CS289 - Algorithmic Machine Learning
Final Project Report

Mauli Shah (004567942)
Nisarg Modi (604590567)

<u>Problem Statement:</u>

Quora Challenge, Labeler: https://www.quora.com/challenges#labeler

When a user adds a question on Quora, he/she is automatically suggested labels (or tags) to mark the question with. These labels can be thought of as categories in which the question falls. For example, a question about Dijkstra's algorithm would probably fit well under the topics "Algorithms" and "Graph Theory".

The goal is to develop a question topic labeler using machine learning/natural language processing techniques. We use Topic Modeling to solve this problem, as a continuation of the paper we presented in class.

Quora provides a dataset for training, along with an evaluation dataset that has correct answers that the model trained can be evaluated against. The challenge has a scoring mechanism to quantify the quality of the model trained. We will use the following to quantify our results:

Scoring

Your score for each question is determined as follows:

$$\frac{\sum_{i=0}^{9} \sqrt{10-i} \cdot (guess_i \in questionTopics)}{\sum_{i=0}^{min(|questionTopics|,10)-1} \sqrt{10-i}}$$

Your raw score is the sum of each question score.

$minScore$ = raw score for classifier that guesses 10 most frequent topics.

Your final score is $200 \cdot \frac{yourRawScore - minScore}{E - minScore}$.

Resource Limits

Your program is limited to 512 MB of memory and must run in 60 seconds or less.

Figure 1: Scoring Mechanism.

Stanford Topic Modeling Toolkit:

We tried the Stanford Topic Modeling Toolbox (TMT) version 0.4.0: http://nlp.stanford.edu/software/tmt/tmt-0.4/. It has functions to train topic models using various models like LDA, Labeled LDA, and Partially Labeled Dirchlet Allocation.

- First we tokenize the dataset using the inbuilt 'SimpleEnglishTokenizer()' function, changing all words to lower-case, and keeping only words and numbers.
- Initially we tried without stemming the dataset. It gave poor results. Here's a snapshot (Figure 2) of the summary.txt file which has a list of relevant words for each topic:

```
45              1914.588022955466
        a       138.97035053931387
        to      110.63121531262362
        how     74.5809275333796
        you     74.0320627953476
        startup 72.18988300447208
        i       56.85778966209576
        do      51.18800019892992
        and     39.09253664505378
        for     34.37657527034822
        my      25.355695155701213
        with    25.049068749644192
        can     24.70240904707087
        your    23.176542497166615
        should  21.721183836414752
        on      21.548682342741635
        it      21.279862974812573
        be      21.131288032046193
        good    20.579807731379155
        company 19.87859872355896
        or      19.447456708736333

32              1268.052775374539
        what    55.424774529591616
        a       48.56407314629293
        to      43.47349955620647
        the     39.73888368972695
        entrepreneurs   35.88132327060705
        are     27.681913084152697
        that    20.53933632689245
        entrepreneur    19.76997136770772
        i       19.562687693448
        have    19.491394826404353
        of      19.404834458845265
        in      19.147653443154667
```

Figure 2: Without stemming

```
45              1361.553115253785
        startup 113.9861161202902
        can     29.706152617773697
        idea    28.40269047850562
        get     23.588074066284534
        good    21.294323469320904
        compani 19.001689425060476
        busi    18.559947678485585
        wai     17.96339270765921
        make    17.604193938136117
        start   17.194808829834532
        product 15.520854316078802
        best    14.201343840456609
        work    13.253450095441698
        team    12.991354132684917
        new     12.145203903599041
        find    11.649295692459948
        build   11.49567705236144
        plan    10.873780065804361
        first   10.730594399134445
        advic   10.466654381970002

32              1041.6942390721222
        entrepreneur    54.62915517623621
        startup 30.56825431236
        success 25.801338555374258
        start   23.145867326145186
        compani 15.522505424431063
        on      15.325038545414184
        get     14.182291839560383
        like    12.08464902432325
        make    11.682442066463794
        want    9.834720280185318
        elon    9.82844219635412
        musk    9.828442191807223
```

