Lab Report 1
Ziphs Law

Irum Hussain
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162005

Karre Kamal
Artificial Intelligence
IIIT Vadodara
Gandhinagar, Gujarat
202161004

Prashant Kumar
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162007

***Abstract*—Zipf's law defines how often words appear in a a large corpus.**

I. Introduction

Most of the electronic text content may be encoded into format supported by various text processing modules. A raw text should be generated from documents in binary formats must - a set of characters as a primary step prior to tokenization. Frequency distribution of words is very skewed. There are a few words that have a very hight frequencies and many words that have low frequencies. This frequency distribution of words described by Zipf's law.

II. Approach

In natural language, few terms are very frequent and many are very rare terms. According to Ziphs law the nth most frequent term has frequency proportional to 1/n.

$$f(r) \propto 1/r \tag{1}$$

$$f(r) \propto 1/r = K/r \tag{2}$$

where K is a normalizing constant. In this equation, r is rank of a word, and f(r) is frequency of the word having rank r.

A. Tokenization and Porter Stemming

```
with open ('IR_1.txt',encoding="utf8") as fin:
data = re.sub('[^a-zA-Z-]',' ',fin.read())
tokens = word_tokenize(data.lower())
corpus = [ps.stem(token) for token in tokens]
```

B. Frequency and Rank relation

If the most frequent term (the) occurs f(r) times

- then the second most frequent term (of) occurs f(r)/2 times
- the third most frequent term (and) occurs f(r)/3 times ...

Equivalent: f(r)=K/r where K is a normalizing factor.

- log f(r) = log K - log r
- Linear relationship between log f(r) and log r

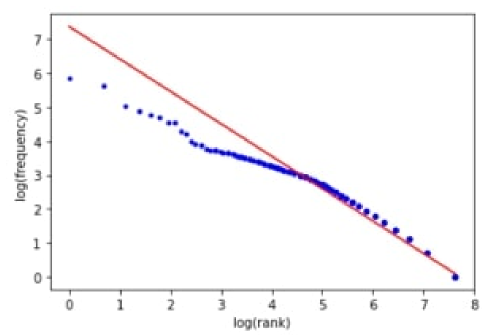


Fig. 1.

C. Word Cloud

The size of each word represents its frequency or relevance in a **Word cloud**, which is a data visualisation tool for visualising text data. A word cloud can be used to highlight key textual data points. Data from social networking websites is frequently analysed using word clouds.

Fig. 2.

D. Hadoop Map Reduce

Hadoop map reduce is the Apache Hadoop processing component. It processes data parallelly in distributed environment. Without removing stopwords and stemming, Map-Reduce on the Large dataset reveals that the same word with symbols attached has separate counts. We discovered the true counts of the words in the dataset after stemming and deleting symbols.

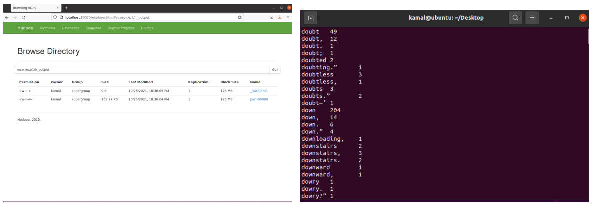


Fig. 3.

III. Conclusion

In Fig. 1 We can see that Frequency of word is inversely proportional to Rank. Hence Ziph’s law is Validated because the slope of the line is found to be negative in the plot of log (r) vs log (f(r)).

References

[1] [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176592/

Lab Report 2
BSBI and TF-IDF

Irum Hussain
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162005

Karre Kamal
Artificial Intelligence
IIIT Vadodara
Gandhinagar, Gujarat
202161004

Prashant Kumar
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162007

***Abstract*—BSBI divide the large corpus into a systematic matrix where terms and documents are rows and columns. TF-IDF is a method for calculating the number of words in a corpus We usually assign each word a score to indicate its significance in the document and corpus. This method is mainly used in information retrieval and text processing.**

I. Introduction

Indexing a huge corpus with limited mainframe resources is a major problem. We can't index the entire corpus because of memory constraints. Block Sort Based Indexing (BSBI) comes to tackle this problem. It creates an index of a large corpus using small blocks. We divide the corpus into equal-sized blocks in BSBI. Then we can do TF-IDF to how significat a word is to a document. We do inverse document frequency which measures the informativeness of term.

