Indian Institute of Information Technology Vadodara

CS 331 Information Retrieval

End Sem Lab Report

Group Member:

Yash Bhimani - 201951042

Devang Delvadiya - 201951055

Manthan Ghasadiya - 201951065

Rushi Javiya - 201951074

Course Co-ordinator:

Dr. Pratik Shah

Contents

I. **Experiment 4 : Vector Space Retrieval**

Objective : Install and Quick hand on Elasticsearch and Kibana.

A. *Introduction*

The Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. The vector space model is an algebraic representation of text documents (and other things) as vectors (such as index terms). It's utilised in information filtering, retrieval, indexing, and ranking relevancy. The SMART Information Retrieval System was the first to employ it.

In the Vector Space Model, Each document or query is a N-dimensional vector where N is the number of distinct terms over all the documents and queries.The ith index of a vector contains the score of the ith term for that vector. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector.

B. *Cosine Similarity*

To compute the similarity(closeness) between two vector,We calculate cosine angle between each document vector and the query vector. In order to compute the similarity between two vectors : d1(document 1), d2(document 2) the cosine similarity is used. Here also, To avoid the effect of document length as possible, the standard way of quantifying the similarity between two documents d1 and d2 is to compute the cosine similarity of their vector representations $\vec{V}(d1)$ and $\vec{V}(d2)$.[3]

$$sim(d1, d2) = \frac{\vec{V}(d1) \cdot \vec{V}(d2)}{\left|\vec{V}(d1)\right| \left|\vec{V}(d2)\right|} \tag{1}$$

C. *Approach*

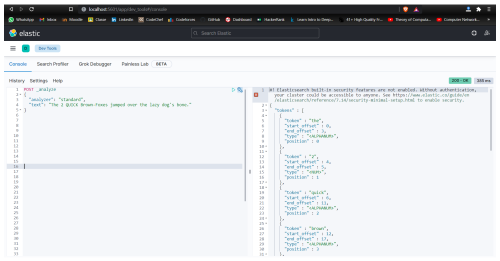
Each document is represented as vector in vector space. We denote by $\vec{V}(d)$ the vector derived from document d, with one component in the vector for each dictionary term. The set of documents in a corpus then may be viewed as a set of vectors in a vector space, in which there is one axis for each term. To consider similarity between two vector(document) in this vector space is defined by cosine similarity. There is a far more compelling reason to represent documents as vectors : we can also view a query as a vector.

D. *Why Elasticsearch/Kibana*

This vector space retrieval can be done easily using elasticsearch and kibana. we have taken few examples to see their practical representation. We have taken English Hindi and Gujarati languages.[5]

E. *Conclusion*

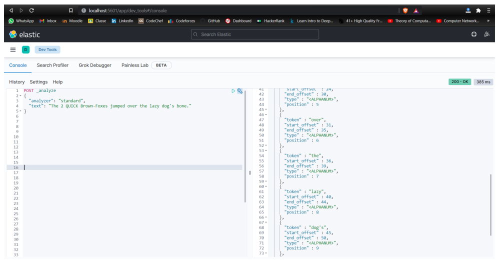
we can briefly summarize that Elasticsearch is at its core a search engine, whose underlying architecture and components makes it fast and scalable. This can be used for many uses cases including search, analytics, and data processing and storage.Elasticsearch creates a data structure known as the Inverted Index which is primarily responsible for the lightning-fast search result retrieval. We first make index text analyzer and then analyze text using index. An index is the highest level entity that you can query against in Elasticsearch. Using Kibana We can analyze this all in real time.[1]



```
POST _analyze
{
  "analyzer": "standard",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication,
  your cluster could be accessible to anyone. See https://www.elastic.co/guide/en
  /elasticsearch/reference/7.14/security-minimal-setup.html to enable security.
{
  "tokens" : [
    {
      "token" : "the",
      "start_offset" : 0,
      "end_offset" : 3,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "2",
      "start_offset" : 4,
      "end_offset" : 5,
      "type" : "<NUM>",
      "position" : 1
    },
    {
      "token" : "quick",
      "start_offset" : 6,
      "end_offset" : 11,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "brown",
      "start_offset" : 12,
      "end_offset" : 17,
      "type" : "<ALPHANUM>",
      "position" : 3
    },
```