Figure 3: With stemming

- After stemming the dataset, we got much better, relevant results as shown in Figure 3.
- The images show the relevant list of keywords for all topics such as topic 45, topic 32 and so on.
- We used Collapsed Variational Bayes Sampler using `TrainCVBOLDA()` function and using the parameter *maxIterations = 10.*

10	9	4.3543374038068E-005	0.000057101	0.0002625827	0.0068825261
11	10	0.0009033642	0.0028569526	0.4035641403	0.0013591894
12	11	0.0012805429	0.0025363043	0.0012371271	0.0010010532
13	12	0.0008250296	0.0002285036	0.0002860416	9.8047070289003E-005
14	13	0.0006587153	0.0005268534	0.0002628781	0.0001245757
15	14	0.0171001852	0.0009340261	0.0004764831	0.0025726915
16	15	0.0047445078	0.0002005894	0.0008186699	0.0027088248
17	16	0.0004692728	0.0005991571	0.0007499593	0.0013746279
18	17	0.0011293665	0.0099341386	0.0045092535	0.0007740184
19	18	0.0004049749	0.0005168943	0.0006466204	0.0002216676
20	19	0.0068158842	0.0037975566	0.7740339922	0.0045809785
21	20	8.3190597005517E-005	0.0002741189	0.0005803445	0.0001458831
22	21	0.0010948016	0.0018421554	0.0013136989	0.0002964891
23	22	0.0020138718	0.0003617211	0.0015750065	0.0002826397
24	23	0.0054528128	0.0112671677	0.173420285	0.0018915865
25	24	4.1755841030026E-005	0.000053354	6.6829467577088E-005	2.9818265948204E-005
26	25	0.0002067267	0.0009408465	0.0009968658	0.0002183013
27	26	3.0701037151732E-006	0.0010461029	5.6666166258031E-006	0.0001925595
28	27	0.005536185	0.0018954429	0.0042424605	0.0080013556
29	28	0.004286664	0.0014857286	0.0002156355	0.0092185097
30	29	9.0964054581611E-005	0.0005900808	0.0005465334	0.0017770793
31	30	0.0003599185	0.0001284764	2.4189606689681E-005	0.0007666858
32	31	0.000342727	0.0084699476	3.7915817367628E-005	0.0073001209

Figure 4: Document-topic distribution using Collapsed Variational Inference.

- We tested the model trained against an evaluation dataset provided by Quora, as described above.
- Figure 4. shows the document-topic distribution obtained: the first column gives the id of the document (here, the question in the evaluation dataset) and rest of the columns give the distribution of all the topics in the corpus for that document.
- Figure 5. below shows the top terms for each topic.

Topic	Top Terms							
Topic 000	fun	tech	startup	on	good	custom	first	even
Topic 001	gave	take	build	time	get	on	start	tech
Topic 002	founder	sold	startup	part	like	start	feel	adeo
Topic 003	startup	space	vs	user	real	find	tech	run
Topic 004	know	basic	person	be	link	do	like	canvas
Topic 005	impact	research	paper	journal	publish	field	work	top
Topic 006	night	vs	feel	do	like	great	give	think
Topic 007	night	toughest	time	character	on	life	think	ever
Topic 008	fashion	popular	with	do	paul	like	adeo	i
Topic 009	i	want	interest	can	recent	give	get	develop
Topic 010	life	character	thing	on	wish	do	with	best
Topic 011	win	war	superior	on	happen	it	take	like
Topic 012	death	mean	be	life	level	art	exist	do
Topic 013	i	can	on	interest	make	develop	want	cse
Topic 014	cse	on	clear	get	like	do	16	vs
Topic 015	life	biggest	right	now	problem	30	like	on
Topic 016	second	think	do	toughest	mean	on	name	clear
Topic 017	word	mean	do	on	sound	with	popular	speech
Topic 018	novel	read	ever	fiction	wish	love	better	written
Topic 019	great	best	on	can	make	do	time	long

Figure 5: Top terms for each topic.

