**Capstone Project Report**

**Career Village Recommender System**

**Course Number : CKME136**

**Session : Winter 2019**

**Student Name : Mona David**

**Mail Id :** [**mona.david@ryerson.ca**](mailto:mona.david@ryerson.ca)

**Ryerson University – G Chang School of Continuing Education**

**GitHub Link :** <https://github.com/MonaDavid/CKME-136-Capstone>

TABLE OF CONTENTS

**1.Introduction………………………………………………………………………………………3**

**2.Literature Review………………………………………………………………………………...3**

**3.Dataset……………………………………………………………………………………………..6**

**4.Approach………………………………………………………………………………………….11**

**5.Data Cleaning and Pre-Processing………………………………………………………………12**

**6.Exploratory Data Analysis……………………………………………………………………….13**

**7.Modelling………………………………………………………………………………………….25**

**8.Conclusions……………………………………………………………………………………….29**

**9.References………………………………………………………………………………………..30**

**Appendix**

**1.Introduction**

**Problem Statement:** The U.S. has almost 500 students for every guidance counselor. Underprivileged youth lack the network to find their career role models, making CareerVillage.org the only option for millions of young people in America and around the globe with nowhere else to turn. CareerVillage.org is a non-profit that crowdsources career advice for youth. The platform has now served career advice from 25,000 volunteer professionals to over 3.5M online learners. The platform uses a Q&A style similar to StackOverflow or Quora to provide students with answers to any question about any career.

**Business Objective: Develop a method to recommend relevant questions to the professionals who are most likely to answer them so that, the right questions reach the professionals and they are more likely answer them.**

**2.Literature Review**

**1. DiRenzo, Marco S., Christy H. Weer, and Frank Linnehan. "Protégé career aspirations: The influence of formal e-mentor networks and family-based role models." *Journal of Vocational Behavior* 83.1 (2013): 41-50.**

To understand the influence of e-mentoring on underprivileged youth I reviewed this study on Protégé Career Aspirations [3]. The study is the data from a 9 month e-mentoring program conducted by icouldbe.org(ICB) which is not for profit organization that creates and manages on-line adult-youth mentoring programs. ICB matches adult mentors with 10,000 students from schools internationally. This study indicated that career aspirations increased as a result of the mentoring process and specifically as a result of the increases in both general and career-based self-efficacy.

**2. Liddy, Elizabeth D. "Natural language processing." (2001).**

To better tackle our objective for the project I researched on how to process the natural language to interpret the questions and answers authored by the students and professionals. This paper by Elizabeth D. Liddy [6] lays the foundation for better understanding what Natural Language Processing (NLP) is and how its main purpose is Natural Language Understanding (NLU). This paper begins with defining NLP and briefs it’s history. The levels of NLP and different approaches to NLP are elucidated. Information Retrieval and Information Extraction are mentioned as the most frequent applications that use NLP.

**3. Fox, Christopher. "A stop list for general text." *Acm sigir forum*. Vol. 24. No. 1-2. New York, NY, USA: ACM, 1989.**

The first step in Information Retrieval to isolate the “English Stop Words” i.e. the most frequently occurring words in English. I came across this paper by Christopher Fox from AT&T Bell Laboratories [4], where an assignment was conducted to generate a list of “Stop Words” from the Brown corpus. The Brown corpus was compiled in 1960 by Henry Kucera and W Nelson Francis at the Brown University, it consists of a million and fourteen words. As a result, we have 421 stop words which are highly effective in filtering the most frequently occurring and semantically neutral words in general English literature. The words have been attached in the appendix of the paper and raked for their frequency of occurrence.

**4. Balakrishnan, Vimala, and Ethel Lloyd-Yemoh. "Stemming and lemmatization: a comparison of retrieval performances." (2014): 174-179.**

The next step in the language was to lemmatize the words ,i.e. to return the word to its base form or dictionary form .I found this article by Vimala Balakrishnan, and Ethel Lloyd-Yemoh[1] on lemmatization which throws light on the importance of language modeling techniques and also compares the performance of Stemming and Lemmatization. The main questions of research in the paper were i) To compare the document retrievals using stemming and lemmatization and ii) compare the stemming and lemmatization techniques against a baseline ranking algorithm (i.e. no language processing). In this experiment the Porter’s stemming algorithm for stemming, LemmaGen for lemmatization and tf-idf was used for the baseline ranking algorithm. The retrieval performances were tested and comparisons were made using the Communications of the Association for Computing Machinery (CACM) collection. The MAP (Mean Average Precision) was calculated by dividing the average precisions with the number of queries. As a result, the study found that the language processing techniques improved the document retrieval and Lemmatization performed better compared to Stemming.

**5. Jain, Abhishek, et al. "Information retrieval using cosine and jaccard similarity measures in vector space model." *International Journal of Computer Applications* 164.6 (2017): 28-30.**

After the basic data processing techniques, now we get into the real deal of using similarity measure for Information Retrieval .This paper by 4 college students from Bharati Vidyapeeth’s college of engineering published in International Journal of Computer Applications[5] lays foundational background for better understanding of the Vector Space Model along with the TF/IDF weighting scheme, also the Cosine and Jaccard similarity measures with TF/IDF scheme are elucidated .It also introduces us to the another improved term weighting scheme TF-ATO, Term Frequency with Average Term Occurrences. The paper states the advantages of Vector Space Model over the Boolean Model and also describes the limitations of Vector Space Model.

**6.Cavnar, William B., and John M. Trenkle. "N-gram-based text categorization." *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*. Vol. 161175. 1994.**

Statistical Language models such as N-gram model has been dominating the language modeling field for almost three decades due to their simplicity and low computational complexity. The paper by William B. Cavnar and John M. Trenkle explains N-Gram Based Text Categorization [2]. It starts with an introduction to N-gram based similarity measures and takes use through the N-gram frequency statistics. They also test the N-gram based text categorization on a language classification task and discuss the advantages of N-gram categorization over other possible approaches. The key benefit of N-gram is, since every string is decomposed into small parts, any errors that are present tend to affect only a limited number of those parts, leaving the remainder intact. If we count N-grams that are common to two strings, we get a measure of their similarity that is resistant to a wide variety of textual errors. The authors collected 3713 language samples from the soc.culture newsgroup hierarchy of the Usenet to test the N-gram based text categorization. The results were the system yielded its best performance at a profile length of 400 N-grams. The system misclassified only 7 articles out of 3478, yielding an overall classification rate of 99.8%.

**7. Mikolov, Tomáš, et al. "Strategies for training large scale neural network language models." *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2011**

In recent times Neural network-based language modeling is gaining popularity. The authors Tomas Mikolov and colleagues describe how to effectively train neural network-based language models on large datasets [7]. For their study they use the Recurrent Neural Network language model (RNN LM). They performed recognition on the English Broadcast News (BN) NIST RT04 task provided to them by IBM containing 430 hours of audio which was roughly 400M tokens. Using the concept of ‘incremental learning’ that is learning simple patterns in the data before complex patterns can be learnt, the approach they took was to begin the training with out-of -domain data and end with the most important in-domain data. The study shows that training RNN model with direct connections can lead to good performance both on perplexity and word error rate, even if very small hidden layers are used.