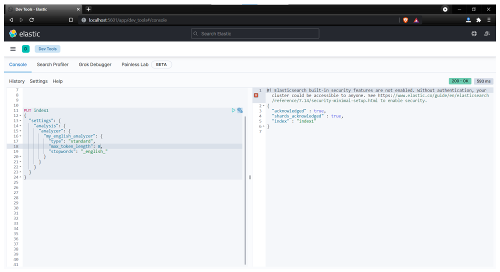
Fig. 1. Example of Defining the standard analyzers



```
POST _analyze
{
  "analyzer": "standard",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."
}
```

```
      "start_offset" : 24,
      "end_offset" : 30,
      "type" : "<ALPHANUM>",
      "position" : 5
    },
    {
      "token" : "over",
      "start_offset" : 31,
      "end_offset" : 35,
      "type" : "<ALPHANUM>",
      "position" : 6
    },
    {
      "token" : "the",
      "start_offset" : 36,
      "end_offset" : 39,
      "type" : "<ALPHANUM>",
      "position" : 7
    },
    {
      "token" : "lazy",
      "start_offset" : 40,
      "end_offset" : 44,
      "type" : "<ALPHANUM>",
      "position" : 8
    },
    {
      "token" : "dog's",
      "start_offset" : 45,
      "end_offset" : 50,
      "type" : "<ALPHANUM>",
      "position" : 9
    },
```

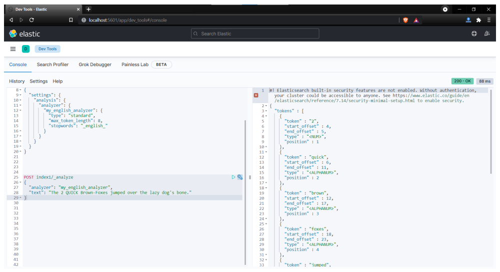
Fig. 2. Output of Defining the standard analyzers



```
PUT index1
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_english_"
        }
      }
    }
  }
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your
  cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch
  /reference/7.14/security-minimal-setup.html to enable security.
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "index1"
}
```

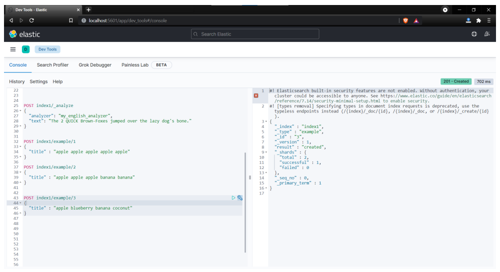
Fig. 3. Defining index1 in Elasticsearch and receiving acknowledgement



```
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_english_"
        }
      }
    }
  }
}



POST index1/_analyze
{
  "analyzer": "my_english_analyzer",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication,
  your cluster could be accessible to anyone. See https://www.elastic.co/guide/en
  /elasticsearch/reference/7.14/security-minimal-setup.html to enable security.
{
  "tokens" : [
    {
      "token" : "2",
      "start_offset" : 4,
      "end_offset" : 5,
      "type" : "<NUM>",
      "position" : 1
    },
    {
      "token" : "quick",
      "start_offset" : 6,
      "end_offset" : 11,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "brown",
      "start_offset" : 12,
      "end_offset" : 17,
      "type" : "<ALPHANUM>",
      "position" : 3
    },
    {
      "token" : "foxes",
      "start_offset" : 18,
      "end_offset" : 23,
      "type" : "<ALPHANUM>",
      "position" : 4
    },
    {
      "token" : "jumped",
```

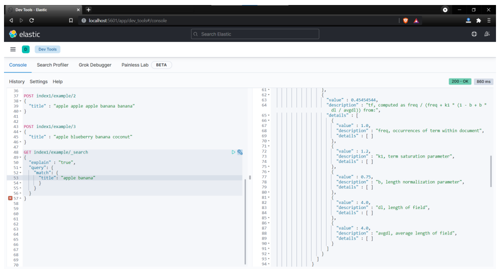
Fig. 4. Analyzing text using index and given text



```
POST index1/_analyze
{
  "analyzer": "my_english_analyzer",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."
}


POST index1/example/1
{
  "title" : "apple apple apple apple apple"
}

POST index1/example/2
{
  "title" : "apple apple apple banana banana"
}


POST index1/example/3
{
  "title" : "apple blueberry banana coconut"
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your
  cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch
  /reference/7.14/security-minimal-setup.html to enable security.
#! [types removal] Specifying types in document index requests is deprecated, use the
  typeless endpoints instead (/{index}/_doc/{id}, /{index}/_doc, or /{index}/_create/{id}
  ).
{
  "_index" : "index1",
  "_type" : "example",
  "_id" : "3",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

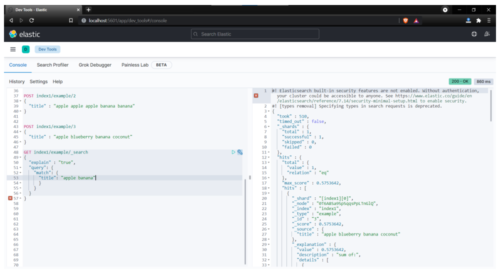
Fig. 5. Posting title of 3 queries into elasticsearch



