SAN FRANCISCO STATE UNIVERSITY

DIVAM HIREN SHAH

18TH DECEMBER 2018

WINE REVIEWS AND PREDICTIONS

SUBJECT: SEARCH ENGINES

PROFESSOR: DR. ANAGHA KULKARNI

INDEX

ACKNOWLDEGEMENT

I am thankful to Computer Science Department of San Francisco State University, for giving me the opportunity to take up a graduate level class – Search Engines, in my senior year of under-graduation. I acknowledge the efforts put in by my professor – Dr. Anagha Kulkarni, and my peers to help me complete my final term project. Throughout the semester, their constant support and advice has helped me gain a lot of learnings and exposure in the field of information retrieval and text classification.

INTRODUCTION

The goal of the term project is to develop a search engine that would use the wine reviews dataset to extract peculiar features of different wines. The search engine should suggest wines that could be best matched with a person's taste according to their search criteria. It should return results in the form of ranked documents that are related or most relevant to users' query. The ranked documents would be the top descriptions extracted from the wine review data set. Each of these documents would be mapped with the wine-variety and wine vinery. Hence, upon figuring out the most relevant document based on user's taste as a query we will be able to suggest the wine variety along with its wine vinery. The term project uses a dataset from Kaggle which is based on Wine Reviews. The dataset consists of different Wine Reviews from 130K twitter handles. The data is published on a third-party website – winemagzine.com. The dataset is scraped from winemagazine.com and is classified into twelve columns. They are as follows: Id, Country, Description, Designation, Variety, Points, Price, Region, Vinery, Region, Title and Twitter handles of the users. The dataset is presented in the form of a .csv format. Each row of the file represents a description of the wine along with it's country, price, points, wine variety, vinery, twitter handle of the publisher and wine vinery.

The overall project primarily focuses on the field of description and uses it for suggesting the most relevant or similar wine variety based on user query. The project also allows the user to find the right vinery based on their query.

RELATED WORK:

I have referred to the following article and blog to get started with my approach or contribution for the term project.

Zhou, Dong and Vincent, Wade. *Latent Document Re-ranking.* Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. 7th Aug 2009. http://www.aclweb.org/anthology/D09-1163

Zeng, Qing., Redd, Doug., Rindflesch, Thomas and Nebeker, Jonathan. *Synonym, Topic Model, and Predicate-based Query Expansion for Retrieving Clinical documents*. AMIA Annual Symposium Proceedings Archive. 3rd Nov 2012. www.ncbi.nlm.nih.gov

Park, Laurence and Ramamohanarao, Kotagiri. *The Sensitivity of Latent Dirichlet Allocation for Information Retrieval.* ECML PKDD 2009, Part II, LNAI 5782. 2009.

The article "Latent Document Re-ranking" talks about how documents can be affected by Latent Dirichlet Allocation model. The user query can be used to along with document-topic matrix to perform the similarity. The article used KL divergence score method to find the distance and re-weight the documents. This seemed to be a bit complicated for me and I was unable to follow the exact procedure. Due to lack of knowledge I was not able to implement it and instead looked for a substitute. Reading up on other blogs such as "machinelearning.com", "towardsdatascience.com" and many others, I came across the idea of Euclidean Distance and implemented that with some reference and tweaks. The article "Synonym, Topic Model, and Predicate-based Query Expansion for Retrieving Clinical documents" by Zeng, Qing. Redd, Dough. Rindflesch, Thomas and Nebeker Jonathan talks about how topic-words may be selected for query expansion. The authors state that we should, apply topic modelling algorithm on the set documents. Once, topics are generated we can find the dominant topic for each document. We can also make the table for topic-document distribution to have a look at the topic distribution for each document. In the article, the authors use term weights for each topic in order to perform query expansion. The terms from matching topics were ranked by summing their weights in all matching Topics. This exposed me to idea of selecting top n words from a selected topic(dominant) from a selected document.

OUR APPROACH

The first step to perform and execute our term project was to create concrete rules that would allow us to guide towards our project goal. We had to create a base line for our project where we could know what a simple search engine like Indri would generate results. In order to run Indri Index builder, it was important for us to know the sample size we would like to use for this project for testing purposes. The total number of documents is approximately 130,000. We use first 5000 documents from the total number of documents for evaluating and testing purposes. We use each description as a single text document. In order to perform that I generated a python script which would parse the .json file of the dataset to generate each row of the file or key of the file as a separate text file in a selected directory. I then named the files with a counter which would be helpful for us, as the name of the file would suggest the Id of the document. For example: 30.txt would tell us that the most relevant document based on the user query was document with ID as 30. This supported us well with having a base line and did not force us to manually check the dataset. After establishing our base line each of us had to come up with approaches that would allow us to build a search engine better than Indri or in fact build a search engine that would be efficient and relevant to what the user wishes to get. Upon building the indri index builder and using its retrieval application interface to retrieve the set of ranked documents based on user query, I analyzed that most of the queries on Indri search engine produced similar results to a tf-idf algorithm that was used earlier in class.

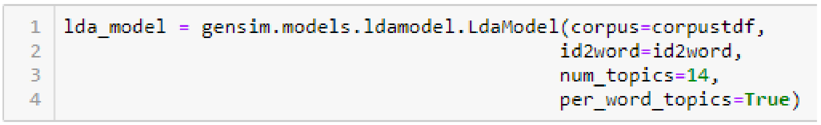
This gave us a hint to kick start our project with optimizing the tf-idf algorithm used in class to make an efficient search based on the user query. Due to the vast dataset, we could take query terms from the user and map it to various fields in the dataset apart from description. We could check if the query terms relate country or region or price fields in the data. This could also lead us to optimize the user query through expansion and other related words (synonyms). We first performed a simple tf-idf algorithm on the document to get the ranked results based on users' taste mentioned in the query. The next step is to analyze user query term by term and check if any of the terms mentioned in the user query contains terms that are region or country oriented. We use the CLDR (country, code, language mapping) data dictionary and a Wikipedia list of demonyms (France: French) and compare with the terms in the fields of country and region in the wine review data set. We also use wordnet to expand on region names. We send the query through countries ranking function and upon successful match, we optimize the tf-idf algorithm and change the parameters of the weights to highlight the importance of the users' query and bubble up the document rank. We also try to handle query terms that would not be present in the document but can be related more by the user. This would be done by using a dictionary of synonyms called "Wordnet". For example, if the query term mentioned the word "sweet" we could look for documents that have words related with "sweet". For example: "sweetness", "fruity", "sugary" and other such synonyms. We simulate query terms to look up for meaning full synonyms with the Wordnet dictionary and then compare with description field in the dataset. This would capture the documents that may directly not be handled by tf-idf algorithm but related to user query. As the queries would be free text queries, the flexibility of user will not be diminished and at the same time synonyms, demonyms will be taken into consideration. This overall would change the rankings of document as each index would be running in parallel and lead to re-weighting of terms and hence affecting the tf-idf of documents. Last but not the least, we check for price related language used in the query. For example, if the user query term contains words like "cheap", "expensive", "low" then we change weight given to the columns on their price field. We have tried to implement a text-blob like approach but not text-blob. This is to check the polarity and negate it words like "not so expensive"," inexpensive". The above-mentioned approach is just for experimenting purposes and does not always gives the right results.