**8. Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009).**

I also wanted to look into Collaborative Filtering for Recommendation systems and that’s when I stumbled upon this survey by Xiaoyuan Su and Taghi M. Khoshgoftaar [8].In this paper they list out the main challenges of collaborative filtering such as data sparsity, scalability, synonymy, gray sheep, shilling attacks, privacy protection, etc., and their possible solutions. They present the 3 main categories of CF techniques which are memory-based, model based, and hybrid CF algorithms. To evaluate the prediction performance of CF, MAE (Mean Absolute Error) is used. Precision and recall, ROC sensitivity is used as a decision support accuracy metric.

**9. Yong, Zhou, Li Youwen, and Xia Shixiong. "An improved KNN text classification algorithm based on clustering." *Journal of computers* 4.3 (2009): 230-237.**

This [9] paper of improved K-NN for Text classification based on clustering interested me due to the fact that it had enhanced results than the traditional KNN text classification algorithms. The researchers used an algorithm that doesn’t use all the training samples and got rid of the uneven distribution that causes multipeak effect. Afterwards it trained all the categories by K-mean clustering to get the cluster centers and they are used as new training samples and new weights were introduced to each cluster center and this was used in the algorithm. The training corpus had 20 categories with 9804 documents and the test set had 9833 documents. The experiments confirmed the effectiveness of this algorithm.

**3.Dataset**

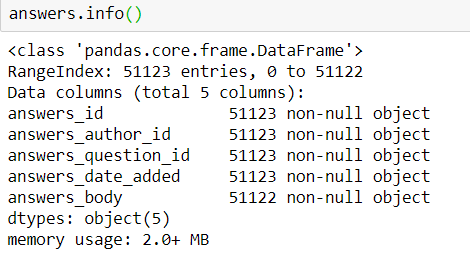
The source of Data is from a Kaggle competition introduced by Career Village to develop a better Recommendation System so that professionals receive relevant questions and more questions are answered by professionals.

Kaggle Link: <https://www.kaggle.com/c/data-science-for-good-careervillage/data>

The Data is in CSV format and will be imported to our Jupyter as CSV files.

**Data Description:**

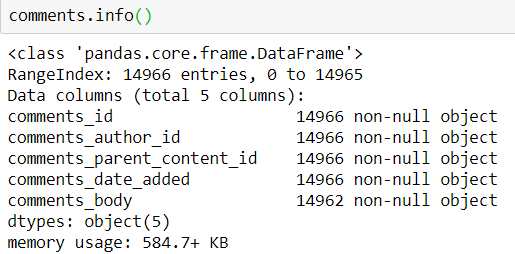
**Answers.csv** : Answers are posted in response to a question .A single question can have many answers and they can be only posted by Professionals. Below are the columns in answers.csv .



**Fig 3.1 Screenshot of columns in Answers.csv.**

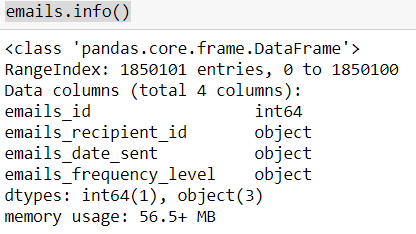
The columns answers\_author\_id is the professionals\_id and will be changed to the same and answers\_question\_id is the questions\_id and will be changed to the same.

**Comments.csv** : Comments can be made on Questions and/or Answers. If a comment is posted on an existing comment ,the existing comment is referred to as the “parent” of that comment. Below are the columns in comments.csv.



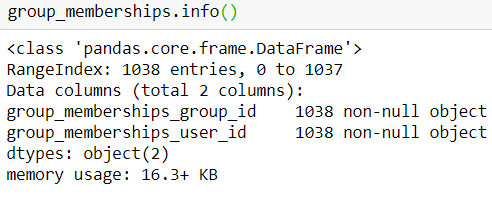
**Fig 3.2 Screenshot of columns in Comments.csv**

**Emails.csv :** Each email corresponds to one specific email being sent to one specific recipient. The frequency\_level refers to the type of email template which includes immediate emails sent right after a question is asked, daily digests, and weekly digests.Below are the columns in Emails.csv.



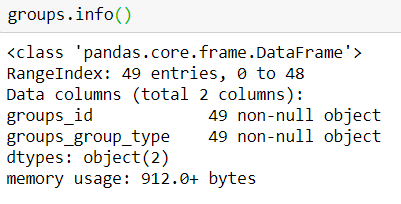
**Fig 3.3 Screenshot of columns in emails.csv**

**Group\_memberships.csv :** Any user can join any group. Below are the columns in the group\_memberships.csv.



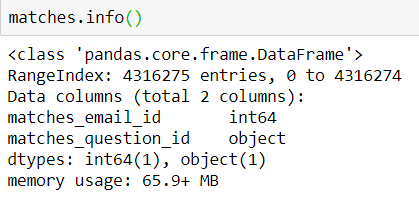
**Fig 3.4 Screenshot of columns in group\_memberships.csv**

**Groups.csv:** Each group is assigned to a type. Below are the columns in groups.csv.



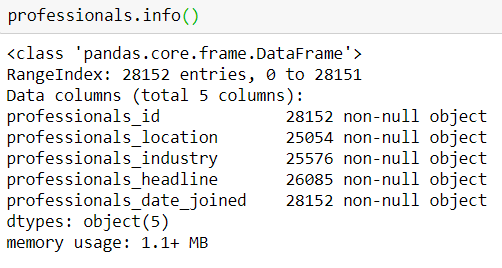
**Fig 3.5 Screenshot of columns in groups.csv**

**Matches.csv** : Each row tells us which questions are included in an email. Each email contains at least 1 question. Below are the columns in matches.csv.



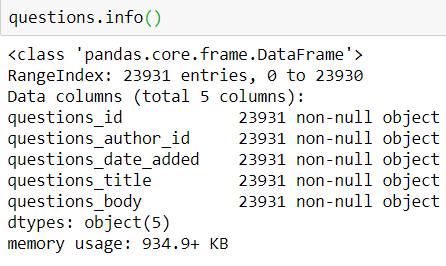
**Fig 3.6 Screenshot of columns in matches.csv**

**Professionals.csv :** They are the professionals who volunteer their time and energy into answering the questions posed by students. Below are the columns in professionals.csv.



**Fig 3.7 Screenshot of columns in professionals.csv**

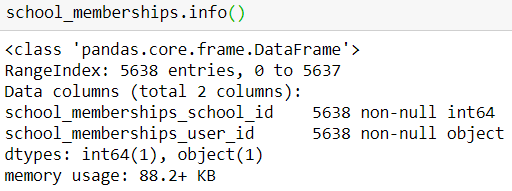
**Questions.csv :** Questions are posted by the students. Below are the columns in questions.csv.



**Fig 3.8 Screenshot of columns in questions.csv**

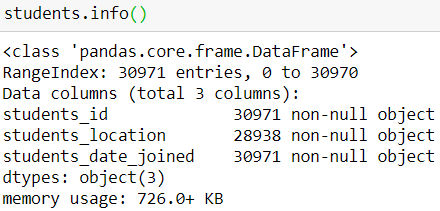
The columns questions\_author\_id will be changed to students\_id.

**School\_memberships.csv** : This is similar to group memberships but it’s for schools instead. Below are the columns in school\_memberships.csv.