```
POST index1/example/2
{
  "title" : "apple apple apple banana banana"
}


POST index1/example/3
{
  "title" : "apple blueberry banana coconut"
}

GET index1/example/_search
{
  "explain" : "true",
  "query": {
    "match": {
      "title": "apple banana"
    }
  }
}
```

```
                },
                {
                  "value" : 0.45454544,
                  "description" : "tf, computed as freq / (freq + k1 * (1 - b + b *
                    dl / avgdl)) from:",
                  "details" : [
                    {
                      "value" : 1.0,
                      "description" : "freq, occurrences of term within document",
                      "details" : [ ]
                    },
                    {
                      "value" : 1.2,
                      "description" : "k1, term saturation parameter",
                      "details" : [ ]
                    },
                    {
                      "value" : 0.75,
                      "description" : "b, length normalization parameter",
                      "details" : [ ]
                    },
                    {
                      "value" : 4.0,
                      "description" : "dl, length of field",
                      "details" : [ ]
                    },
                    {
                      "value" : 4.0,
                      "description" : "avgdl, average length of field",
                      "details" : [ ]
                    }
                  ]
                }
              ]
            }
```

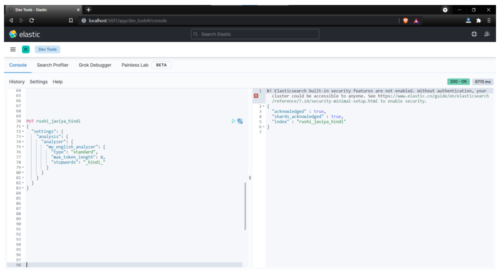
Fig. 6. Testing with different queries into elasticsearch



```
POST index1/example/2
{
  "title" : "apple apple apple banana banana"
}


POST index1/example/3
{
  "title" : "apple blueberry banana coconut"
}

GET index1/example/_search
{
  "explain" : "true",
  "query": {
    "match": {
      "title": "apple banana"
    }
  }
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication,
  your cluster could be accessible to anyone. See https://www.elastic.co/guide/en
  /elasticsearch/reference/7.14/security-minimal-setup.html to enable security.
#! [types removal] Specifying types in search requests is deprecated.
{
  "took" : 510,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 0.5753642,
    "hits" : [
      {
        "_shard" : "[index1][0]",
        "_node" : "0T6A8Sa9SpSqqsPpLTnGlQ",
        "_index" : "index1",
        "_type" : "example",
        "_id" : "3",
        "_score" : 0.5753642,
        "_source" : {
          "title" : "apple blueberry banana coconut"
        },
        "_explanation" : {
          "value" : 0.5753642,
          "description" : "sum of:",
          "details" : [
            {
```

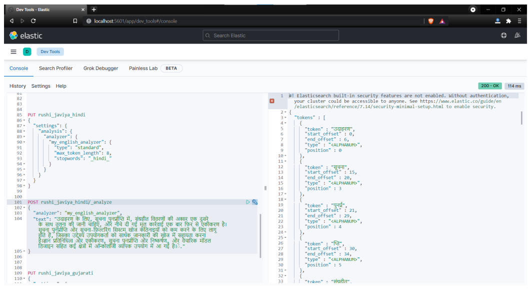
Fig. 7. Testing with different queries into elasticsearch



```
PUT rushi_javiya_hindi
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_hindi_"
        }
      }
    }
  }
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your
  cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch
  /reference/7.14/security-minimal-setup.html to enable security.
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "rushi_javiya_hindi"
}
```

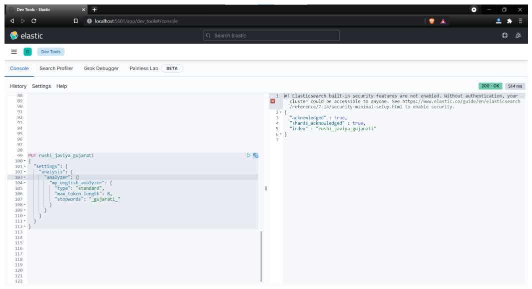
Fig. 8. Defining standard analyzer for Hindi language