MY CONTRIBUTION

Upon reading articles and research papers, I learnt that I could bring in the concept of topic modelling and extract hidden patterns from vast set of documents. This would let us know what each type of document is talking about and what are the most relevant terms to that document. After a final computation of tf-idf algorithm on individual parameters as mentioned above, we bring in the concept of Latent Dirichlet Allocation model. This is an unsupervised approach used to extract hidden pattern in text documents. This model generates topics for each document. The LDA model is trained on 5000 documents and tested with results of tf-idf algorithm. This would help us the know what the topics user are interested in. Once, we find the dominant topic in each document, we can use the top keywords in the topic for query expansion to optimize the search results. One associated risk with this approach is, we may expand the query if it is already too long. However, that can be prevented upon setting some rules. For example, if the query is more than 10 terms then we shall not expand the query. Rather use the approach of finding the distance between the query-topic-document matrix and topic-document matrix and return the most similar documents related with user query. This can help us achieve efficiency and prevent some ambiguity in the user query. If the user query is small, we can use query expansion and internally call the tf-idf algorithm to generate the final list of documents. Out of many topic modeling algorithms, I found Latent Dirichlet Allocation (LDA) to be the most efficient one as it gave clear topics to my dataset. However, I did not test other topic modeling algorithms such as Non-negative Matrix Factorization (NMF) as it relies on linear algebra. LDA, on the other hand is based on probabilistic graphical modeling. The aim of the algorithm is to produce 2 matrices. That are: Document-term matrix and Topic-Word matrix. Latent Dirichlet Allocation learns from the data and uses it’s learning to predict topics for any unseen document. In our case the unseen document is the user query. In order to perform LDA on a set of documents, the first step is to pre-process and clean the documents. Pre-processing majorly consists of normalization of text, that is converting document to lower case, tokenization of text into words, removing stop words such as ‘at’, ‘it’, ’the’, ‘and’ and many other words. I have also taken into account the length of words. Any word with length less than or equal three has been ignored. These words would majorly be stop-words. One of the most important analysis made by me in pre-processing step was the use of lemmatization and stemming. When we perform lemmatization through spacy model in python, we have an easy way of focusing on POS tags, i.e. Part of Speech tags. These would be nouns, adverbs, adjectives and verbs. Moreover, I noticed the model created better topics and words. This can be supported by low perplexity and high loglikelihood score of the model. These two components of the model expose us to understand the what influences the model. On the other hand, using the porter stemmer stemming of text would reduce the score of the loglikelihood and increase perplexity of the model. The only reason to use the porter stemmer was because of the sample size of the data. We only trained the model on 5000 documents. In addition, in-order to execute the project in a coherent way and function it with other segments of the project I chose the porter stemmer and not lemmatization. After deciding upon the pre-processing part, I also remove any special characters and symbols which may be present in the regular document through use of regular expression. This is to avoid any corruption in the model and run the search efficiently. One of the difficult tasks to decide upon the Latent Dirichlet model was to figure out the right use of library. There were two possibilities: use of the gensim library or use of scikit-learn library.

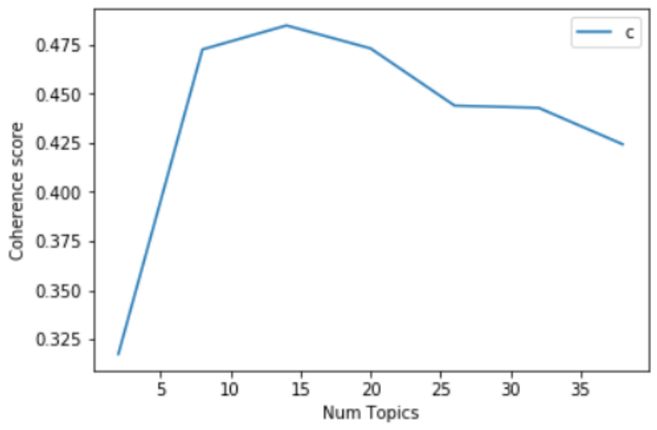
<u>GENSIM LIBRARY:</u>

I first tried my approach with genism library. The LDA model in the genism library uses corpus in the form of bag of words format. That is nothing but describing how many times each word in the document appears. It calculates the term-document frequency. The method in gensim library uses this corpus as it's learning data to suggest top topics from the set of 5000 documents.

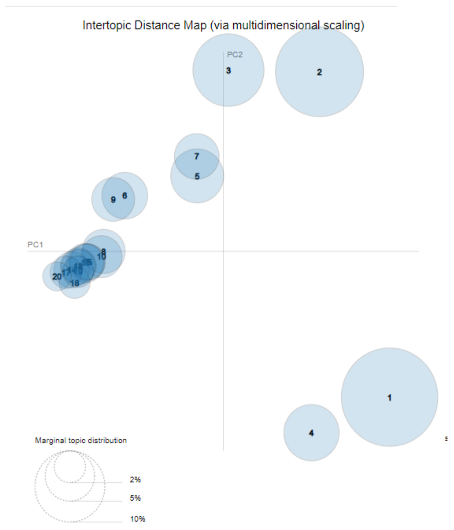


```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpustdf,
                                           id2word=id2word,
                                           num_topics=14,
                                           per_word_topics=True)
```

One thing to note about the LDA models in the both libraries is that, we need to decide upon the optimal number of topics to be generated for a given dataset. The model while using gensim library gives us low perplexity score and an average coherent score. The above two factors are deciding factors for selecting the optimal number of topics. Below is a small graph generated by use of matplotlib.pylot library in python.



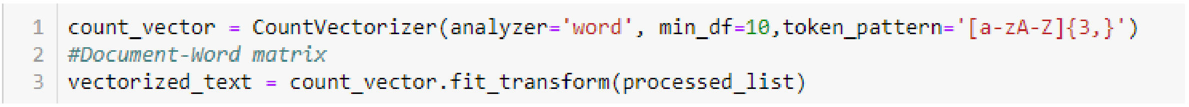
The x-axis represents the number of topics and y-axis represents corresponding coherence score for every x. We can infer from the graph that, we get the highest coherence score in the range of 0.45 to 0.48 when we select any topics between the range of 10 to 20 topics. Hence, I chose 14 topics as my optimal number of topics for the dataset as it gave me the highest coherence score and also gave me a very low perplexity score. The scores generated were much higher and better than what the model through scikit-learn library generated. However, it is interesting to note that topics generated in gensim library were too congested and the cluster overlapped for majority of the topics.