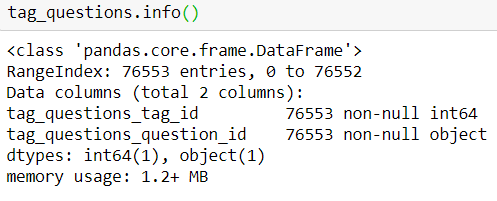
**Fig 3.9 Screenshot of columns in school\_memberships.csv**

**Students.csv** : This file contains the information about the students and their locations. Below are the columns in students.csv.



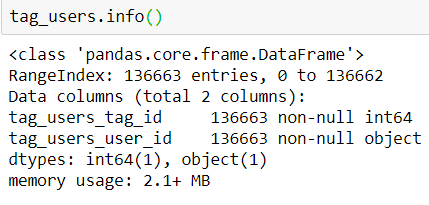
**Fig 3.10 Screenshot of columns in students.csv**

**Tag\_questions.csv:** This file contains the questions along with the tags to which they are tagged to. Below are the columns in tag\_questions.csv



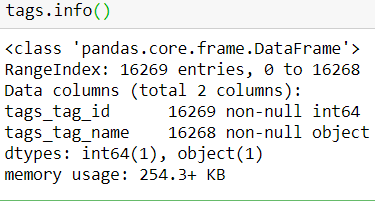
**Fig 3.11 Screenshot of columns in tag\_questions.csv**

**Tag\_users.csv:** This file has the information about the tags any user follows.Below are the columns present in tag\_users.csv.



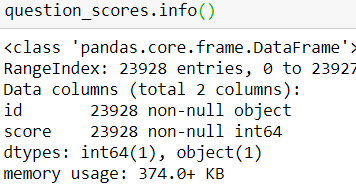
**Fig 3.12 Screenshot of columns in tag\_users.csv**

**Tags.csv:** This file contains the tag names and their ids. Below are the columns present in tags.csv.



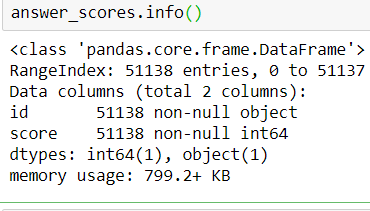
**Fig 3.13 Screenshot of columns in tags.csv**

**Question\_scores.csv:** "Hearts" scores for each question.Below are the columns present in question\_scores.csv.



**Fig 3.14 Screenshot of columns in questions\_scores.**

**Answer\_scores.csv:** "Hearts" scores for each answer.Below are the columns present in answers\_scores.csv.



**Fig 3.15 Screenshot of columns in answer\_scores**

**4.Approach**

Load Data

Clean & Preprocess

Exploratory data analysis

Modelling

Conclusion

Train & Test Data

Model 1

Model 2

**5. Data Cleaning and Pre-Processing**

* We import the dataset from Kaggle and name the dataframes as same as the CSV file names.
* We look for missing values in columns from each of the above dataframes .
* The dataframes Emails,Group\_Memberships,Groups,Matches,Questions,School\_Mamberships,Tag\_Questions,Tag\_Users,Question\_Scores,Answer\_Scores have no missing values.
* Since all the data is either string or date type objects no imputation is done for the missing values.
* The column names are changed accordingly in respective dataframes.