```
PUT rushi_javiya_hindi
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_hindi_"
        }
      }
    }
  }
}


POST rushi_javiya_hindi/_analyze
{
  "analyzer": "my_english_analyzer",
  "text": "उदाहरण के लिए, सूचना पुनर्प्राप्ति में, संग्रहीत विवरणों की अक्सर एक दूसरे
    के साथ तुलना की जानी चाहिए, और नीचे दी गई मूल कार्रवाई एक बार फिर से एकीकरण है।
    सूचना पुनर्प्राप्ति और सूचना-फ़िल्टरिंग सिस्टम खोज कठिनाइयों को कम करने के लिए लागू
    होते हैं, जिसका उद्देश्य उपयोगकर्ता को सार्थक जानकारी की खोज में सहायता करना
    है।ज्ञान प्रतिनिधित्व और एकीकरण, सूचना पुनर्प्राप्ति और निष्कर्षण, और वैचारिक मॉडल
    डिजाइन सहित कई क्षेत्रों में ऑन्कोलॉजी व्यापक उपयोग में आ गई है।ठे."
}


PUT rushi_javiya_gujarati
{
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication,
  your cluster could be accessible to anyone. See https://www.elastic.co/guide/en
  /elasticsearch/reference/7.14/security-minimal-setup.html to enable security.
{
  "tokens" : [
    {
      "token" : "उदाहरण",
      "start_offset" : 0,
      "end_offset" : 6,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "सूचना",
      "start_offset" : 15,
      "end_offset" : 20,
      "type" : "<ALPHANUM>",
      "position" : 3
    },
    {
      "token" : "पुनर्प्र",
      "start_offset" : 21,
      "end_offset" : 29,
      "type" : "<ALPHANUM>",
      "position" : 4
    },
    {
      "token" : "प्ति",
      "start_offset" : 30,
      "end_offset" : 34,
      "type" : "<ALPHANUM>",
      "position" : 5
    },
    {
      "token" : "संग्रहीत",
```

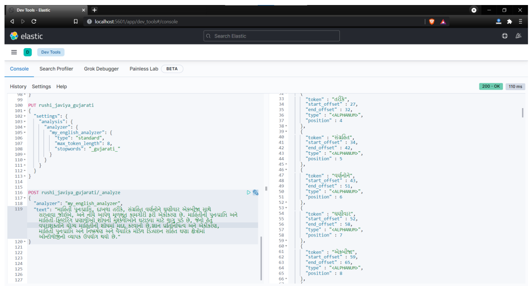
Fig. 9. Analyzing text using indexer and given text in Hindi



```
PUT rushi_javiya_gujarati
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_gujarati_"
        }
      }
    }
  }
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your
  cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch
  /reference/7.14/security-minimal-setup.html to enable security.
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "rushi_javiya_gujarati"
}
```

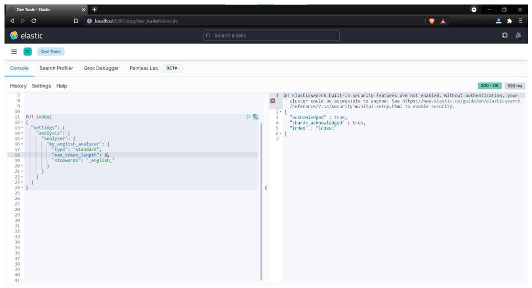
Fig. 10. Defining standard analyzer for Gujarati language



```
}

PUT rushi_javiya_gujarati
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_gujarati_"
        }
      }
    }
  }
}