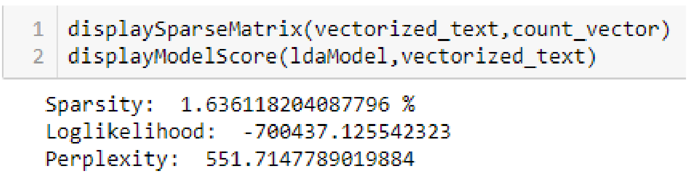
From the above graph we can clearly conclude how topics are clustered and congested.

<u>SCIKIT-LEARN APPROACH:</u>

In the scikit-learn, the LDA model makes use of count vectorizer library. This is primarily used to clear out the noise from the pre-processed data and build a matrix of token counts.



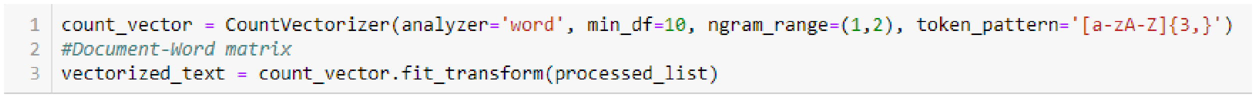
```
count_vector = CountVectorizer(analyzer='word', min_df=10,token_pattern='[a-zA-Z]{3,}')
#Document-Word matrix
vectorized_text = count_vector.fit_transform(processed_list)
```



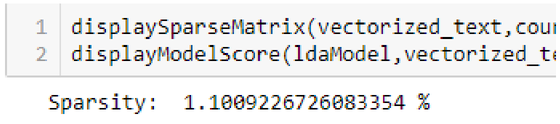
```
displaySparseMatrix(vectorized_text,count_vector)
displayModelScore(ldaModel,vectorized_text)
```

```
Sparsity:  1.636118204087796 %
Loglikelihood:  -700437.125542323
Perplexity:  551.7147789019884
```

In the above snippet, vectorized_text is the document-word matrix and has been generated with help of other parameters like stop words, analyzer and token_pattern. We then use the LDA library to generate the LDA model. One of the other parameters, that I thought shall be useful is ngram_range which could help us to create bigrams preserve the meaning of other words. However, inclusion of the ngram_range parameter harmed my scores of the model and decreased the sparsity of my countVecotrizer model. This indeed affected my topic cluster.



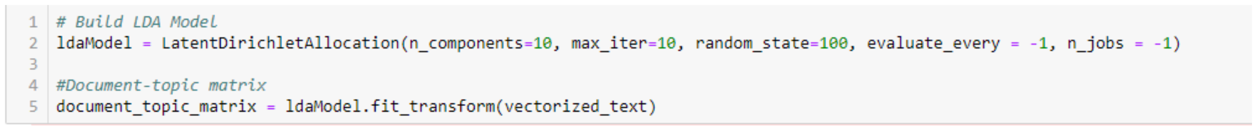
```
count_vector = CountVectorizer(analyzer='word', min_df=10, ngram_range=(1,2), token_pattern='[a-zA-Z]{3,}')
#Document-Word matrix
vectorized_text = count_vector.fit_transform(processed_list)
```



```
displaySparseMatrix(vectorized_text,cou
displayModelScore(ldaModel,vectorized_t
```

```
Sparsity:  1.1009226726083354 %
```

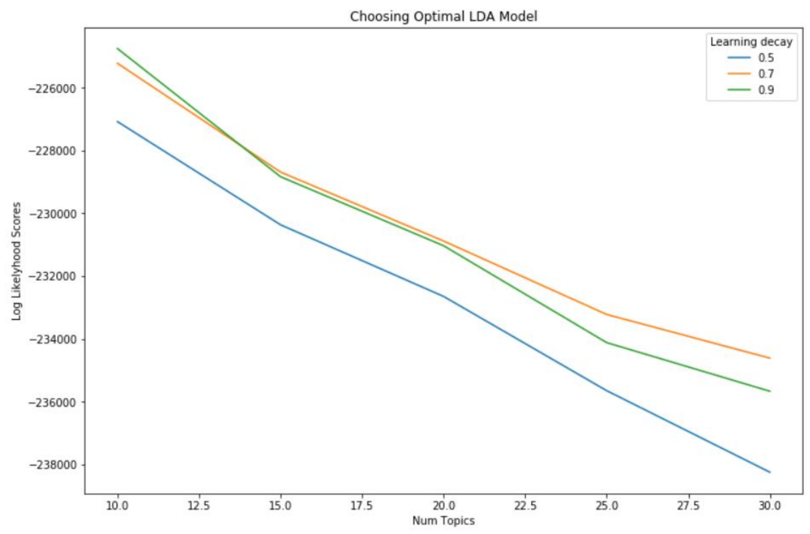
The document_topic_matrix represents the score of each topic in each document. Parameters such as learning_method are used because of the large datasets. Other parameters like n_jobs, batch size is used to run LDA model efficiently. Batch size represents the number of documents to be used in each iteration. n_jobs mean number of processes to be used. -1 indicates usage of all processes. The max_iter parameters indicates number of iterations.



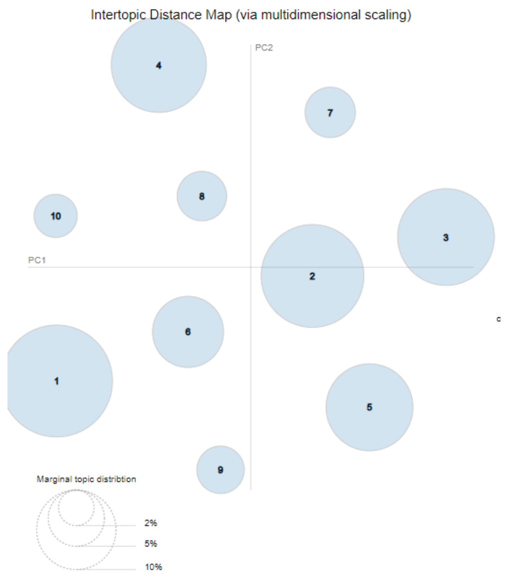
```
# Build LDA Model
ldaModel = LatentDirichletAllocation(n_components=10, max_iter=10, random_state=100, evaluate_every = -1, n_jobs = -1)

#Document-topic matrix
document_topic_matrix = ldaModel.fit_transform(vectorized_text)
```

I have selected the number of topics to be 10 in scikit-learn approach because it gives me the best loglikelihood score and low perplexity score. One of the factors that harms these scores is size of the data and what we pass in countVecotrizer library.



From the above graph we can incur that when we select 10 topics, we get better log likelihood score. It is surprising to know that despite a poor perplexity score than gensim library, we still get good clusters of topics and they are widely spread. Below is a graph that depicts the visualization of topics.



We can infer that the topics are clear and widely distributed when we use the LDA model through scikit learn model.