Answers

Answers\_author\_id 🡪 professionals\_id

Answers\_question\_id 🡪 questions\_id

Group\_Memberships

Group\_memberships\_group\_id🡪groups\_id

Group\_memberships\_user\_id 🡪 user\_id

Groups

Groups\_group\_type 🡪 groups\_type

Matches

Matches\_email\_id 🡪 emails\_id

Matches\_question\_id 🡪 questions\_id

Questions

Questions\_author\_id 🡪 students\_id

School\_Memberships

School\_memberships\_school\_id 🡪 school\_id

School\_memberships\_user\_id 🡪 user\_id

Tag\_questions

Tag\_questions\_tag\_id 🡪 tags\_id

Tag\_questions\_question\_id 🡪 questions\_id

Tag\_Users

Tag\_users\_tag\_id 🡪 tags\_id

Tag\_users\_user\_id 🡪 user\_id

Tags

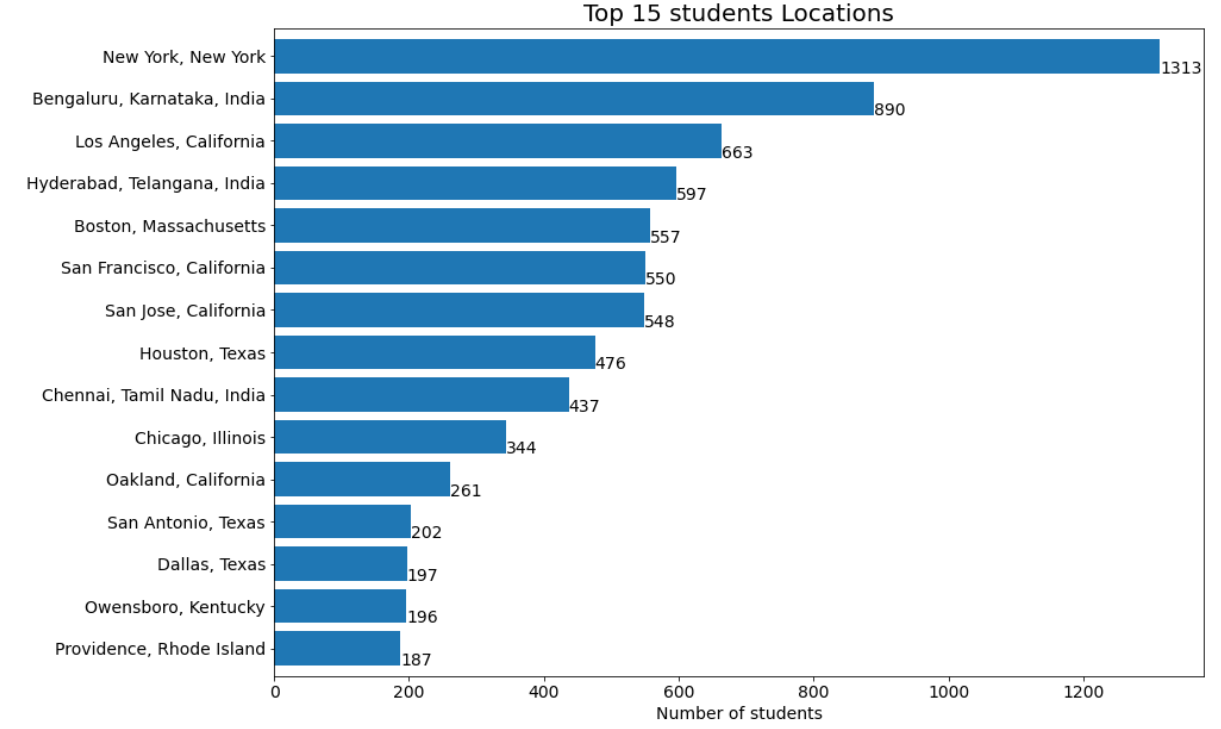
Tags\_tag\_id 🡪 tags\_id

Tags\_tag\_name 🡪 tag\_name

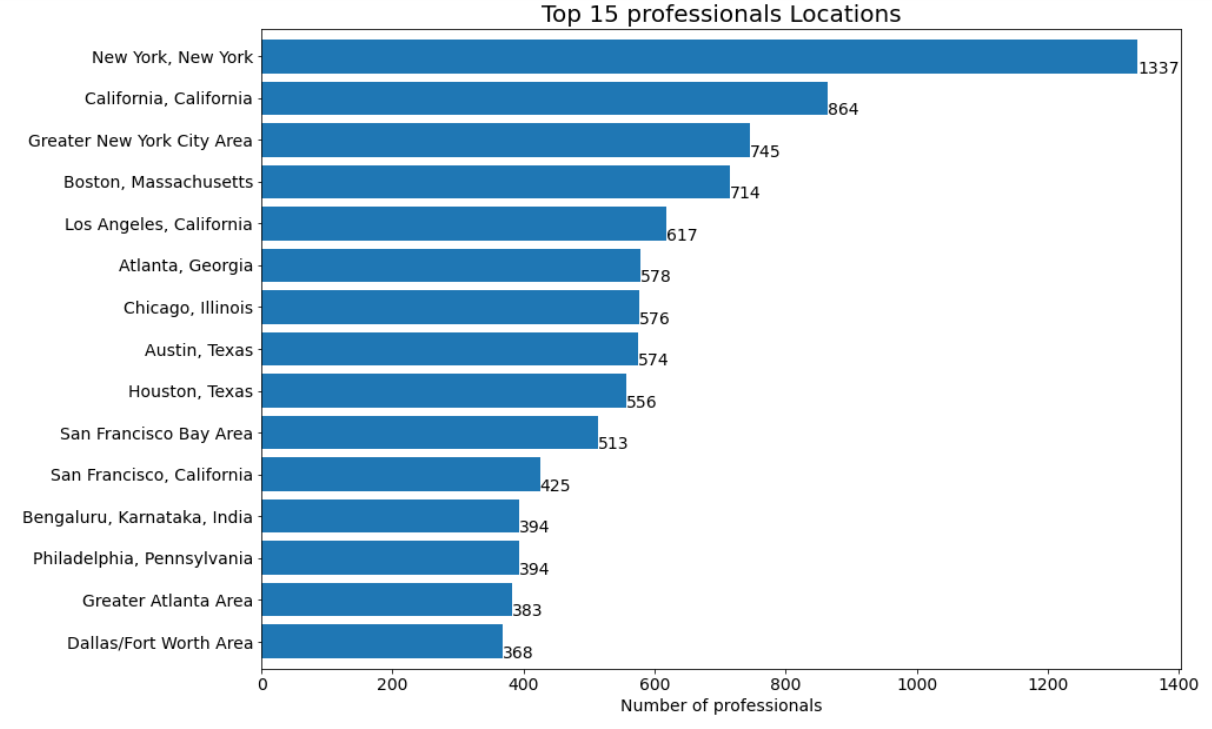
* The Datetime objects in the dataframes that were imported as strings are converted into Python datetime objects.

**6.Exploratory Data Analysis**

* We explore to see if there is any location-based correlation between students and professionals in terms of answered questions.
* We see that the highest Number of students are from New York City and the next highest number is from Bengaluru India.
* The highest Number of Professionals are from New York as well and the next highest number is from California.
* We have the top 15 locations from the students and the professionals.



**Fig 6.1 Graph of the top 15 locations Students are From**

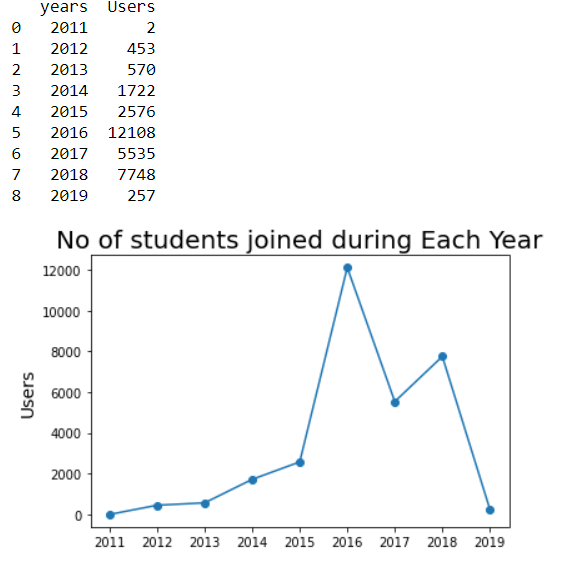


**Fig 6.2 Graph of the top 15 locations Professionals are From**

* We look at the earliest and recent dates the students have joined, the No of students joined in each year and the steady growth of students from 2011 to 2019.

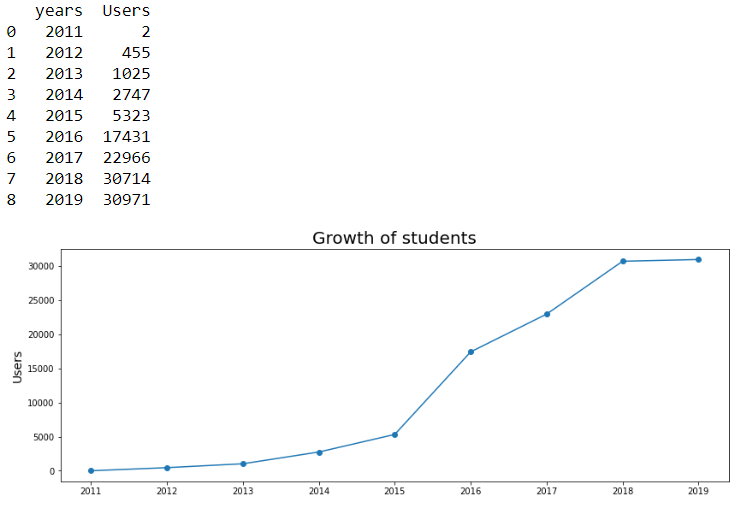


**Fig 6.3 Screenshot of Earliest Date and Recent Date the students have joined.**



**Fig 6.4 Dataframe of the Students joined during each year and graph representing the same.**

The above figure shows that maximum No Of students have joined in the year 2016 and it has gradually declined in the year 2019.



**Fig 6.5 Dataframe of the Students growth over the years and the graph representing the same.**

From the above figure we see that the student’s enrollment has grown over years steadily.

* We look at the earliest and recent dates the professionals have joined, the No of professionals joined in each year and the steady growth of students from 2011 to 2019.