POST rushi_javiya_gujarati/_analyze
{
  "analyzer": "my_english_analyzer",
  "text": "માહિતી પુનઃપ્રાપ્તિ, દાખલા તરીકે, સંગ્રહિત વર્ણનોને ઘણીવાર એકબીજા સાથે
    સરખાવવા જોઈએ, અને નીચે આપેલ મૂળભૂત કામગીરી ફરી એકીકરણ છે. માહિતીની પુનઃપ્રાપ્તિ અને
    માહિતી-ફિલ્ટરિંગ પ્રણાલીઓ શોધની મુશ્કેલીઓને ઘટાડવા માટે લાગુ પડે છે, જેનો હેતુ
    વપરાશકર્તાને યોગ્ય માહિતીની શોધમાં મદદ કરવાનો છે.જ્ઞાન પ્રતિનિધિત્વ અને એકીકરણ,
    માહિતી પુનઃપ્રાપ્તિ અને નિષ્કર્ષણ અને વૈચારિક મોડેલ ડિઝાઇન સહિત ઘણા ક્ષેત્રોમાં
    ઓન્ટોલોજીનો વ્યાપક ઉપયોગ થયો છે."
}
```

```
      "token" : "તરીકે",
      "start_offset" : 27,
      "end_offset" : 32,
      "type" : "<ALPHANUM>",
      "position" : 4
    },
    {
      "token" : "સંગ્રહિત",
      "start_offset" : 34,
      "end_offset" : 42,
      "type" : "<ALPHANUM>",
      "position" : 5
    },
    {
      "token" : "વર્ણનોને",
      "start_offset" : 43,
      "end_offset" : 51,
      "type" : "<ALPHANUM>",
      "position" : 6
    },
    {
      "token" : "ઘણીવાર",
      "start_offset" : 52,
      "end_offset" : 58,
      "type" : "<ALPHANUM>",
      "position" : 7
    },
    {
      "token" : "એકબીજા",
      "start_offset" : 59,
      "end_offset" : 65,
      "type" : "<ALPHANUM>",
      "position" : 8
    },
```

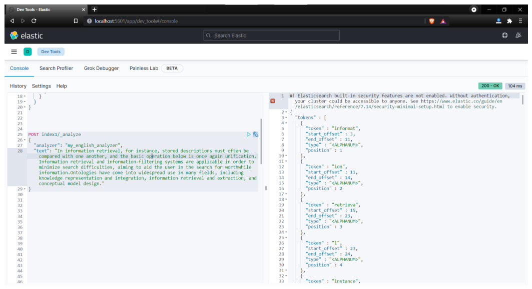
Fig. 11. Analyzing text using indexer and given text in Gujarati



```
PUT index1
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_english_analyzer": {
          "type": "standard",
          "max_token_length": 8,
          "stopwords": "_english_"
        }
      }
    }
  }
}
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your
  cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch
  /reference/7.14/security-minimal-setup.html to enable security.
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "index1"
}
```

Fig. 12. Defining standard analyzer for English language



```
    }
  }
}



POST index1/_analyze
{
  "analyzer": "my_english_analyzer",
  "text": "In information retrieval, for instance, stored descriptions must often be
    compared with one another, and the basic operation below is once again unification.
    Information retrieval and information-filtering systems are applicable in order to
    minimize search difficulties, aiming to aid the user in the search for worthwhile
    information.Ontologies have come into widespread use in many fields, including
    knowledge representation and integration, information retrieval and extraction, and
    conceptual model design."
}
```

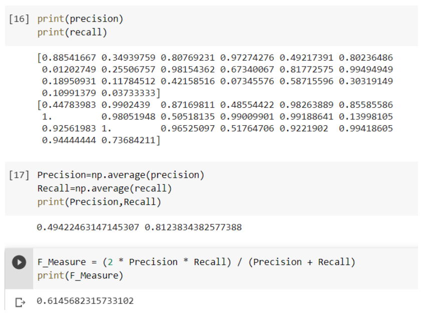
```
#! Elasticsearch built-in security features are not enabled. Without authentication,
  your cluster could be accessible to anyone. See https://www.elastic.co/guide/en
  /elasticsearch/reference/7.14/security-minimal-setup.html to enable security.
{
  "tokens" : [
    {
      "token" : "informat",
      "start_offset" : 3,
      "end_offset" : 11,
      "type" : "<ALPHANUM>",
      "position" : 1
    },
    {
      "token" : "ion",
      "start_offset" : 11,
      "end_offset" : 14,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "retrieva",
      "start_offset" : 15,
      "end_offset" : 23,
      "type" : "<ALPHANUM>",
      "position" : 3
    },
    {
      "token" : "l",
      "start_offset" : 23,
      "end_offset" : 24,
      "type" : "<ALPHANUM>",
      "position" : 4
    },
    {
      "token" : "instance",
```

Fig. 13. Analyzing text using indexer and given text in English

II. **LAB-5 | Text classification algorithms**

A. *Rocchio Algorithm*

1) Introduction: For document vectors, there are three classifier models. The Rocchio model is the first of one in this. The Rocchio Algorithm is a well-known method of implementing relevance feedback. It simulates a method for combining relevance feedback data into a vector space model. The Rocchio feedback technique, was created with help of Vector Space Model. The algorithm assumes that most users have a broad idea of whether document should