Topic Prediction and Query Expansion:

After getting the right number of topics and words, we can now generate topics for an unseen document or a user query. The query falls into a dominant topic and we can roughly get an idea about what the user is looking for and what might be relevant documents for that particular topic and user query. Of course, the model will not always generate the right topics however we can influence the user query by expanding through usage of dominant words under each topic. This would guide the user query towards relevant document. This can help the user explore documents which may go un-noticed in the tf-idf approach. This could be used more as a suggestion feature in the search engine. We first select top ‘n’ documents from the list of results or initial retrieval performed by tf-idf algorithm. We then select the dominant topic for each selected document. Out of the selected topics, we choose the most frequent word in the topic arranged by its component weights. This tells us how many times the word has appeared in the documents that are relevant to a particular topic. We select two words from each topic and in total select top 2 documents. This would then mean that we expand the user query by 4 words. This approach can be used when the user query has only 2-3 terms. Expanding the user query internally by 4 words would give him more clarity on what documents or wine variety he is looking for. This could be used as an application of suggestion to the user. We can then recall the tf-idf algorithm which would re-weight the documents because of internal inclusion of keywords from the generated topics of top-n documents in initial retrieval. However, there is always a risk when we expand the user query as the algorithm i.e. selection of top words from selected topics may not be precise enough to maintain the originality of the user query.

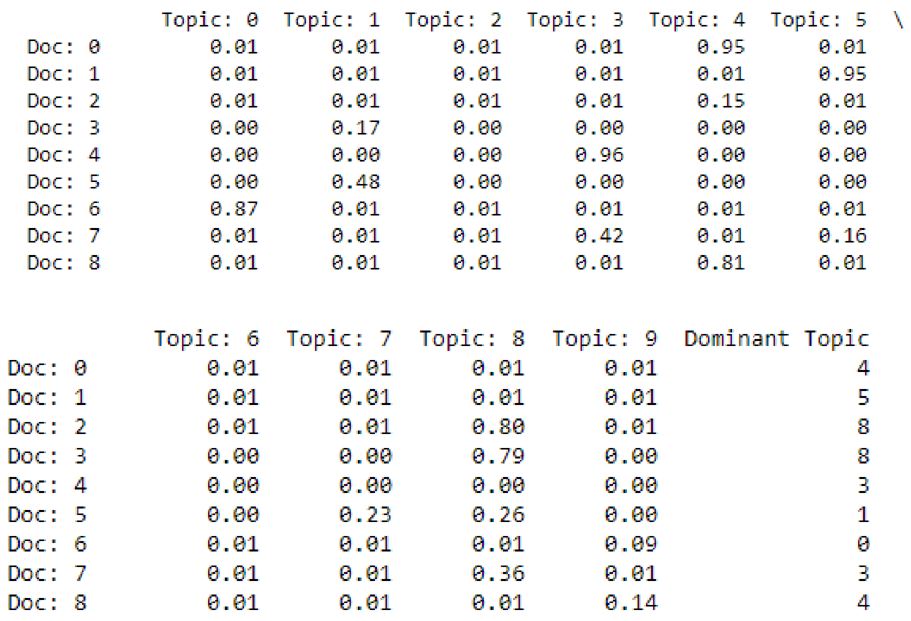
Euclidean Distance:

The other plausible approach is not to use query expansion, as it might even mislead the user to other wine-variety. In this case, we can calculate the Euclidean distance between the resulted-document topic distribution based on the query and topic-document matrix distribution of the loaded original document. The shortest distance calculated will mean that, the selected document in the topic-document matrix is most similar document for the topic generated by the first document in initial retrieval process by original user query.

My contribution is purely based on topic-modelling and query expansion. It relies heavily on the tf-idf algorithm once query expansion is performed.

<u>Overview of topic distribution and Scores:</u>

<u>Topic Distribution:</u>

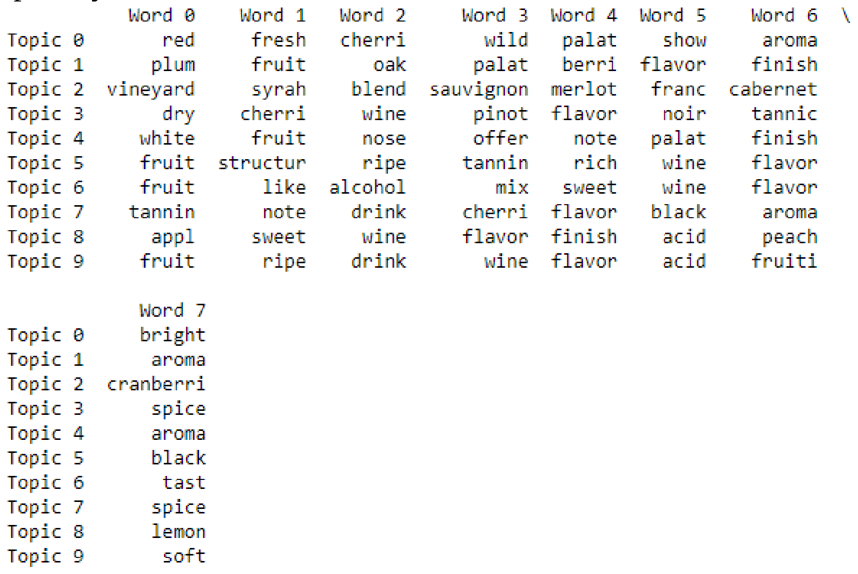


```
         Topic: 0  Topic: 1  Topic: 2  Topic: 3  Topic: 4  Topic: 5  \
Doc: 0       0.01      0.01      0.01      0.01      0.95      0.01
Doc: 1       0.01      0.01      0.01      0.01      0.01      0.95
Doc: 2       0.01      0.01      0.01      0.01      0.15      0.01
Doc: 3       0.00      0.17      0.00      0.00      0.00      0.00
Doc: 4       0.00      0.00      0.00      0.96      0.00      0.00
Doc: 5       0.00      0.48      0.00      0.00      0.00      0.00
Doc: 6       0.87      0.01      0.01      0.01      0.01      0.01
Doc: 7       0.01      0.01      0.01      0.42      0.01      0.16
Doc: 8       0.01      0.01      0.01      0.01      0.81      0.01

         Topic: 6  Topic: 7  Topic: 8  Topic: 9  Dominant Topic
Doc: 0       0.01      0.01      0.01      0.01               4
Doc: 1       0.01      0.01      0.01      0.01               5
Doc: 2       0.01      0.01      0.80      0.01               8
Doc: 3       0.00      0.00      0.79      0.00               8
Doc: 4       0.00      0.00      0.00      0.00               3
Doc: 5       0.00      0.23      0.26      0.00               1
Doc: 6       0.01      0.01      0.01      0.09               0
Doc: 7       0.01      0.01      0.36      0.01               3
Doc: 8       0.01      0.01      0.01      0.14               4
```

As we can see most dominant topic in document 0 is topic 4 with 95% probability.