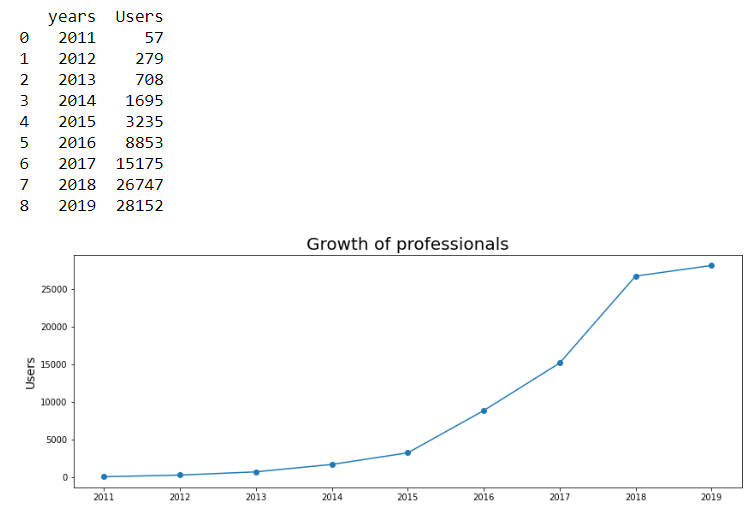


**Fig 6.6 Screenshot of Earliest and Recent dates the professionals have joined.**



**Fig 6.7 Dataframe of the professionals joined in each year and graph representing the same.**

We see that the maximum number of professionals have joined in the year 2018 and the numbers have steeply fallen in 2019.



**Fig 6.8 Dataframe representing the growth of professionals over the years and graph representing the same.**

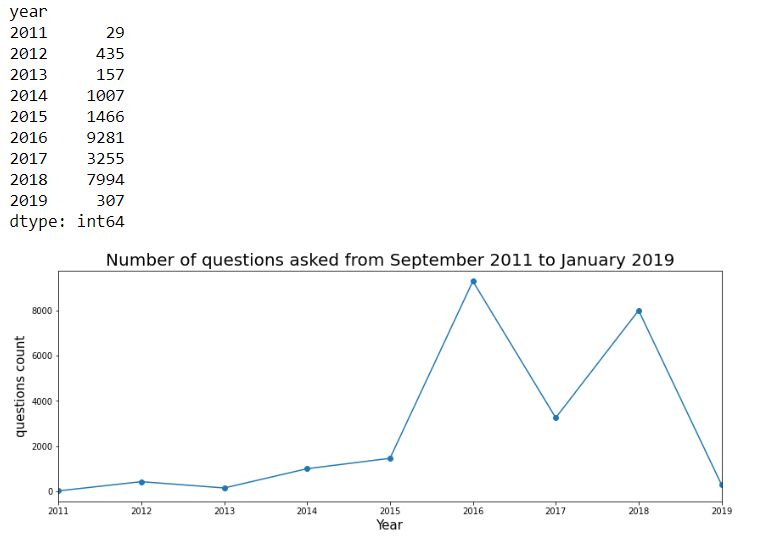
We see that the professional’s enrollment over the years have grown over the years and has been increasing since the year 2015.

* We look at the timeline of the questions added to the site by students .

In the below screenshot we see the earliest date and the most recent dates questions have been added to the site.



**Fig 6.9 Screenshot of the Earliest and the most recent dates the questions where added.**



**Fig 6.10 Dataframe containing the Year and No of Questions asked in a Year and graph representing the same**.

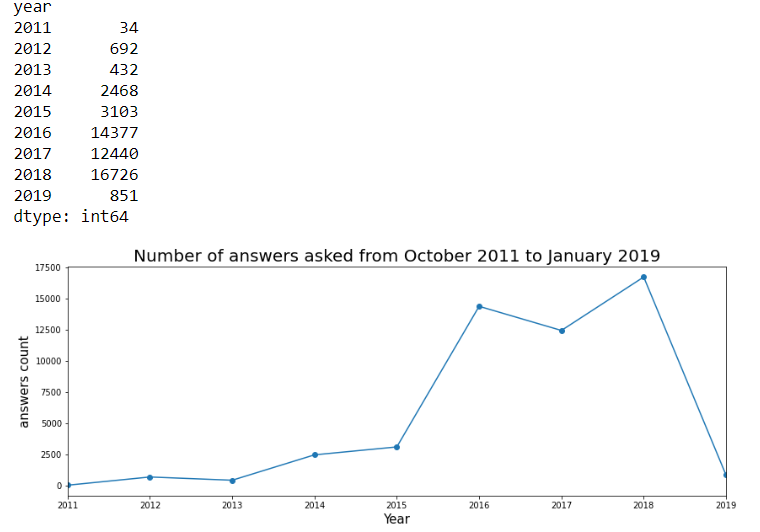
From the above figure we see that the maximum questions were asked in the year 2016 which also happened to be the year maximum students enrolled in the site.

Observations from the previous graph of student’s enrollment in the site shows that the figures of No of Questions asked and No of Students Enrolled is almost similar and are co-related .

* We look at the timeline of answers posted by the professionals .In the below screenshot we see the earliest date and the most recent date the professionals answered the questions.



**Fig 6.11 Screenshot containing the most recent and earliest dates the answers were added.**

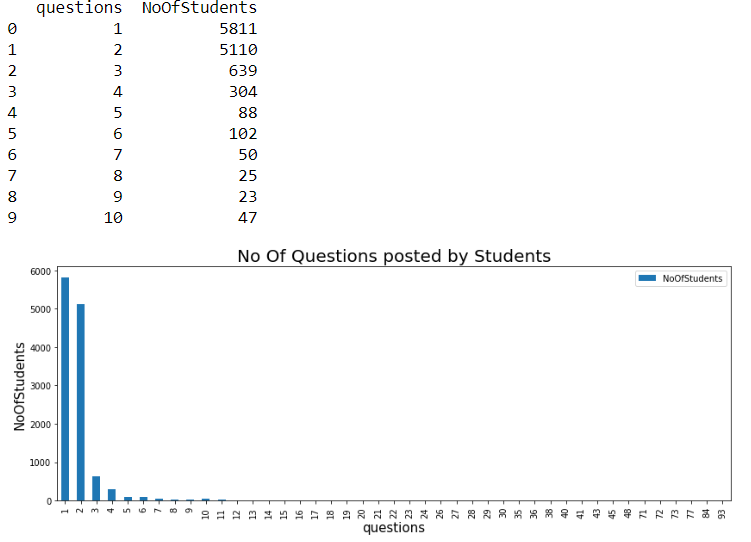


**Fig 6.12 Dataframe containing the Years and the No of Answers added in each year and graph representing the same.**

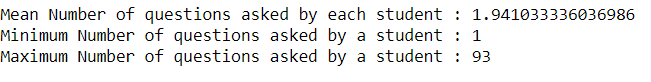
From the above figure we see that the maximum answers where added in the year 2018 which is the same year maximum no of professionals were enrolled in the site.

Comparing the graph of professional enrollment we find that even though No Of professionals were increased in the year 2017,the No Of answers provided dropped in the year and increased the next year.

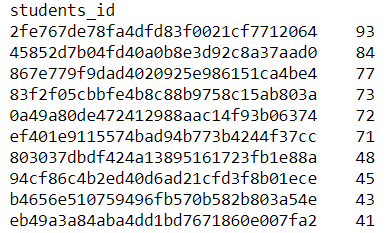
* We see that most number of students ask 1 or 2 questions and the number of students who as more than 2 questions are quite small and the number keeps decreasing as the number of questions raises. This can be seen from the below figure showing the dataframe containing the Number of questions and the numbers of students asking question.