```
[16] print(precision)
     print(recall)

     [0.88541667 0.34939759 0.80769231 0.97274276 0.49217391 0.80236486
      0.01202749 0.25506757 0.98154362 0.67340067 0.81772575 0.99494949
      0.18950931 0.11784512 0.42158516 0.07345576 0.58715596 0.30319149
      0.10991379 0.03733333]
     [0.44783983 0.9902439  0.87169811 0.48554422 0.98263889 0.85585586
      1.         0.98051948 0.50518135 0.99009901 0.99188641 0.13998105
      0.92561983 1.         0.96525097 0.51764706 0.9221902  0.99418605
      0.94444444 0.73684211]

[17] Precision=np.average(precision)
     Recall=np.average(recall)
     print(Precision,Recall)

     0.49422463147145307 0.8123834382577388

     F_Measure = (2 * Precision * Recall) / (Precision + Recall)
     print(F_Measure)

     0.6145682315733102
```

Fig. 14. Rocchio Classifier

be classified as relevant or non-related. As a result, the user's search query is modified to include an arbitrary percentage of relevant and non-related classes in order to improve the search engine's recall and, potentially, precision.

2) Code Explanation: We're trying to discover the vector space of label in terms of word id in this code. Pre-processing the data and retrieving information from the data is done first. After that, we create a numpy array with the number of labels divided by the number of different words. Now we'll fill in the word id versus label matrix that was produced earlier. Matrix created will be shown as below.

vector space	word$_id1$	word$_id2$	word$_id3$
label$_1$	6	87838	877
label$_2$	7	78	5145
label$_3$	655	788	8507
label$_4$	545	18474	8560
label$_5$	88	878	6355

Now we normalize the the label vector.

Testing Now we have all the label vectors,. We will try to discover the absolute difference between all of them using the testing element. If the element is not present in the list then we have to find the label vector with the smallest difference. If an element does not appear in the list, it is ignored. The confusion matrix is now printed. Following that, we determine the document's precision and recall. After obtaining both precision and recall, we attempt to calculate the F score, which can be used to determine the Rocchio classifier model's accuracy.

3) Time Complexity of Rocchio Algorithm: The following table shows the time complexity for training and testing. Calculating the euclidean distance between a class centroid and the corresponding document can add complexity.

4) Limitation of Rocchio Algorithm: [15] When it comes to multi-modal classes and relationships, the Rocchio method frequently fails. It can miss-classify the similar objects. For example, despite having similar origins, the two inquiries "India" and "Pakistan" will seem significantly farther away in the vector space model.

B. KNN Algorithm

1) Introduction: The decision boundary is defined locally by the K nearest neighbour or KNN classification. K suggests the number of closest neighbours in this case. As a result, in order to work with KNN algorithm, we must calculate the distance between each new data point and training example. As a distance metric, we'll use the Euclidean distance algorithm. Our KNN Model selects K database items that are closest to the new data point. Now it chooses the high priority one, which implies that the most common label among those K entries will be the new data point's class. This process will repeat for all test datasets.

2) Code Explanation: In the first step, we have imported all of the necessary libraries and mounted the Google Drive with Colab. After that, we taken the news group dataset and combined all of the files into a single variable. We have taken the training and testing datasets and labels in the some variable as we move along. The data was then fitted into the vector variable after we initialised the tfidfvector in a single variable.

As we progress, we'll use the KNN classifier from the SkLearn library to initialise the variable. To check for any mistakes, we have used the cross validation method. Now we've arrived at the most important part of the code. We divided the data set into train and test parts in the test-classifier function. The train data set aids in the training of our model using the KNN classifier, whereas the test data set aids in the evolution of our model. We fit the training data set into the model and predicted the test data set in this final function.

As a result, now we are able to determine the model's accuracy, precision, recall, and f1 score for each document in the end. Also, we got the macro and weighted average of precision, recall and f1 score in the end.

3) Pros of KNN Algorithm: [10]

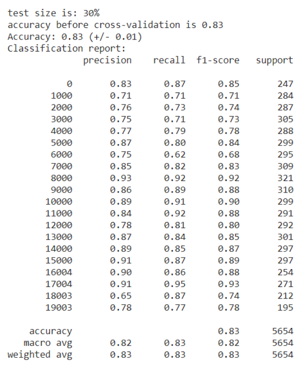
- We can train the model significantly faster with the K-nearest neighbour classifier than with conventional classification algorithms.
- The KNN approach can also be used to train a model with nonlinear data.
- The KNN algorithm is a simple, instance-based learning method.
- It can be applied to the regression issue.
- The average of the values of the k nearest neighbours is used to compute the object's output value.

4) Cons of KNN Algorithm:

- Because Euclidean distance is sensitive to magnitudes, the model's accuracy may be low
- When it comes to data storage, it demands a lot of memory compared to other classifier models.
- The KNN approach allows us to train the model more quickly, but the testing step is highly slow and requires more memory than other classifiers.
- The KNN approach cannot be used to train models with big data sets.

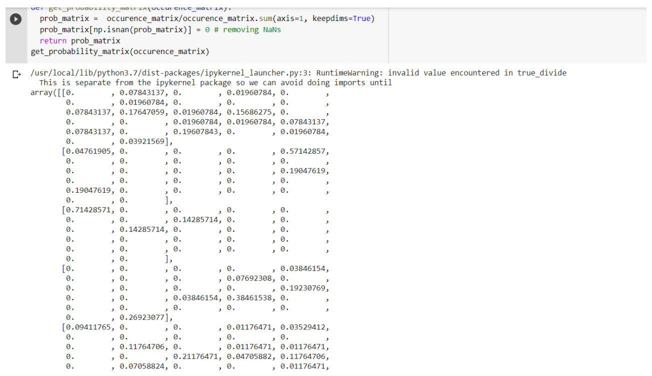
C. Naive Algorithm

Text classification is the process of classifying the texts. We usually classify them to help us grasp the various texts and their relevant classes. A Naive Bayes classifier is a