<u>Topic keywords:</u>



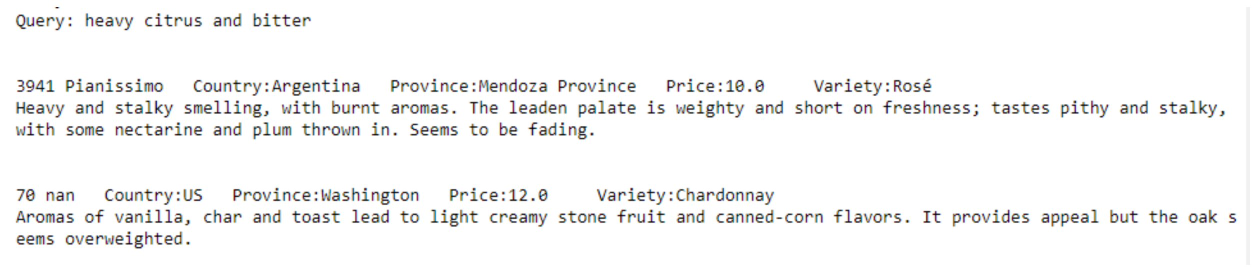
```
            Word 0     Word 1   Word 2     Word 3  Word 4  Word 5    Word 6  \
Topic 0        red      fresh   cherri       wild   palat    show     aroma
Topic 1       plum      fruit      oak      palat   berri  flavor    finish
Topic 2   vineyard      syrah    blend  sauvignon  merlot   franc  cabernet
Topic 3        dry     cherri     wine      pinot  flavor    noir    tannic
Topic 4      white      fruit     nose      offer    note   palat    finish
Topic 5      fruit   structur     ripe     tannin    rich    wine    flavor
Topic 6      fruit       like  alcohol        mix   sweet    wine    flavor
Topic 7     tannin       note    drink     cherri  flavor   black     aroma
Topic 8       appl      sweet     wine     flavor  finish    acid     peach
Topic 9      fruit       ripe    drink       wine  flavor    acid    fruiti

              Word 7
Topic 0       bright
Topic 1        aroma
Topic 2   cranberri
Topic 3        spice
Topic 4        aroma
Topic 5        black
Topic 6         tast
Topic 7        spice
Topic 8        lemon
Topic 9         soft
```

RESULTS AND ANALYSIS

We all know, there is no better computing brain than human. Algorithms are always in a risk with demanding changes and evolution. When it comes to topic-modelling and basics of machine learning we do not know what is right or what is irrelevant to the user. I have performed few analyses on query and made some analyses on how the model responds in different cases. The model works well with some queries where as it has no effect on some queries. There may be queries which may fetch irrelevant documents to the user or there might be incomplete or ambiguous queries which may in fact capture documents that the user maybe looking by the above-mentioned approaches.

Query: heavy citrus and bitter



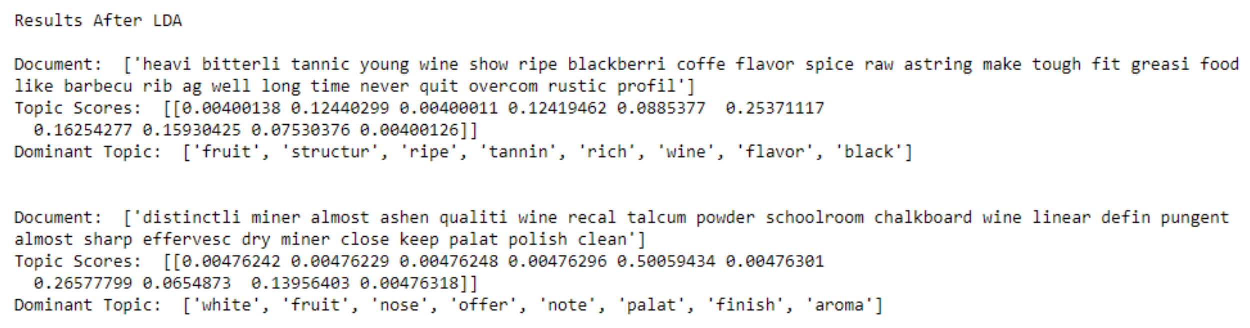
```
Query: heavy citrus and bitter


3941 Pianissimo   Country:Argentina   Province:Mendoza Province   Price:10.0     Variety:Rosé
Heavy and stalky smelling, with burnt aromas. The leaden palate is weighty and short on freshness; tastes pithy and stalky,
with some nectarine and plum thrown in. Seems to be fading.


70 nan   Country:US   Province:Washington   Price:12.0     Variety:Chardonnay
Aromas of vanilla, char and toast lead to light creamy stone fruit and canned-corn flavors. It provides appeal but the oak s
eems overweighted.
```

The above image shows a simple tf-idf ranking of all documents based on the user query.

However, after applying LDA we get the dominant topics and topic-document distribution scores for each selected topic

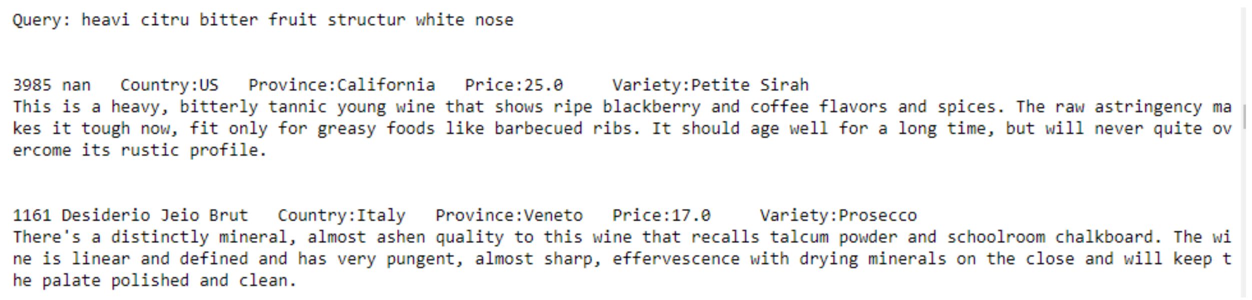


```
Results After LDA

Document:  ['heavi bitterli tannic young wine show ripe blackberri coffe flavor spice raw astring make tough fit greasi food
like barbecu rib ag well long time never quit overcom rustic profil']
Topic Scores:  [[0.00400138 0.12440299 0.00400011 0.12419462 0.0885377  0.25371117
  0.16254277 0.15930425 0.07530376 0.00400126]]
Dominant Topic:  ['fruit', 'structur', 'ripe', 'tannin', 'rich', 'wine', 'flavor', 'black']


Document:  ['distinctli miner almost ashen qualiti wine recal talcum powder schoolroom chalkboard wine linear defin pungent
almost sharp effervesc dry miner close keep palat polish clean']
Topic Scores:  [[0.00476242 0.00476229 0.00476248 0.00476296 0.50059434 0.00476301
  0.26577799 0.0654873  0.13956403 0.00476318]]
Dominant Topic:  ['white', 'fruit', 'nose', 'offer', 'note', 'palat', 'finish', 'aroma']
```

Query gets expanded from “heavy bitter” to “heavy citrus and bitter fruit structure white nose” suggesting user to check wine varieties with fruit or structure or white
Note: The query gets stemmed once it is expanded to maintain uniformity.