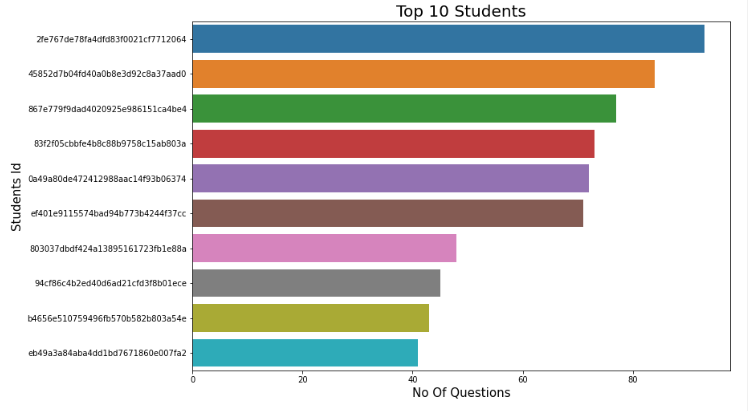


**Fig 6.13 Dataframe consisting of Questions and No of Students and bar graph representing the same.**



**Fig 6.14 Screenshot of mean, min and max No Of Questions asked by students**

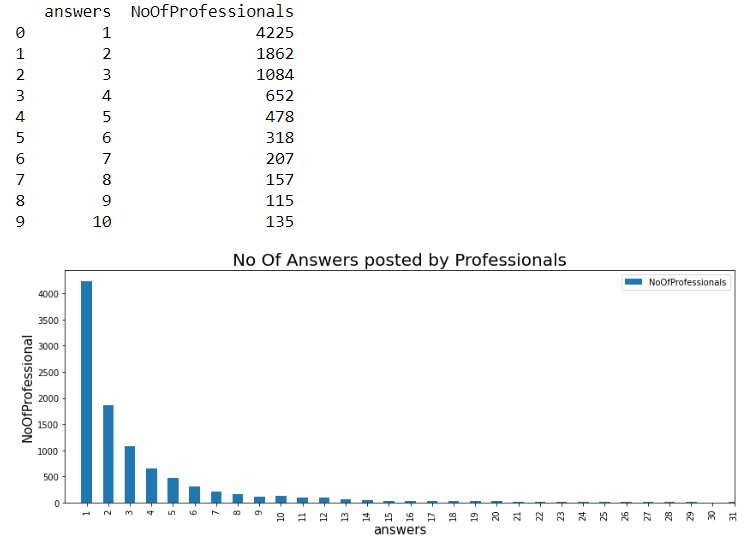




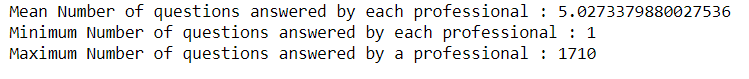
**Fig 6.14 Graph of Top 10 Students**

From the above graph we see that 6 students have asked more than 70 questions .

* A similar trend is observed in the below graph where professionals post the answers .Maximum number of professionals answer 1 or 2 or 3 questions.

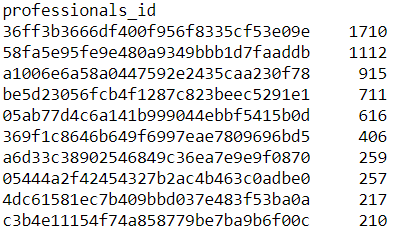


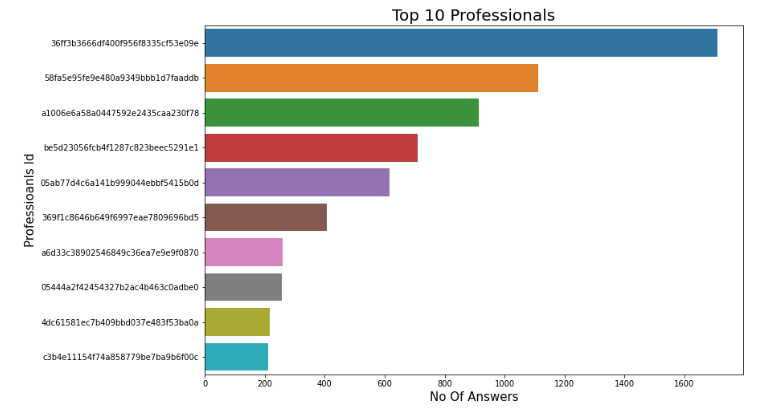
**Fig 6.15 Dataframe consisting of Answers and No of Professionals and bar graph representing the same.**



**Fig 6.16 Screenshot of mean, min and max No of Questions answered by professionals**.

We look at the top 10 professionals who have answered the most number of questions .





**Fig 6.17 Graph of top 10 Professionals**

From the above graph we see that 2 professionals have answered more than 1000 questions and they account for almost 4% of the total number of answers posted.

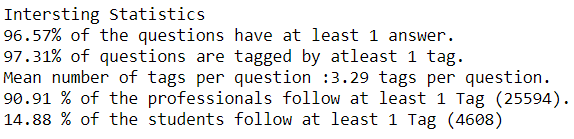
* Next, we calculated the ratio of questions that were answered within a single day or took more day to be answered.

We see from the below figure that most of the questions were answered within a single day or took more than 10 days to be answered.



**Fig 6.18 Graph showing the ratio of questions answered within certain Number Of Days.**

Some Interesting Statistics from our Dataset:

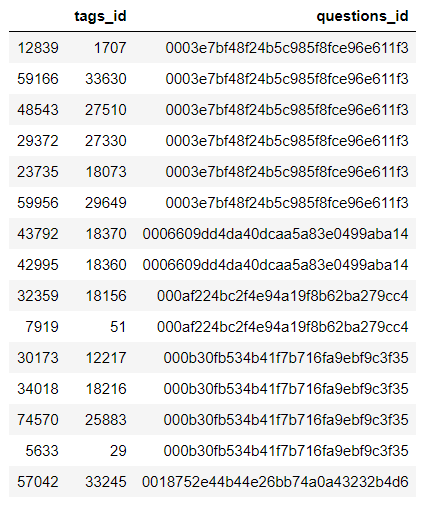


**7.Modelling**

We use the content-based recommender system. I have proposed 2 solutions.

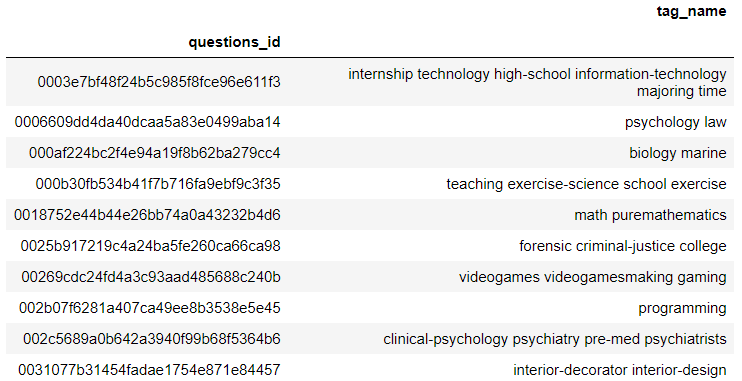
**Solution 1:** To identify if a current question matches any other previous questions and locate the authors who have previously answered such questions and send them the relevant questions.Below are the steps for our proposed solution.

Step 1:Identify tags associated with a question .One question can have multiple tags associated with it as shown in the below figure.