```
test size is: 30%
accuracy before cross-validation is 0.83
Accuracy: 0.83 (+/- 0.01)
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       247
        1000       0.71      0.71      0.71       284
        2000       0.76      0.73      0.74       287
        3000       0.75      0.71      0.73       305
        4000       0.77      0.79      0.78       288
        5000       0.87      0.80      0.84       299
        6000       0.75      0.62      0.68       295
        7000       0.85      0.82      0.83       309
        8000       0.93      0.92      0.92       321
        9000       0.86      0.89      0.88       310
       10000       0.89      0.91      0.90       299
       11000       0.84      0.92      0.88       291
       12000       0.78      0.81      0.80       292
       13000       0.87      0.84      0.85       301
       14000       0.89      0.85      0.87       297
       15000       0.91      0.87      0.89       297
       16004       0.90      0.86      0.88       254
       17004       0.91      0.95      0.93       271
       18003       0.65      0.87      0.74       212
       19003       0.78      0.77      0.78       195

    accuracy                           0.83      5654
   macro avg       0.82      0.83      0.82      5654
weighted avg       0.83      0.83      0.83      5654
```

Fig. 15. KNN Classifier



```
def get_probability_matrix(occurence_matrix):
  prob_matrix =  occurence_matrix/occurence_matrix.sum(axis=1, keepdims=True)
  prob_matrix[np.isnan(prob_matrix)] = 0 # removing NaNs
  return prob_matrix
get_probability_matrix(occurence_matrix)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value encountered in true_divide
  This is separate from the ipykernel package so we can avoid doing imports until
array([[0.        , 0.07843137, 0.        , 0.01960784, 0.        ,
        0.        , 0.01960784, 0.        , 0.        , 0.        ,
        0.07843137, 0.17647059, 0.01960784, 0.15686275, 0.        ,
        0.        , 0.        , 0.01960784, 0.01960784, 0.07843137,
        0.07843137, 0.        , 0.19607843, 0.        , 0.01960784,
        0.        , 0.03921569],
       [0.04761905, 0.        , 0.        , 0.        , 0.57142857,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.19047619,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.19047619, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        ],
       [0.71428571, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.14285714, 0.        , 0.        ,
        0.        , 0.14285714, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.03846154,
        0.        , 0.        , 0.        , 0.07692308, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.19230769,
        0.        , 0.        , 0.03846154, 0.38461538, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.26923077],
       [0.09411765, 0.        , 0.        , 0.01176471, 0.03529412,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.11764706, 0.        , 0.01176471, 0.01176471,
        0.        , 0.        , 0.21176471, 0.04705882, 0.11764706,
        0.        , 0.07058824, 0.        , 0.        , 0.01176471,
```

Fig. 16. Naive Bayes Classifier

data classification technique based on Bayes' theorem. The Naive Bayes model is based on conditional probability of two occurrences occurring based on the odds of each individual event occurring.

1) How It Works: Texts are lengthy and contain a big amount of words. The number of times a word appears in a single text data set (i.e. term frequency-tf) and the number of times that word appears across all text data sets (i.e. inverse document frequency-idf) are used to construct feature vectors. The probabilities of appearing of the words in the text among the texts of a specific category are stored in feature vectors representing text, allowing the algorithm to calculate the likelihood of that text belonging to that category. Once we've determined the likelihoods of several categories, we'll look for the category with the highest probability, and we'll have done our text classification.