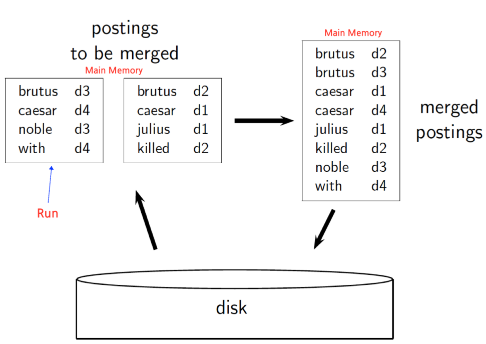


Fig. 1.

A. Porter Stemming and BSBI

We do Porter Stemming to reduce a word to its word stem. We are dividing the corpus into small blocks where each block contains word, document id and its corresponding term frequency. We are initially fixing a block size as a threshold to increase the main memory and cache optimization. If the block size exceeds the so far created posting list of block size will be updated in a new file and stored locally. In the same way we will be getting some files based on the corpus size. At last we will merge all the posting list files to a single file.

```
Pseudocode

def bsbi():
    current_block=0
    freq_dict = defaultdict(dict)
    for doc in Corpus:
        id = doc["id"]
        text = doc["body"]
        for word in text:
            word = porter.stem(word)
            if word not in freq_dict:
                current_block += 1
            if not freq_dict[word].__contains__(id):
                freq_dict[word].update({id:word_count})
                current_block += 1
        if current_block >= BLOCK_SIZE:
            Create Posting Lists using freq_dict
            word->(doc_id, tf)->(doc_id, tf)->
```

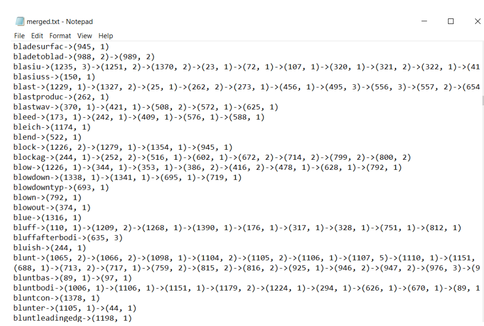


Fig. 2.

Metrics

- Precision tells us how many number of retrieved docs are relevant
- Recall tells us how many number of relevant docs are retrieved
- F1 score tells us weighted average of Precision and Recall

B. TF(Term Frequency)-IDF(Inverse Document Frequency)

The TF-IDF for a word in a document is calculated by multiplying two metrics:

- The term frequency of a word in a document (tf)
- The inverse document frequency of the word (idf)

So, if the word is very common and presents in many documents, this number will tends 0. Otherwise, it will tends 1.

TF-IDF score of a word in a document will be given by tf*idf. The higher the score, the more significant that term becomes in that particular document.

$$tfidf(t,d) = tf(t,d) * idf(t) \tag{1}$$

$$idf(t) = log[n/df(t)] + 1 \tag{2}$$

where n = no. of documents in the corpus.

II. Conclusion

For large corpus BSBI is a good solution to do indexing. BSBI is efficient in memory wise. In BSBI we do sorting 2 times. First we store Posting lists and then we do merging.

References

[1] https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Lab Report 3
Boolean Retrieval

Irum Hussain
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162005

Karre Kamal
Artificial Intelligence
IIIT Vadodara
Gandhinagar, Gujarat
202161004

Prashant Kumar
Data Science
IIIT Vadodara
Gandhinagar, Gujarat
202162007

***Abstract*—Explore Boolean Retrieval model and how to process Boolean queries and do Weighted Zone Scoring.**

I. Introduction

Any boolean expression can be answered using the Boolean Retrieval paradigm. Query is usually specified using operators from Boolean logic (AND, OR, NOT).It views each document as a set of terms. For any query evaluation, there are two possible results (TRUE and FALSE).. Boolean Retrieval is also known as exact-match retrieval.

- Assumption : All documents in the retrieved set are equivalent in terms of relevance.

II. Approach

A. Porter Stemming and BSBI

We do Porter Stemming to reduce a word to its word stem. We are dividing the corpus into small blocks where each block contains word, document id and its corresponding term frequency. We are initially fixing a block size as a threshold to increase the main memory and cache optimization. If the block size exceeds the so far created posting list of block size will be updated in a new file and stored locally. In the same way we will be getting some files based on the corpus size. At last we will merge all the posting list files to a single file.

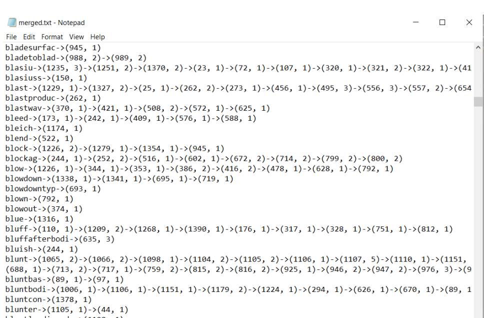


Fig. 1.