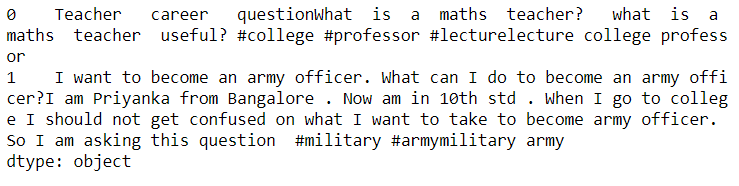
**Fig 7.1 Table showing the tags associated with a question.**

Step 2: Create a tag string of all the tags associated with a question as shown below.



**Fig 7.2 Tag String associated with Questions.**

Step 3:Create a questions corpus using the question’s title, question’s body and the tag strings .An example is shown below of the top 2 question corpus.



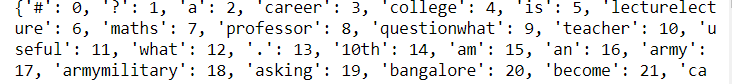
**Fig 7.3 Questions Corpus**

Step 4:Perform Text Processing

-We use the Gensim Package for the process of text processing.

-We tokenize the words in the questions corpus.

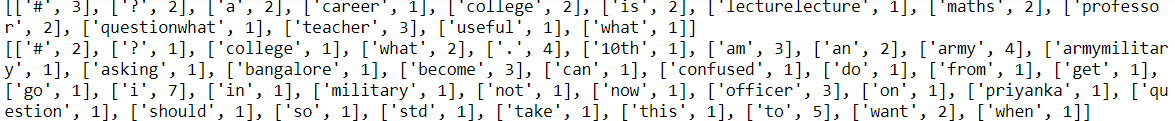
- A Gensim creates a dictionary that houses each word and assigns a unique token to the word. The below figure shows the top 20 words and their unique ids.



**Fig 7.4 20 Words and their unique ids**

- The dictionary object is used to create bag of words corpus. It is this dictionary and the bags of words corpus that are used as inputs to the models Gensim provides.

The below figure shows the corpus i.e. the word and the frequency with which the word occurs in the document.



**Fig 7.5 Words and their frequency**

Step 5: We create a TF-IDF Model for the corpus. Below figure shows the word and the assigned weight for the word. Words that occur more frequently have smaller weights compared to words that occur less frequently. For training purpose we use whole of the data.

Ex : 

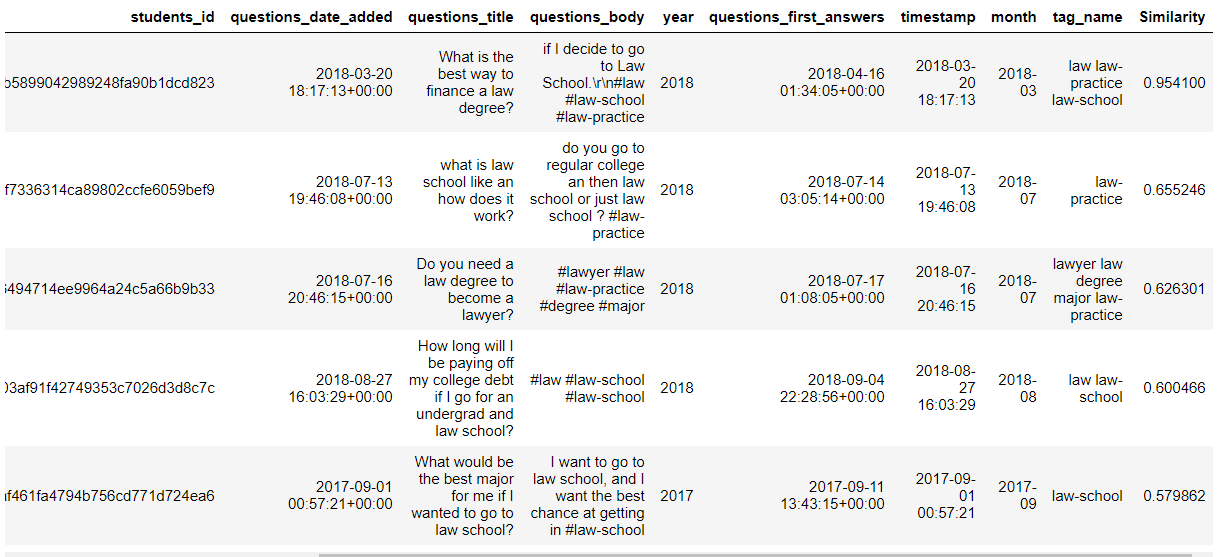
**Fig 7.6 words and their weight.**

In the above figure we see that the words ‘I ’ and ‘in’ have less weights and the word military has a greater weight.

Step 6:Create a similarity checker using genism. The similar object in genism is used to measure cosine similarity of a dynamic query against a static corpus of the documents.

Step 7:We take a Query='What is the best way to finance a law degree? #law #law-practice #law-school' to test tf-idf model.’ The sample query is then tokenized and tf-idf model is applied on the query corpus. The similarity matrix is applied on the query tfidf model.

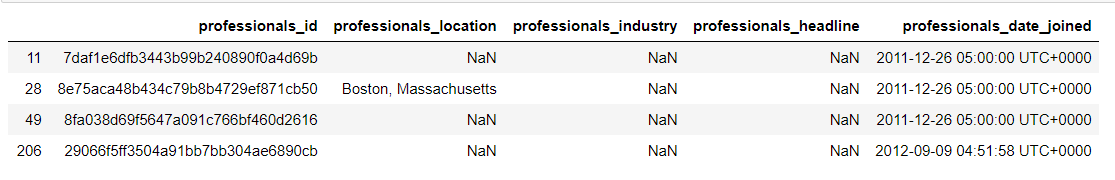
Step 8 : We set a similarity threshold of .10 and generate questions that have a similarity threshold of more than .10 .



**Fig 7.7 Similar questions are generated.**

The first recommendation is the same question with a similarity score of 0.95 and the next 2 recommendations have a similarity score of 0.655.

Step 9:Professionals who answered the previous questions are identified and relevant questions will be sent to them.



**Fig 7.8 Professionalids of the professionals who answered the previous questions.**

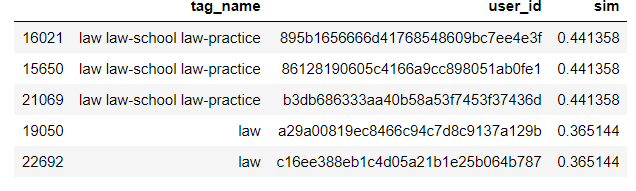
**Solution 2:**We identify professionals who follow particular tags and match them with tags associated with the query and send the questions to those professionals.

Step 1:Identiy the tags professionals follow and create a tag-string of all the tags the professionals follow. Create a word corpus of the same.

Step 2:Text Processing is done like in the previous model for the query and the word model.

The TF-IDF is generated for the tags corpus and applied on the query and similarity matrix is generated.

Step 3:The matrix with a cosine similarity for the tags is generated and shown as below.



**Fig 7.8 Recommender professionals who follow similar tags.**

**8.Conclusions**

**Model 1:** The model allows the students to look at all the past answers for a similar kind of questions and then lets them decide whether they wuld wat to post a new question os not.