- But surprisingly, the results were very bad. We tried tweaking the parameters like filtering labels that appear in less than 5 documents and increasing the *maxIterations = 100.*
- Yet, there was no relation between the topics predicted and the actual answers.
- We also tried using other samplers like Gibbs Sampler (`TrainGibbsLDA()`).
- Gibbs Sampler has some bug in the library function itself and it fails to give an output.

Stanford TMT has poor documentation. It was challenging to figure out the APIs, resolving time-consuming bugs. We also found that Stanford TMT's algorithms do not perform well on short texts. So, we tried Mallet next.

Mallet:

Mallet is a machine learning tool which has different packages for classification, sequence tagging, topic modeling. We used the topic modeling package which has a fast and scalable implementation of Gibbs Sampling. It is highly optimized for inferring topics from new documents. We used the command line interface provided by mallet to create and train topic model. We used labelled LDA approach as it takes the existing labels into account to train the model and gives a relevant keyword distribution for every topic. Here is a section of topic-keyword output obtained using mallet:

```
165 164     135     214     java framework play jvm spring implement structures net jsf jsp
    appeal struts garbage ide oracle isn't implementation scala systems worth
166 165     130     339     investors angel investor convertible equity invest meeting
    financing note potential difference super typical list questions sheet discount
    angellist emerging structure
167 166     141     221     people country things visiting international time shocking
    encounter trip italy interesting baffling destinations foreigner unexpected stay germany
    photos travel thing
168 167     175     297     e-commerce online ecommerce flipkart platform sales site
    retailers website sites what's websites mortar brick losses retail products payments
    selling shopify
169 168     200     122     mental depression illness ways achieve friend strive self-esteem
    low part difference toxoplasmosis psychiatric ledge lied laziness reasoned nash endure
    premarital
170 169     137     301     amazon microsoft work amazon's india flipkart amazon.com buy
    tech directi recommendation creator parviz babak bangalore kindle market book fresher
    worst
171 170     67      132     estate real house rent buy afford home people apartment invest
```

Figure 6: Topic-keyword output

Mallet identified 186 topic labels. This is accurate since out of 250, 64 labels are not representing any topic. When this model was used for the existing data, it gave a weak document topic distribution, which made us look into other approaches of topic modeling.

Our Approach and Algorithm:

Since Mallet and Stanford TMT failed to give good results, we researched on different topic modeling techniques that can be used. We referred to Sanjeev Arora's paper on topic modeling and built a tool using their idea of anchor words.

Data Preprocessing:

This involves 3 steps:

1. Removing special characters from input file
2. Removing english stop words like a, what, how etc (This step is done to get the keywords for every question)
3. Stemming : This involves converting the word into roots. By this step, keywords can be better matched to see their frequency and help in further analysis. For stemming we

have used PorterStemmer's java implementation. (http://tartarus.org/martin/PorterStemmer/java.txt)

Data Splitting:

After data preprocessing, the training and testing section of data is separated. The code for this section is given in Preprocessor.java.

Construction of matrices:

The tool we built constructs 3 matrices:

1. Word-topic matrix : Since every question in the training data can have multiple topic IDs, we have associated each word in question with multiple topics. The count of the topic is also stored to get an intuition of how often the topic appears for the word.
2. Topic-word matrix: This stores the number of words associated with a topic. Since the distribution of topic for every word is stored in the word-topic matrix, this matrix only maintains the list of words. This is used for normalization.
3. Word occurrence matrix: Every cell stores the occurrence of word in the training document.