B. Boolean retrieval model and Zone Scoring

Usually boolean queries use AND, OR and NOT to join query terms

- Document matches condition or not. This tells us the Precision.

Perhaps the most basic model for constructing an IR system on the primary commercial retrieval tool for the past three decades.
Search systems that uses Boolean model till date:

- Emails, Library catalogs, Mac Spotlight

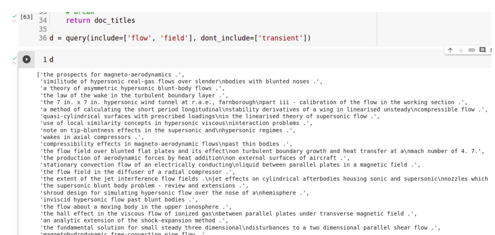


Fig. 2.

Zone Scoring

Given a Boolean query and a document, Weighted Zone Scoring computes a linear combination of zone scores, where each document's zone adds a Boolean value, and awards a score in the interval [0, 1] to the pair (q1, q2).

```
ZoneScore(q1, q2)
  float scores[N] = [0]
  constant g[ℓ]
  p1 ← postings(q1)
  p2 ← postings(q2)
  // scores[] is an array with a score entry for each document, initialized to zero.
  //p1 and p2 are initialized to point to the beginning of their respective postings.
  //Assume g[] is initialized to the respective zone weights.
  while p1 ≠ NIL and p2 ≠ NIL
  do if docID(p1) = docID(p2)
       then scores[docID(p1)] ← WeightedZone(p1, p2, g)
            p1 ← next(p1)
            p2 ← next(p2)
     else if docID(p1) < docID(p2)
            then p1 ← next(p1)
            else p2 ← next(p2)
  return scores
```

Fig. 3.

Let si be the Bollean score denoting a match (or absence thereof) between q and the i th zone. Let g1,g2,...,gn $\in [0,1]$ $such\ that$ $\sum_{i=1}^{n} gi = 1$.Then the weighted zone score is defined to be

$$\sum_{i=1}^{n} gi * si \in [0,1] \tag{1}$$

III. Conclusion

Results are very predictable and easy to explain to users. Boolean query operands can be of any document attribute. Documents can be quickly discarded from consideration from consideration in the scoring process, so more efficient.

References

[1] https://nlp.stanford.edu/IR-book/html/htmledition/weighted-zone-scoring-1.html

Lab Report 4
Vector Space Retrieval

Irum Hussain
202162005

Karre Kamal
202161004

Prashant Kumar
202162007

***Abstract*—The aim here is to install and experiment with Elasticsearch and Kibana.**

I. Introduction

Elasticsearch is an open source ,distributed, JSON based search and analytics engine which is capable of addressing a large number of use cases.Elasticsearch is based on Apache Lucene search engine and allows you to perform and combine different types of searches as needed, including structured, unstructured, geographic, and metric.Elasticsearch uses aggregation to zoom out to see trends and patterns in your data. It can quickly store, search, and analyze large amounts of data in near real time and get answers in milliseconds. It uses document-based structures instead of tables and schemas and has extensive REST APIs for storing and retrieving data. Basically, Elasticsearch can be thought of as a server that can process JSON requests and return JSON data.Kibana is a visualization layer built on Elasticsearch that allows users to analyze and visualize their data.This allows you to merge Elasticsearch data into multiple indexes and combine it with other SQL / NoSQL / REST API data sources to create visualizations with a user-friendly user interface.We run queries and using dev tools in kibana try to understand how search results are returned based on our query. We also do vector space retrieval where set of documents are represented as vectors and is fundamental to a large number of information retrieval ranging from scoring documents on a query, document classification and document clustering.

A. Similarity model in Elasticsearch

Elasticsearch calls the Lucene Practical Scoring Function. This function generates a relevance score that Elasticsearch uses to sort documents when data is requested.It is based on vector space retrieval model which uses tf-idf(term frequency-inverse document frequency).

1) A. Vector Space Model:

Document Freuency (d_f):The number of documents in the collection that contains a particular term .

Term Frequency(t_f): Counting the number of occurences of the term in the document.