It saves a lot of time for the professionals as this model will eliminate duplicate answers .

The issue with the model is that it does not take into consideration the professionals who haven’t answered before. This might leave a good number of authors without questions to answer.To tackle this issue we have the 2nd Model.

**Model 2:** The Career Village site mandates that each user registered under professional has to follow a minimum of 1 tag so ,each professional would have a tag which he/she follows hence ,they would receive a question that has a similar tag.

**9.References**

* Balakrishnan, Vimala, and Ethel Lloyd-Yemoh. "Stemming and lemmatization: a comparison of retrieval performances." (2014): 174-179.
* Cavnar, William B., and John M. Trenkle. "N-gram-based text categorization." *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*. Vol. 161175. 1994.
* DiRenzo, Marco S., Christy H. Weer, and Frank Linnehan. "Protégé career aspirations: The influence of formal e-mentor networks and family-based role models." *Journal of Vocational Behavior* 83.1 (2013): 41-50.
* Fox, Christopher. "A stop list for general text." *Acm sigir forum*. Vol. 24. No. 1-2. New York, NY, USA: ACM, 1989.
* Jain, Abhishek, et al. "Information retrieval using cosine and jaccard similarity measures in vector space model." *International Journal of Computer Applications* 164.6 (2017): 28-30.
* Liddy, Elizabeth D. "Natural language processing." (2001).
* Mikolov, Tomáš, et al. "Strategies for training large scale neural network language models." *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2011.
* Su, Xiaoyuan, and Taghi M. Khoshgoftaar. "A survey of collaborative filtering techniques." *Advances in artificial intelligence* 2009 (2009).
* Yong, Zhou, Li Youwen, and Xia Shixiong. "An improved KNN text classification algorithm based on clustering." *Journal of computers* 4.3 (2009): 230-237.

**Appendix :**

**Model-1:**

#Creating Tag-String

tagnames = pd.merge(tag\_questions,tags)

tagnames = tagnames.drop(['tags\_id'],axis = 1)

print(tagnames.sort\_values('questions\_id')[:15])

tagnames\_pivot = tagnames.pivot\_table(index = 'questions\_id',values = 'tag\_name',aggfunc = lambda x: " ".join(x))

tagnames\_pivot.head(10)

tagnames\_pivot["questions\_id"] = tagnames\_pivot.index

tagnames\_pivot['questions\_id']=tagnames\_pivot.index

print(f'Number of questions asked : {questions["questions\_id"].count()} ')

print(f'Number of questions with tags : {len(tagnames\_pivot)}')

tagnames\_pivot = tagnames\_pivot.reset\_index(drop = True)

#Combining questions with corresponding tags

questions\_tags = pd.merge(questions,tagnames\_pivot)

print(f'Number of questions with tags : {len(questions\_tags)}')

questions\_tags.head(3).T

#Combine QuestionTitle,Body and Tagnames

questions\_corpus = questions\_tags["questions\_title"] + questions\_tags["questions\_body"] + questions\_tags["tag\_name"]

questions["text"] = questions\_tags["questions\_title"] + questions\_tags["questions\_body"] + questions\_tags["tag\_name"]

questions\_corpus.head(2)

#Text Processing

#Tokenizing Data

gen\_docs = [[w.lower() for w in word\_tokenize(text)] for text in questions\_corpus]

#Creating Dictionary

dictionary = gensim.corpora.Dictionary(gen\_docs)

print(dictionary)

#To see the unique ids

print(dictionary.token2id)

#Creating Document-Term Matrix

#The words are replaced with their ids.

#The corpus contains the word id and the frequency of the word in each document

corpus = [dictionary.doc2bow(gen\_doc) for gen\_doc in gen\_docs]

# Show the Word Weights in Corpus

for doc in corpus:

print([[dictionary[id], freq] for id, freq in doc])

#Create TF-IDF Model

tfidf = gensim.models.TfidfModel(corpus)

# Show the TF-IDF weights

for doc in tfidf[corpus]:

print([[dictionary[id], np.around(freq, decimals=2)] for id, freq in doc])

#words that occur more frequently across the documents get smaller weights.

#Creating Similarity Checker

similarquestions = gensim.similarities.Similarity("",tfidf[corpus],num\_features=len(dictionary))

#Taking a question which is included in the training set

Query='What is the best way to finance a law degree?. #law #law-practice #law-school'

Query

query\_dict = [w.lower() for w in word\_tokenize(Query)]

query\_corpus = dictionary.doc2bow(query\_dict)

query\_tfidf = tfidf[query\_corpus]

query\_similar = similarquestions[query\_tfidf]

#Displaying the most similar questions from past

questions\_tags["Similarity"] = query\_similar

ques = questions\_tags[questions\_tags["Similarity"]>= similaritythreshold]

ques = ques.sort\_values("Similarity",ascending = False)

#Identify Professionals who answered These questions

questionslist = ques["questions\_id"]

questionslist\_answers = answers[answers["questions\_id"].isin(questionslist)]

#professionals who answered

answeredprofessionals = set(questionslist\_answers["professionals\_id"])

solution = professionals[professionals["professionals\_id"].isin(answeredprofessionals)]

**Model – 2**

#Identify Professionals with tags most similar to the questions asked.

user\_tags = pd.merge(tag\_users,tags)

user\_tags = user\_tags.drop(["tags\_id"],axis = 1)

user\_tags.sort\_values("user\_id")[:10]

#The first user has 3 tags.This means that one user can have many tags

#Concatenate all tags of a user to form a tag-string

tag\_pivot = user\_tags.pivot\_table(values = "tag\_name",index="user\_id",aggfunc = lambda x : " ".join(x))

tag\_pivot["user\_id"] = tag\_pivot.index

print("Number of all users with tags:",len(tag\_pivot))

tag\_pivot=tag\_pivot.reset\_index(drop=True)

tag\_pivot.head()

#Filtering only the professionals from all the users

prof\_tags= tag\_pivot[tag\_pivot['user\_id'].isin(professionals['professionals\_id'])]

print("Number of professionals with tags:",len(prof\_tags))

raw\_tags = prof\_tags["tag\_name"]

# #TextProcessing

gen\_docs = [[w.lower() for w in word\_tokenize(text)] for text in raw\_tags]

#Creating Dictionary

dictionary = gensim.corpora.Dictionary(gen\_docs)

#print(dictionary)

#To see the unique ids

#print(dictionary.token2id)

#Creating Document-Term Matrix

#The words are replaced with their ids.

#The corpus contains the word id and the frequency of the word in each document

corpus = [dictionary.doc2bow(gen\_doc) for gen\_doc in gen\_docs]

tf\_idf = gensim.models.TfidfModel(corpus)

sims = gensim.similarities.Similarity("",tf\_idf[corpus],num\_features=len(dictionary))

#For the same query

query\_doc = [w.lower() for w in word\_tokenize(Query)]

query\_doc\_bow = dictionary.doc2bow(query\_doc)

query\_doc\_tf\_idf = tf\_idf[query\_doc\_bow]

sim=sims[query\_doc\_tf\_idf]

prof\_tags['sim']=sim

prof\_tag=prof\_tags[prof\_tags['sim']>=0.10]

prof\_tag=prof\_tag.sort\_values('sim',ascending=False)

prof\_tag.head()