Anchor Words:

From the word occurrence matrix, we observed that even though there were unique words which occurred only once in the document, they were by default related to multiple topics. Keeping this in mind, we identified the unique words and gave high weightage to their corresponding topics. Hence any data in training which has these unique words with get these topics. We tested this approach with the testing data and it identified categories for 7/100 questions. At least one category is correctly identified with this approach.

Non-unique words:

For other words, we maintain the count of overlapping topics in every question. These topics are then sorted in the descending order. Thus the top topics are most relevant for the question. This predicted the topics for remaining questions.

Predicting Test-data:

The test questions are then fed into this model. Each test question gets a topic distribution as the output. The output file is a text file which has space separated topics sorted in the degree of relevance. This is then fed into the test script provided by Quora to see how well the model is able to predict.

Techniques Incorporated in the approach:

1. Normalization
2. Ngrams

Initially, we ran the output obtained with the testing script provided by Quora. This is the result we obtained:

Min score: 18.7475433775
Max score: 100
Your raw score: 42.48609075
Your normalized score: 0.292157903396

The score obtained can be further improved using the techniques mentioned above.

Normalization:

Analyzing the topic-word matrix showed that the range of words for every topic is between 0 and 3097. Thus the mean value is 755. Thus every topic has 755 words associated with it. This leads to inaccuracy and the low score shown above. Thus we normalize the topic count with the word frequency and the number of words associated with every topic. Doing this we saw substantial improvement in the model.

Here is the output obtained after normalization of values:

Min score: 18.7475433775
Max score: 100
Your raw score: 59.4654444478
Your normalized score: 0.501128245998

As we can see, the normalized score is doubled after incorporating normalization.

N-grams:

Next we incorporated N-grams into the model. We started with 2 grams approach. Thus we store 2 consecutive words in the question into the above mentioned matrices. After running the model, we observed many relevant 2-gram pairs being formed:

drink water :6
googl glass :10
long term :16
non technic :21
top 10 :29
worst thing :22
silicon vallei :163
tv show :46
rafael nadal :3

This one shows the 2-gram word on the left and it's count. We gave higher weight to N gram words than normal key words. The basic intuition is that such words appearing together multiple times indicates a strong relationship between them. Hence the overlapping topics will have higher relevance. We implemented this and got good results.

The results by N-gram are as follows:

Min score: 18.7475433775
Max score: 100
Your raw score: 55.4902426827
Your normalized score: 0.45220416505

Future Work:

There was a talk hosted by Dr. Haixun Wang of Facebook, at UCLA on 03/08/2016, on 'Short Text Understanding and Semantic Search'. After attending the seminar we realized that there are a bunch of different techniques that can be used for topic modeling on short texts. Currently, we are in touch with Dr. Haixun who has advised us to look into some recent papers in this field that he has sent us. He has also asked us to try Convolutional Neural Networks and see if the results improve. We plan to continue this work over the next few weeks.

Conclusion:

We used different techniques to solve the Quora's labeler challenge. The whole project helped us in gaining a deep insight into topic modeling and its challenges. We learnt different machine learning tools for topic modeling like Mallet, Stanford TMT, NLTK and used them to gain better understanding of data. Using this insight and the niche research done in this area, we tried our own algorithm which would best suit the data. The highest score obtained so far is 135.50 and we were able to achieve an above average score in this challenge.

References:

1. Arora et al. A Practical Algorithm for Topic Modeling with Provable Guarantees: http://arxiv.org/pdf/1212.4777v1.pdf
2. Arora et al. Learning Topic Models — Going beyond SVD: http://arxiv.org/pdf/1204.1956v2.pdf
3. Stanford NLP Topic Modeling Toolbox: http://nlp.stanford.edu/software/tmt/tmt-0.4/
4. Mallet: http://mallet.cs.umass.edu/topics.php
5. http://tartarus.org/martin/PorterStemmer/java.txt
6. http://www.mimno.org/articles/labelsandpatterns/
7. Quora challenge: https://www.quora.com/challenges#labeler