Inverse document frequency (id_f): The logarithm of number of documents in the collection (or index) divided by the number of documents that contain the word. It is highest when a term occurs many times within a small collection of documents but is lowest when term occurs in all documents i.e. zero.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

Overlap Score measure: the score of a document is the sum over all query terms of the number of times each of the query terms occurs in a document. Now to quantify the

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}.$$

similarity between two documents we use cosine similarity which takes care of the length of the document.So cosine similarity between two documents d1 and d2 is shown below.

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$

Query boost:At search time users can specify boosts to each query, sub-query, and each query term, hence the contribution of a query term to the score of a document is multiplied by the boost of that query term.

coord-factor(q,d):A document may match a multi term query without containing all the terms of that query (this is correct for some of the queries), and users can further reward documents matching more query terms through a coordination factor, which is usually larger when more terms are matched.

Query normalization (queryNorm): Total of the squared weights of the query.

normalization (norm): The inverse square root of the number of terms in the field is used to calculate the field length normalisation (norm).

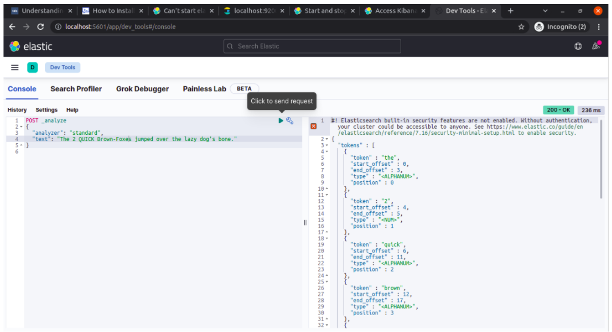
Lucene's Conceptual scoring formula: For query q and document d is defined as follows:

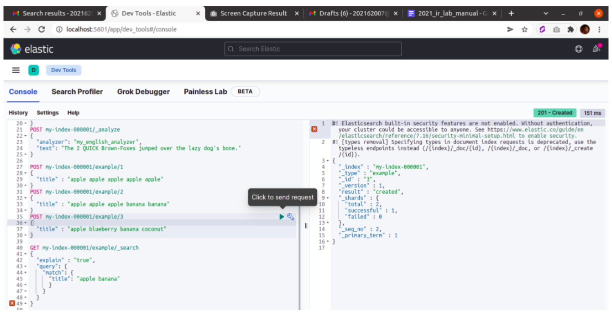
score(q,d) = queryNorm(q) coord(q,d) SUM(t_f(t ind),id_f(t),t.getBoost(),norm(t,d))(t in q)

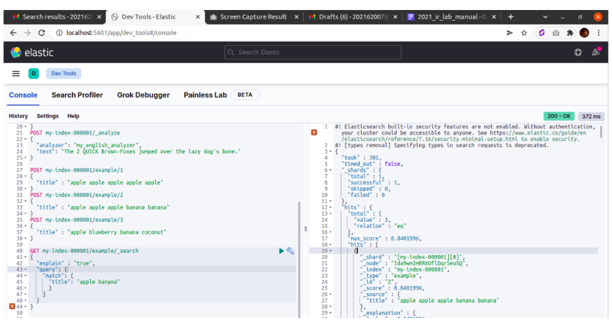
II. Results

We found that we could change the priority or ranking by specifying fields such as indexBoost and queryBoost in

Elasticsearch. Kibana helps in visualisation by providing easy-to-understand graphic representation of our dataset.







CONCLUSION

The combination of Elasticsearch and Kibana makes it a very powerful tool for information retrieval. They can be together used to analyse data quickly and efficiently.

REFERENCES

1.https://www.elastic.co/blog/found-similarity-in-elasticsearch
2.https://lucene.apache.org/core/451/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html
3. Manning, C. D., Raghavan, P., Schu tze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.

Lab Report 5
Text Classification

Irum Hussain
202162005

Karre Kamal
202161004

Prashant Kumar
202162007

***Abstract*—The large volume of unstructured documents and text are difficult to deal with. Obtaining specific information from this large set of data takes a lot of time. One of the techniques is text classification. Text classification task is to assign a document to one or more category. Text classifiers can be used to organize, structure, and categorize any data.**

I. Introduction

The growth of the Internet has had a major impact on data generation. Most of the world's data is in text format.There is need to access and use this data efficiently and easily hence, text classification is widely studied problem in research community.Text classification is the process of classifying a text document into a fixed number of predefined classes. Text classification applications include spam filtering, email routing, sentiment analysis, voice recognition and more.It is a supervised learning approach in which we predict the class label of incoming document based on the training model built from a training set of documents labelled with classes.The most important step of the text classification pipeline is choosing the best classifier. So we are going to understand Naive Bayes classifier, Rocchio classifier, K Nearest Neighbor classifier by training and testing on 20 Newsgroup dataset.