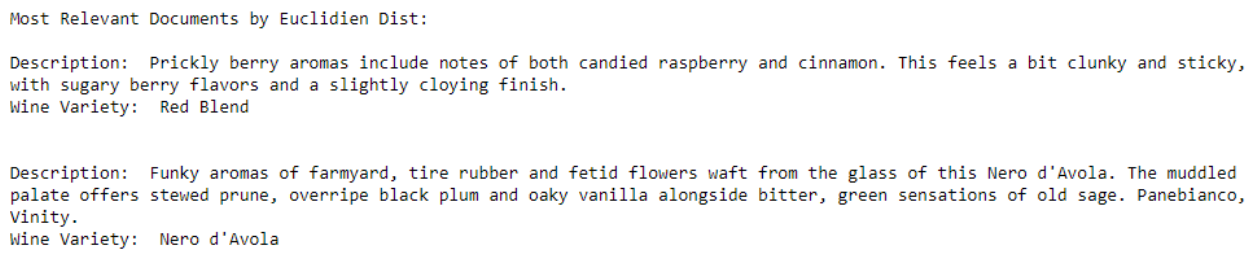
```
Query: heavi citru bitter fruit structur white nose


3985 nan    Country:US    Province:California    Price:25.0     Variety:Petite Sirah
This is a heavy, bitterly tannic young wine that shows ripe blackberry and coffee flavors and spices. The raw astringency ma
kes it tough now, fit only for greasy foods like barbecued ribs. It should age well for a long time, but will never quite ov
ercome its rustic profile.


1161 Desiderio Jeio Brut    Country:Italy    Province:Veneto    Price:17.0     Variety:Prosecco
There's a distinctly mineral, almost ashen quality to this wine that recalls talcum powder and schoolroom chalkboard. The wi
ne is linear and defined and has very pungent, almost sharp, effervescence with drying minerals on the close and will keep t
he palate polished and clean.
```

We can see how query expansion leads to a re-weighing of documents and affects the ranks of the initial retrieval. Moreover, it still does not harm the original meaning of the user query.

Applying the Euclidean Distance approach, we can look at how the query is targeted towards similar document with related topic keywords. In this approach, we calculate the distance between document-topic distribution and the document-query topic distribution. The smallest distance is the closest or most similar document.



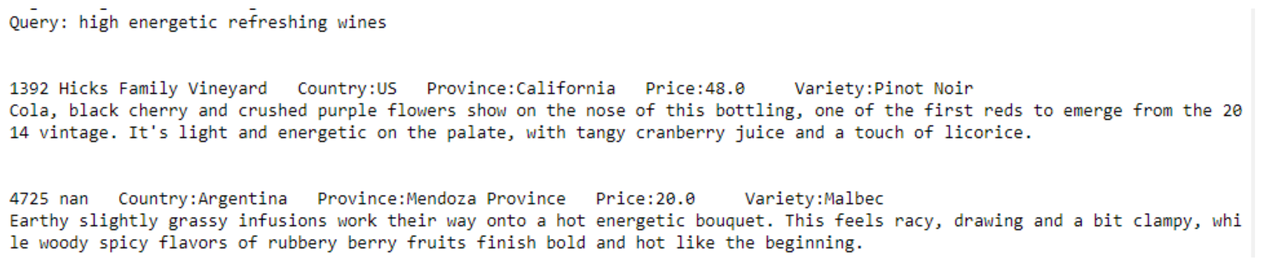
```
Most Relevant Documents by Euclidien Dist:

Description:  Prickly berry aromas include notes of both candied raspberry and cinnamon. This feels a bit clunky and sticky,
with sugary berry flavors and a slightly cloying finish.
Wine Variety:  Red Blend


Description:  Funky aromas of farmyard, tire rubber and fetid flowers waft from the glass of this Nero d'Avola. The muddled
palate offers stewed prune, overripe black plum and oaky vanilla alongside bitter, green sensations of old sage. Panebianco,
Vinity.
Wine Variety:  Nero d'Avola
```

Query 2: high energetic refreshing wines

Performing tf-idf

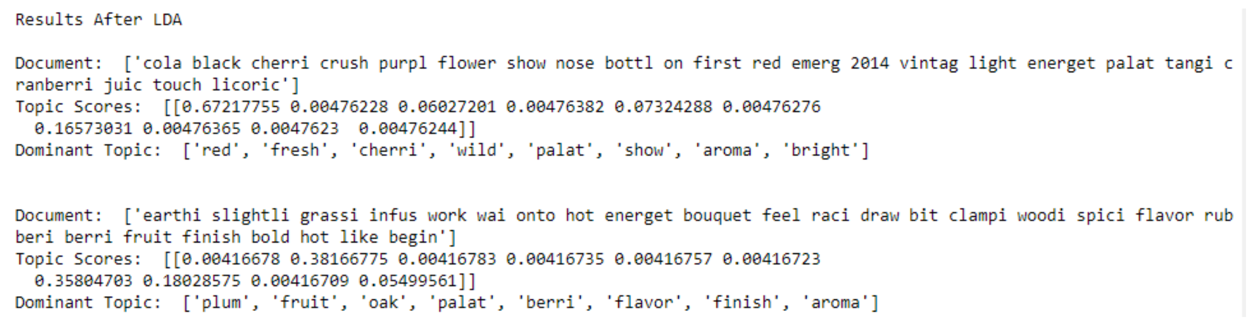


```
Query: high energetic refreshing wines


1392 Hicks Family Vineyard    Country:US    Province:California    Price:48.0     Variety:Pinot Noir
Cola, black cherry and crushed purple flowers show on the nose of this bottling, one of the first reds to emerge from the 20
14 vintage. It's light and energetic on the palate, with tangy cranberry juice and a touch of licorice.


4725 nan    Country:Argentina    Province:Mendoza Province    Price:20.0     Variety:Malbec
Earthy slightly grassy infusions work their way onto a hot energetic bouquet. This feels racy, drawing and a bit clampy, whi
le woody spicy flavors of rubbery berry fruits finish bold and hot like the beginning.
```

Results after performing LDA:

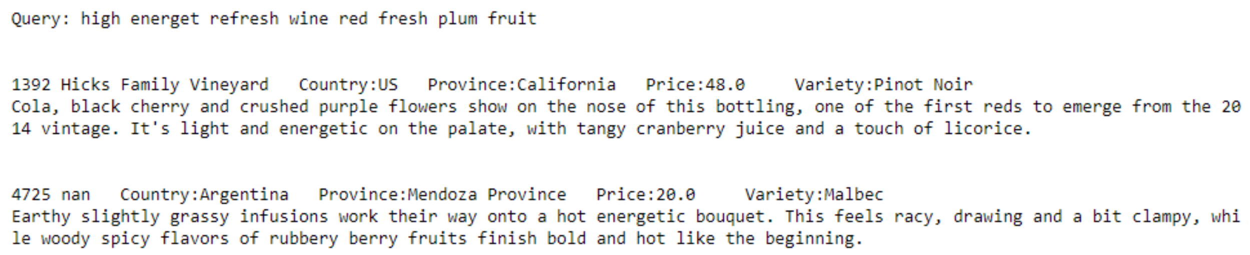


```
Results After LDA

Document:  ['cola black cherri crush purpl flower show nose bottl on first red emerg 2014 vintag light energet palat tangi c
ranberri juic touch licoric']
Topic Scores:  [[0.67217755 0.00476228 0.06027201 0.00476382 0.07324288 0.00476276
  0.16573031 0.00476365 0.0047623  0.00476244]]
Dominant Topic:  ['red', 'fresh', 'cherri', 'wild', 'palat', 'show', 'aroma', 'bright']


Document:  ['earthi slightli grassi infus work wai onto hot energet bouquet feel raci draw bit clampi woodi spici flavor rub
beri berri fruit finish bold hot like begin']
Topic Scores:  [[0.00416678 0.38166775 0.00416783 0.00416735 0.00416757 0.00416723
  0.35804703 0.18028575 0.00416709 0.05499561]]
Dominant Topic:  ['plum', 'fruit', 'oak', 'palat', 'berri', 'flavor', 'finish', 'aroma']
```

Results after query expansion:



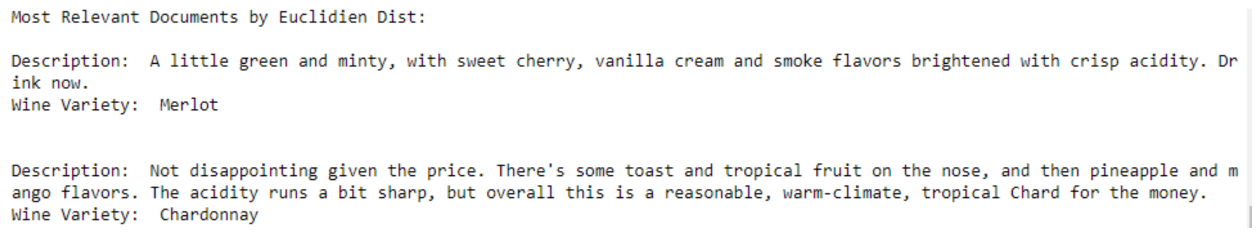
```
Query: high energet refresh wine red fresh plum fruit


1392 Hicks Family Vineyard    Country:US    Province:California    Price:48.0     Variety:Pinot Noir
Cola, black cherry and crushed purple flowers show on the nose of this bottling, one of the first reds to emerge from the 20
14 vintage. It's light and energetic on the palate, with tangy cranberry juice and a touch of licorice.


4725 nan    Country:Argentina    Province:Mendoza Province    Price:20.0     Variety:Malbec
Earthy slightly grassy infusions work their way onto a hot energetic bouquet. This feels racy, drawing and a bit clampy, whi
le woody spicy flavors of rubbery berry fruits finish bold and hot like the beginning.
```

As we can see from above results that query expansion in this case fails as it does not influence the ranking of tf-idf document. It gives the similar documents as tf-idf does.

However, Euclidean still gives us a unique set of documents:



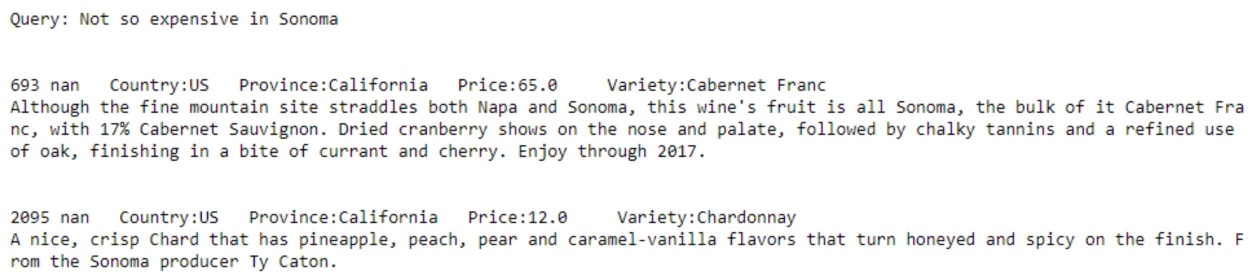
```
Most Relevant Documents by Euclidien Dist:

Description:  A little green and minty, with sweet cherry, vanilla cream and smoke flavors brightened with crisp acidity. Dr
ink now.
Wine Variety:  Merlot


Description:  Not disappointing given the price. There's some toast and tropical fruit on the nose, and then pineapple and m
ango flavors. The acidity runs a bit sharp, but overall this is a reasonable, warm-climate, tropical Chard for the money.
Wine Variety:  Chardonnay
```

Query 3: Not so expensive in Sonoma

A simple tf-idf: Highlighting the importance of price and country/region:



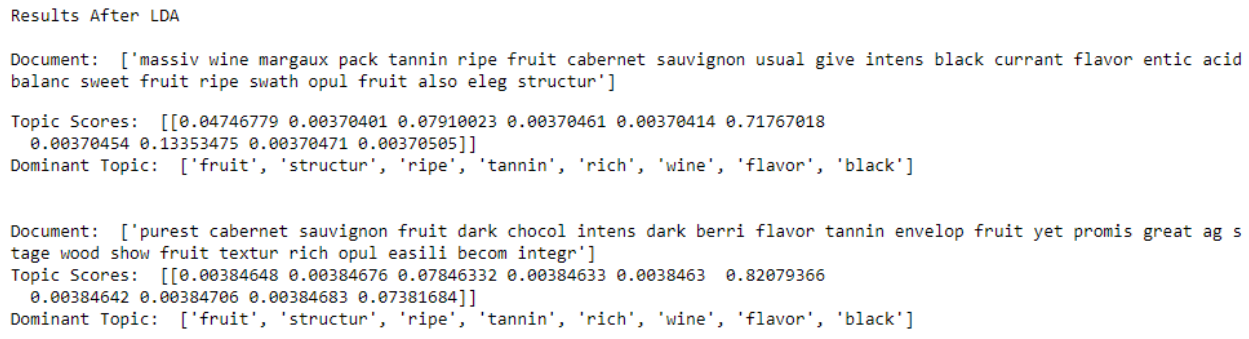
```
Query: Not so expensive in Sonoma


693 nan    Country:US    Province:California    Price:65.0     Variety:Cabernet Franc
Although the fine mountain site straddles both Napa and Sonoma, this wine's fruit is all Sonoma, the bulk of it Cabernet Fra
nc, with 17% Cabernet Sauvignon. Dried cranberry shows on the nose and palate, followed by chalky tannins and a refined use
of oak, finishing in a bite of currant and cherry. Enjoy through 2017.


2095 nan    Country:US    Province:California    Price:12.0     Variety:Chardonnay
A nice, crisp Chard that has pineapple, peach, pear and caramel-vanilla flavors that turn honeyed and spicy on the finish. F
rom the Sonoma producer Ty Caton.
```

We can see how the user query gets mapped with country and in fact the algorithm tries to bring up wines which are not so expensive.

