstarte	jo1/the-ellen-degeneres	I would like to make Elle	Ellen may not get my ca	0.140391847	0.193649167

Fig. 5: Kickstarter Results

2.3. Reddit Project - Pre-Processing

This script generates the BOW and TF-IDF output which is then followed by LDA with BOW and LDA with TF-IDF on the corpus for post content column from the reddit dataset.

Gensim doc2bow

```
In [11]: bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
         bow_corpus[4]

Out[11]: [(26, 1), (34, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1), (45, 1)]
```

```
In [12]: bow_doc_4 = bow_corpus[4]
         for i in range(len(bow_doc_4)):
             print("Word {} (\"{}\") appears {} time.".format(bow_doc_4[i][0],
                                                       dictionary[bow_doc_4[i][0]],
         bow_doc_4[i][1]))

         Word 26 ("long") appears 1 time.
         Word 34 ("posit") appears 1 time.
         Word 40 ("need") appears 1 time.
         Word 41 ("recoveri") appears 1 time.
         Word 42 ("safe") appears 1 time.
         Word 43 ("scar") appears 1 time.
         Word 44 ("speedi") appears 1 time.
         Word 45 ("wish") appears 1 time.
```

TF-IDF

```
In [13]: from gensim import corpora, models
         tfidf = models.TfidfModel(bow_corpus)
         corpus_tfidf = tfidf[bow_corpus]
         from pprint import pprint
         for doc in corpus_tfidf:
             pprint(doc)
             break

         [(0, 0.29952230827039616),
          (1, 0.21299857368565608),
          (2, 0.18070336398167702),
          (3, 0.08893386976084357),
          (4, 0.19854932216994936),
          (5, 0.30058642538983177),
          (6, 0.3676221123492878),
          (7, 0.46620073546823254),
          (8, 0.2805737356475635),
          (9, 0.1324365446187328),
          (10, 0.3470843989810606),
          (11, 0.2713600210339164),
          (12, 0.1972323962427399),
          (13, 0.11325352605661813)]
```

LDA using BOW

```
In [14]: lda_model = gensim.models.LdaMulticore(bow_corpus, num_topics=10, id2word=dictionary, passes=2, workers=2)
```

```
In [15]: for idx, topic in lda_model.print_topics(-1):
             print('Topic: {} \nWords: {}'.format(idx, topic))

         Topic: 0
         Words: 0.058*"time" + 0.032*"thank" + 0.030*"hand" + 0.030*"post" + 0.030*"question" + 0.030*"rule" + 0.030*"kind" +
         0.030*"read" + 0.029*"comment" + 0.029*"concern"
         Topic: 1
         Words: 0.032*"test" + 0.019*"virus" + 0.017*"posit" + 0.013*"take" + 0.012*"time" + 0.012*"sure" + 0.011*"infect" +
         0.011*"know" + 0.011*"negat" + 0.011*"go"
         Topic: 2
         Words: 0.058*"https" + 0.044*"covid" + 0.043*"remov" + 0.026*"reddit" + 0.024*"post" + 0.017*"posit" + 0.016*"articl"
         + 0.014*"comment" + 0.013*"messag" + 0.011*"question"
         Topic: 3
         Words: 0.041*"smell" + 0.025*"test" + 0.022*"tast" + 0.019*"covid" + 0.018*"posit" + 0.016*"symptom" + 0.014*"like" +
         0.012*"day" + 0.012*"lose" + 0.011*"thing"
         Topic: 4
         Words: 0.046*"test" + 0.041*"covid" + 0.017*"symptom" + 0.016*"posit" + 0.013*"antibodi" + 0.011*"negat" + 0.011*"doc
         tor" + 0.011*"vaccin" + 0.010*"result" + 0.009*"get"
         Topic: 5
         Words: 0.043*"test" + 0.038*"symptom" + 0.025*"fever" + 0.023*"day" + 0.023*"cough" + 0.020*"feel" + 0.018*"posit" +
         0.016*"throat" + 0.013*"headach" + 0.012*"sore"
         Topic: 6
```

Fig. 6: Reddit Analysis

2.4. Reddit Project - Topic Modelling

The output of the analysis is stored in a .csv file for both LDA with BOW and LDA with TF-IDF which can further be analyzed using similarity comparison techniques.

	post_content	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	Topic10
0	Hey, so IÃ¢â‚¬â„¢m experiencing a burning sensation r	0	0	0	0	0	0.20325	0.743399	0	0	0
1	Sounds like a panic attack. I had the same thing towarc	0.010003	0.01001	0.010003	0.010008	0.010006	0.010007	0.909944	0.010008	0.010005	0.010006
2	I really hope so. Every time I think I have it, idk why, bu	0.011115	0.011114	0.011113	0.011114	0.011114	0.011114	0.011114	0.463953	0.011115	0.447134
3	I tested POSITIVE for COVID19. I had something similar	0	0	0.182622	0	0	0.760206	0	0	0	0
4	Wow, I really wish you a safe and speedy recovery. Ho	0.011115	0.011116	0.011115	0.011114	0.011115	0.011114	0.011113	0.011113	0.899972	0.011114
5	Thank you for your submission!	0.974285	0	0	0	0	0	0	0	0	0
6	It might be combo of panic attack and gerd, if u panic	0	0	0	0	0	0	0.834983	0	0.147616	0
7	I have anxiety so it that might be the case. IÃ¢â‚¬â„¢ve	0	0	0	0	0	0	0	0	0.924974	0
8	17ftm, occasional smoker, non drinker, no relevant	0	0	0	0	0.090598	0.671913	0.221199	0	0	0
10	itÃ¢â‚¬â„¢s also been an oddly long time for me - almc	0	0.314466	0	0	0	0.612784	0	0	0	0

Fig. 7: Reddit Results

3. CONCLUSION | Beyond Learning

Overall, this Independent study was a very good learning experience. I have learnt a lot about web-scraping and advanced python. Since Kickstarter and reddit are some of the most used sites for funding projects and sharing ideas, it's very interesting to scrape the data from these sites and analyses the data to find new insights. I have also learned how to use some of the advanced libraries in Python for data preprocessing. I learned about Topic Modelling and specifically about Latent Dirichlet Allocation (LDA), Term Frequency - Inverse Document Frequency (TF-IDF), Cosine Similarity and how these techniques are used in predictive text analytics. The next steps in the project are to apply other similarity techniques and more advanced text analytics tools like LSA, SVD etc. and see if we can improve upon our findings and find out new ways to process textual data.

4. REFERENCES | Credits

Following is the list of all the references:

- https://www.kickstarter.com/
- https://www.reddit.com/
- https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

---------------------------------[End]---------------------------------