A. Classification Algorithms

1) A. Naive Bayes Classification:
Naive Bayes classification is probabilistic machine learning model based on the Bayes theorem.The probability of a document d being in class c is P(t_k—c) is the conditional

$$P(c|d) \propto P(c) \prod_{1\leq k\leq n_d} P(t_k|c)$$

probability of term tk occurring in a document of class c.We interpret P(t_k—c) as a measure of how much evidence t_k contributes that c is the correct class. P(c) is the prior probability of a document occurring in class c.

```
TrainMultinomialNB(ℂ, 𝔻)
V ← ExtractVocabulary(𝔻)
N ← CountDocs(𝔻)
for each c ∈ ℂ
do N_c ← CountDocsInClass(𝔻, c)
    prior[c] ← N_c / N
    text_c ← ConcatenateTextOfAllDocsInClass(𝔻, c)
    for each t ∈ V
    do T_ct ← CountTokensOfTerm(text_c, t)
    for each t ∈ V
    do condprob[t][c] ← (T_ct + 1) / Σ_t'(T_ct' + 1)
return V, prior, condprob

ApplyMultinomialNB(ℂ, V, prior, condprob, d)
W ← ExtractTokensFromDoc(V, d)
for each c ∈ ℂ
do score[c] ← log prior[c]
    for each t ∈ W
    do score[c] += log condprob[t][c]
return arg max_{c∈ℂ} score[c]
```

2) B. K Nearest Neighbour: KNN is a supervised learning algorithm which determines the decision boundary locally.It is based on the hypothesis that it expects a test document d to have the same label as the training documents located in the local region surrounding d. Here K is the number of nearest neighbours to be considered. For a new data point in vector space, kNN looks for k points which are nearest to that data by selecting distance measure such as euclidean distance, manhattan distance and minkowski distance. These K points then becomes the nearest neighbours of that new data point and is assigned to the class which represents the most points among those k neighbours. For finding most optimum k value, first differentiate the training and validation dataset and then plot the validation error curve.

```
Train-kNN(ℂ, 𝔻)
𝔻' ← Preprocess(𝔻)
k ← Select-k(ℂ, 𝔻')
return 𝔻', k

Apply-kNN(ℂ, 𝔻', k, d)
S_k ← ComputeNearestNeighbors(𝔻', k, d)
for each c_j ∈ ℂ
do p_j ← |S_k ∩ c_j| / k
return arg max_j p_j
```

3) C. Rocchio Classifier: One of the main task in vector space is to create good class boundaries between the classes and rocchio classification is one of the ways . It uses centroids to define the boundaries.It is a form of relevance feedback where the average of the relevant documents, corresponds to the most important component of the Rocchio vector in relevance feedback ,is the centroid of of relevant documents.

```
TrainRocchio(ℂ, 𝔻)
for each c_j ∈ ℂ
do D_j ← {d : ⟨d, c_j⟩ ∈ 𝔻}
    μ⃗_j ← (1/|D_j|) Σ_{d∈D_j} v⃗(d)
return {μ⃗_1, ..., μ⃗_J}

ApplyRocchio({μ⃗_1, ..., μ⃗_J}, d)
return arg min_j |μ⃗_j − v⃗(d)|
```

II. Approach

Dataset: The 20 newsgroups dataset comprises of around 18000 newsgroups posts on 20 topics split into two subsets. One is for training and the other one is for testing. The split between the tain and test data set is based upon a messages posted before and after a specific date. Each record in the corpus is actually a text file. Then we create tokens using CountVectorizer() which counts the number of times a word appears in each file giving us a count matrix. Then we transform our matrix using TfidfTransformer() which transforms the count matrix into Tf-idf representaion.The goal of using tf-idf instead of the count matrix is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence e less informative than features that occur in a small portion of the training corpus. This comprises of our training dataset.

Naive Bayes:We use MultinomialNB (the multinomial Naive Bayes classifier) present in sklearn.naive_bayes which is suitable for discrete classification. Scikit-learn has a class called Pipeline, which allows us to add the functions that we want to use on our input data into the pipeline for the classifier.Then we take the training dataset and feed it in the classifier.We then get a matrix from it for which we calculate the accuracy, precision and recall using the test set.