After performing LDA:



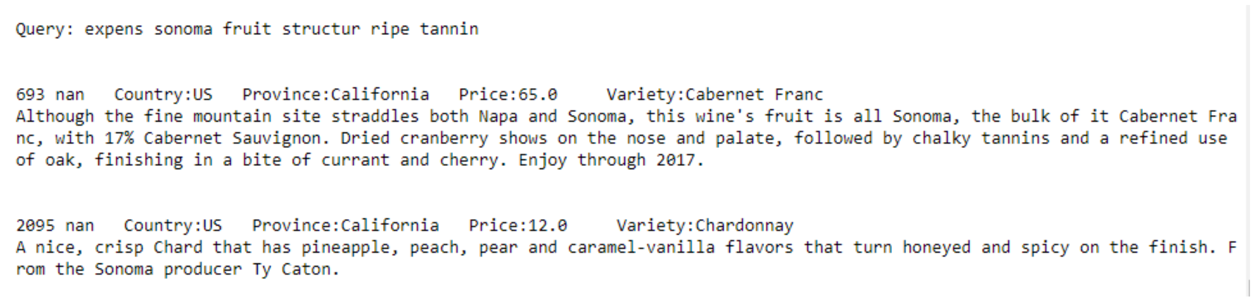
```
Results After LDA

Document:  ['massiv wine margaux pack tannin ripe fruit cabernet sauvignon usual give intens black currant flavor entic acid
balanc sweet fruit ripe swath opul fruit also eleg structur']

Topic Scores:  [[0.04746779 0.00370401 0.07910023 0.00370461 0.00370414 0.71767018
  0.00370454 0.13353475 0.00370471 0.00370505]]
Dominant Topic:  ['fruit', 'structur', 'ripe', 'tannin', 'rich', 'wine', 'flavor', 'black']


Document:  ['purest cabernet sauvignon fruit dark chocol intens dark berri flavor tannin envelop fruit yet promis great ag s
tage wood show fruit textur rich opul easili becom integr']
Topic Scores:  [[0.00384648 0.00384676 0.07846332 0.00384633 0.0038463  0.82079366
  0.00384642 0.00384706 0.00384683 0.07381684]]
Dominant Topic:  ['fruit', 'structur', 'ripe', 'tannin', 'rich', 'wine', 'flavor', 'black']
```

Query Expansion:



```
Query: expens sonoma fruit structur ripe tannin


693 nan    Country:US    Province:California    Price:65.0     Variety:Cabernet Franc
Although the fine mountain site straddles both Napa and Sonoma, this wine's fruit is all Sonoma, the bulk of it Cabernet Fra
nc, with 17% Cabernet Sauvignon. Dried cranberry shows on the nose and palate, followed by chalky tannins and a refined use
of oak, finishing in a bite of currant and cherry. Enjoy through 2017.


2095 nan    Country:US    Province:California    Price:12.0     Variety:Chardonnay
A nice, crisp Chard that has pineapple, peach, pear and caramel-vanilla flavors that turn honeyed and spicy on the finish. F
rom the Sonoma producer Ty Caton.
```

We can see that we lose out on the original meaning of the query, as stop words like “not”, “so” get removed and the user query is different from the original one. In addition, the LDA model does not influence does not make a huge difference in the top 5 documents that are retrieved. All in all, the topics generated by the LDA model are helpful as they determine the characteristics of wine and also, let us know what each document talks about. However, we still cannot make an efficient use of the model. This where concepts like decision tree (Random Forest Tree) and other turn out to be helpful for a better match of documents.

CONCLUSION:

I have implemented Latent Dirichlet Allocation Model at the backend of my project. The user query is entered and TF-IDF with various indices like country, region, price and description is used to calculate the ranks of the documents. I train my LDA model on 5000 documents and test the user query to predict its topic. Once we perform the initial retrieval, I select the top 2 documents and its dominant topic. Under each dominant topic I use top key-words and implement the concept of query expansion. As we can look at the results and understand that query expansion not always generates the best results, I also bring in the concept of euclidean distance which may be used to push the user query towards the most similar document. This could have been better performed with KL-divergence approach as discussed in one of the articles in related work section. Due to lack of knowledge I was not able to implement it and instead looked for a substitute – Euclidean Distance. One of the drawbacks we encountered was, our dataset was relatively fresh and lacked evaluation process. We now have learnt our lesson and understood what are the factors that are needed to be considered in order to choose a dataset. I also learnt various algorithms for topic-modelling and different classifiers which were thought in class. This class has made me learn jupyter notebook and exposed me to python programming language.

FUTURE WORK:

In the future, we can work on optimal algorithms that derive meaningful topic-keywords from the topic which may help the user to extract better sets of results. Currently, I am picking up most frequent terms from 2 topics and then appending it with user query. We can also make use of decision trees like Random Forest Tree and analyze its output. We can also try to implement KL divergence score after initial retrieval of documents and applying Latent Dirichlet Allocation model. There are lot of options that can be performed on a well classified dataset. Usage of Relevance Feedback and other such techniques including accessing of query logs could lead us to more efficient and qualitative search engine.

LIBRARIES AND REFERENCES:

1) Latent Dirichlet Allocation model is imported from scikit-learn library

   from sklearn.decomposition import LatentDirichletAllocation

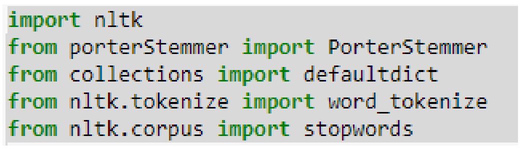
   ➢ https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

2) Count Vectorizer model:

   from sklearn.feature_extraction.text import CountVectorizer

   ➢ https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

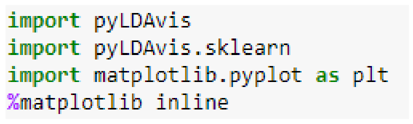
3) Natural Language ToolKit used for pre-processing and cleaning the documents



```
import nltk
from porterStemmer import PorterStemmer
from collections import defaultdict
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

   ➢ https://www.nltk.org/

4) pyLDAvis is used for data visualization of topics.



```
import pyLDAvis
import pyLDAvis.sklearn
import matplotlib.pyplot as plt
%matplotlib inline
```

   ➢ https://pyldavis.readthedocs.io/en/latest/

I have researched, and applied techniques mentioned in the following articles and research paper which has helped me achieve my contributions and related work.