Rocchio Classifier:We use NearestCentroid present in sklearn.neighbours.nearestCentroid where each class is represented by its centroid, with test samples classified to the class with nearest centroid..Scikit-learn has a class called Pipeline, which allows us to add the functions that we want to use on our input data into the pipeline for the classifier.Then we take the training dataset and feed it in the classifier.We then get a matrix from it for which we calculate the accuracy, precision and recall using the test set.

K-Nearest Neighbour Classifier: We use KNeighborsClassifier present in sklearn.neighbors with k and weights as parameters.Scikit-learn has a class called Pipeline, which allows us to add the functions that we want to use on our input data into the pipeline for the classifier.Then we take the training dataset and feed it in the classifier.We then get a matrix from it for which we calculate the accuracy, precision and recall using the test set.

III. Results

The precision , recall and f1 scores for few categories of 20 newsgroup dataset for naive bayes, rocchio and K nearest neighbour classifier have been computed.

```
Accuracy 0.9357565511411665
                 precision    recall  f1-score   support

sci.electronics       0.95      0.90      0.92       393
        sci.med       0.94      0.94      0.94       396
      sci.space       0.92      0.97      0.94       394

       accuracy                           0.94      1183
      macro avg       0.94      0.94      0.94      1183
   weighted avg       0.94      0.94      0.94      1183

(None, array([[353,  14,  26],
        [ 17, 371,   8],
        [  3,   8, 383]]))
```

Fig. 1. Naive Bayes Classifier

```
Accuracy 0.8571428571428571
                 precision    recall  f1-score   support

sci.electronics       0.84      0.80      0.82       393
        sci.med       0.85      0.84      0.85       396
      sci.space       0.88      0.93      0.90       394

       accuracy                           0.86      1183
      macro avg       0.86      0.86      0.86      1183
   weighted avg       0.86      0.86      0.86      1183

array([[313,  47,  33],
       [ 44, 334,  18],
       [ 14,  13, 367]])
```

Fig. 2. K Nearest Neighbour Classifier

```
Accuracy 0.801352493660186
                 precision    recall  f1-score   support

sci.electronics       0.66      0.91      0.76       393
        sci.med       0.88      0.63      0.74       396
      sci.space       0.96      0.86      0.91       394

       accuracy                           0.80      1183
      macro avg       0.83      0.80      0.80      1183
   weighted avg       0.83      0.80      0.80      1183

array([[359,  24,  10],
       [142, 250,   4],
       [ 45,  10, 339]])
```

Fig. 3. Rocchio classifier

Conclusion

Classification tasks are one of the most essential issues in machine learning. As text And the document dataset grows year by year, having a better document categorization system for this growing information requires learning and discerning these text classification algorithms.The classification of the news articles could be expanded to multi-label text articles, as there are many news articles that do not belong to a single category and are instead a mixture of various categories.

Refereneces

1.Manning, C. D., Raghavan, P., Schu tze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.

2.https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

3.Ali, N., Neagu, D. Trundle, P. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. SN Appl. Sci. 1, 1559 (2019). https://doi.org/10.1007/s42452-019-1356-9

4.https://www.researchgate.net/publication/228961211_Explaining_naiveBayes_classifications

Lab Report 6
Latent Semantic Indexing

Irum Hussain
202162005

Karre Kamal
202161004

Prashant Kumar
202162007

***Abstract*—To help analyse relationships between terms in the documents or between the documents in the corpus we aim to experiment with latent semantic indexing based on singular value decomposition.**

I. Introduction

Latent Semantic Indexing (LSI) is a retrieval and indexing mechanism which uses a mathematical technique called Singular Value Decomposition (SVD) to identify patterns of relationships between terms and concepts in unstructured collections of text data. Singular Value Decomposition is a factorization of a real or complex matrix into simpler matrices where each matrix is a coloumn vector times row vector. The SVD chooses rank pieces in order of importance.

A. Singular Value Decomposition

Singular value decomposition decomposes a matrix into three components where two of these components are representations of word and documets in the approximated vector space while the third components provides weights to these vectors.The decomposition of a matrix is useful when the matrix is not of full rank. That is, the rows or columns of the matrix are linearly dependent.

B. Latent Semantic Indexing

The latent semantic indexing tries to overcome the problems of polysemy(same word but multiple meanings) and synonymy(two words but same meaning).It takes the low rank term document matrix from SVD which gives a new representation of each document in the corpus.We then cast queries into this low rank representation as well and compute similarities between document and query.if two terms are used in similar contexts or have similar meaning then they will have similar vectors in the reduced dimension representation. It is thus a method for dimensionality reduction where we take objects from high dimensional space and cast them in low dimensional space.The projection into the latent semantic space is such that when measured by the sum of the squares of the differences the representations in the original space are changed as little as possible.

II. Approach

A large sparse term-document matrix A is created from the corpus which is not a full rank matrix.Let n be the rank of txd matrix A where t is the number of terms and d is the number of documents SVD take this matrix A and in lower dimension A' "distance" between the two matrices as measured by the 2-norm is minimized: .Then we apply

$$\Delta = \left\| A - \hat{A} \right\|_2$$

Singular value decomposition as T and D have orthonormal

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^T$$

coloumns where (TT^T) = (D^TD) = I and rank(A_k) =r where r¡n.

$$\hat{A}_{t\times k} = T_{t\times k} S_{k\times k} (D_{d\times k})^T$$

After constructing a low-rank approximation A' to the term-document matrix, for a value of r that is far smaller than the original rank of A. We use this new r-dimensional representation to compute similarities and map the query vector q to q'.q' is a simple vector in space of terms. The query vector

$$\hat{q} = q^T T_{t\times k} S^{-1}{}_{k\times k}$$

is then compared to all document vectors, and the documents ranked by their similarity (nearness) to the query. One measure of similarity is the cosine similarity between the query vector and document vector. Now if a new document arrives then we either recompute the SVD of a new term-document matrix or fold the new terms and documents when they arrive.Folding new documents and terms require less time and space so new documents can be folded into this LSI representation thus we keep on incrementally adding new documents to LSI representation as follows.

$$A = TSD^T$$

$$T^T A = T^T TSD^T$$

$$T^T A = SD^T$$

III. Results

```
#finding the most similar document for the query vector by taking the dot product between query and
#each column in the vt matrix and normalize them
similarities = []
for i in range(len(vt)):
    similarities.append(list(np.dot(q,vt[:,i])/(np.linalg.norm(q)*np.linalg.norm(vt[:,i])))[0])

print('Most similar document:')
idx = similarities.index(max(similarities))
print(idx)
corpus[idx]
```

```
Most similar document:
1030
'note on creep buckling of columns .'
```

Conclusion

In a nutshell, Latent semantic indexing is able to overcome the problem of synonymy and polysemy. Also since its a mathematical approach therefore it is independent of the languages. It is also tolerant to noise data which is misspelled words,unreadable characters etc. Though it has its own disadvantages such as a large sparse matrix , time consuming but it still has a wide scope in future.

References

1.Manning, C. D., Raghavan, P., Schu tze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.

2.Strang, G. (1976). Linear algebra and its applications. New York: Academic Press

3.G.W. Furnas et al., Information retrieval using a singular value decomposition model of latent semantic structure, in Proceedings of SIGIR, 1988.

4.Deerwester, Dumais, Furnas, Lanouauer, and Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science, 41 (1990), pp. 391-407.

Lab Report 8
Support Vector Machine

Irum Hussain
202162005

Karre Kamal
202161004

Prashant Kumar
202162007

***Abstract*—The aim of this lab is to classify two classes which are non linearly separable via support vector machines. Also learn and experiment with kernel SVM**

I. Introduction

Support Vector Machine is a supervised learning used for regression and classification.Support Vector Machine (SVM) tries to maximize predictive accuracy while automatically avoiding over fitting to the data. SVM is based on vector space model where we try to find a decision bondary between classes that is maximally far away from any of the point of the training dataset. A classifier with a large margin ensures high certainty classification decisions.Non-linear data set are difficult to classify using a linear hyperplane.Thus we want to apply a function that projects / transforms the data in such a manner that the data becomes linearly separable. The idea of the kernel SVM is to enable operations to be performed in the input space rather than the high dimensional feature space. We want the function to perform mapping of the attributes of the input space to the feature space.

A. Theory

1) A. Support Vector MAchine:
For a two-class, linearly separable training data sets, there are lots of possible linear separators but only one of them achieves maximum separation.To choose among all the hyperplanes(linear separators), we specify the intercept term b. Because the hyperplane is perpendicular to the normal vector w, all points x' on the hyperplane satisfy the equation

$$\vec{w}^{\mathrm{T}}\vec{x} = -b.$$

Now for a set of training data points D(point x_i ,class label y_i) the two data classes are +1 and 1 and the intercept term is b The linear classifier is then represented as :

$$f(\vec{x}) = \mathrm{sign}(\vec{w}^{\mathrm{T}}\vec{x} + b)$$

Each points distance from the hyperplane is

$$r_i = y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b)/|\vec{w}|,$$

$$\rho = 2/|\vec{w}|$$

The geometric margin is Now we get a quadratic optimization problem and need to solve it for w and b.Thus we optimize the quadratic function with linear constraints and construct a dual problem where a Langlier's multiplier i is associated. SVM as a minimisation problem is to find w and b such that:

- $\frac{1}{2}\vec{w}^{\mathrm{T}}\vec{w}$ is minimized, and
- for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^{\mathrm{T}}\vec{x}_i + b) \geq 1$

2) B. kernel SVM: Now for data that cannot be classified by a linear classifier, the idea is to map the original feature space to some higher-dimensional feature space where the training set is separable while preserving the relations between the data points . Thus a kernel function K is a function that corresponds to a dot product in expanded feature space. A kernel function K is continuous, symmetric, and a positive definite gram matrix.

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^{\mathrm{T}}\phi(\vec{x}_j).$$

B. Approach

For the first part , we randomly generate dataset which follows a quadratic distribution. Then we split this dataset into (20:80) test and training datasets. Then we use polynomial kernel svm that represents the similarity of vectors in a feature space over polynomials of the original variables enabling learning of non-linear models.

Dataset: The 20 newsgroups dataset comprises of around 18000 newsgroups posts on 20 topics split into two subsets. One is for training and the other one is for testing. The split between the tain and test data set is based upon a messages posted before and after a specific date. Each record in the corpus is actually a text file. Then we create tokens using CountVectorizer() which counts the number of times a word appears in each file giving us a count matrix. Then we transform our matrix using TfidfTransformer() which transforms the count matrix into Tf-idf representaion.The goal of using tf-idf instead of the count matrix is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence e less informative than features that occur in a small portion of the training corpus. This comprises of our training dataset.

Then we use SGD classifier from the sklearn.linearmodel which implements regularized linear models with stochastic gradient descent (SGD) learning. The gradient of the loss is estimated each sample at a time and the model is updated with a decreasing strength schedule (aka learning rate). SGD allows minibatch learning via the partial$_fitmethod.$

We then get a matrix from it for which we calculate the accuracy, precision and recall using the test set.

II. Results

For part 1 the results are as shown below:

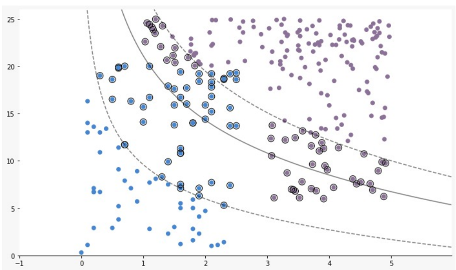


Fig. 1. SVM plot

```
              precision    recall  f1-score   support

           0       0.66      0.91      0.76       393
           1       0.88      0.63      0.74       396
           2       0.96      0.86      0.91       394

    accuracy                           0.80      1183
   macro avg       0.83      0.80      0.80      1183
weighted avg       0.83      0.80      0.80      1183
```

Fig. 2. precision ,recall and f1 score

For part 2 the precision, recall and f1 score for the same categories used in naive bayes have been done for SVM as well

```
1 SVM(categories)

*** SVM Model ***
Newsgroup Categories :  ['sci.med', 'sci.space', 'sci.electronics']
Accuracy : 93.99830938292477%
                 precision    recall  f1-score   support

sci.electronics       0.89      0.97      0.93       393
        sci.med       0.95      0.91      0.93       396
      sci.space       0.98      0.94      0.96       394

       accuracy                           0.94      1183
      macro avg       0.94      0.94      0.94      1183
   weighted avg       0.94      0.94      0.94      1183

Confusion Matrix :
 [[381   9   3]
 [ 32 360   4]
 [ 13  10 371]]
```

Fig. 3. Newsgroup SVM accuracy

Conclusion

Support Vector Machines acts is one of the best approach to data modeling and classification. The kernel mapping provides a common base for most of the commonly employed model architectures based on different datasets for comparisons to be performed and is able to classify non linear datasets. The SVM is dependent on very few support vectors leading to less memory and concise models . It is affected only by the points on the margin and works well with high dimensional data.Kernel SVM are also very versatile and able to classify large no of databases of many types.

References

1.

1.Manning, C. D., Raghavan, P., Schu tze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.

2.Strang, G. (1976). Linear algebra and its applications. New York: Academic Press

3.https://en.wikipedia.org/wiki/Polynomial_kernel

4.https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

5. Tutorial slides by Andrew Moore. Http://www.cs.cmu.edu/ awm

LAB CODE LINK:

https://github.com/Kamal-prog-code/IR-Lab