2) Code Explanation: To begin, import all the necessary libraries into the code and attach Google Drive to it. Following that, separate lists for document class, confidence score, and training labels are created. The relevant data is put in the label variable after importing the label file from the drive. Now data is imported, which consists primarily of words and their counts. We have completed the mapping procedure, and now we can focus on frequencies. As per the document, doc N will have a label at (N-1) location during the labelling procedure. Then, using the Theorem, we are able to determine the probability of each class.

3) Pros of Naive Algorithm: [13]

- Calculation time is small in comparison to other models.
- Capable to handle the problem of multiple-class prediction.
- Works very well with numerical input variables than category inputs.

4) Cons of Naive Bayes Algorithm:

- The naive Bayes assume that all features are independent which is not possible in the real life scenario.
- In some circumstances, its estimation may be incorrect.
- Zero frequency can be a problem for this model. Due to which smoothing technique is used for correctness.

D. Conclusion

- f1 score of KNN classifier : 0.83
- f1 score of Rocchio classifier : 0.61
- f1 score of Naive Bayes classifier : 0.78

Knn is the strongest text classifier among these three models, according to the f1 score.

link to code folder

III. LAB-2 | Block Sort Based Indexer and Building Term-Document Matrix (TF-IDF)

A. *Block Sort Based Indexer*

1) Introduction: We begin by traversing the collection and assembling all term–docID pairs. The pairs are then sorted with the term as the primary key and the docID as the secondary key. Finally, we organise each term's docIDs into a postings list and compute statistics such as term and document frequency. All of this may be done in memory for small collections.[9]

2) Process: We represent words as termIDs, which are unique serial numbers, to make index construction more efficient. We can either create the inverted index in the first run and then build the mapping from terms to termIDs on the fly while processing the collection, or we can do both in a two-pass method.

We must will use an external sorting method, i.e. one that requires disc, because main RAM is inadequate. The essential condition of such an algorithm for reasonable performance is that it reduce the amount of random disc seeks during sorting — sequential disc reads are far quicker than seeks.The blocked sort-based indexing algorithm, or BSBI in Figure 4.2 of code, is one solution. BSBI (i)divides the collection into equal-sized parts, (ii) sorts the termID–docID pairings in memory, (iii) saves intermediate sorted results to disc, and (iv) combines all intermediate results into the final index.

```
BSBINDEXCONSTRUCTION()
1 n ← 0
2 while (all documents have not been processed)
3 do n ← n +1
4 block ← PARSENEXTBLOCK()
5 BSBI-INVERT(block)
6 WRITEBLOCKTODISK(block, fn)
7 MERGEBLOCKS( f1, . . . , fn; fmerged)
```

Figure 4.2 of code Blocked sort-based indexing. The algorithm stores inverted blocks in files f1, . . . , fn and the merged index in fmerged.

3) How does it work: Documents are divided into termID–docID pairs and stored in memory until a fixed-size block (PARSENEXTBLOCK in Figure 4.2 of code) is full. We set a block size that fits perfectly within memory to enable a rapid in-memory sort. The block is then written on disc in reverse order. Inversion consists of two phases. To begin, we sort the termID–docID pairs. Then, in a posts list, POSTING, we gather all termID–docID pairings that have the same termID. A posting is nothing more than a docID. The outcome is saved to disc as an inverted index for the block we just read. In the last phase, the approach merges the ten blocks into a single massive combined index.

B. *Term-Document Matrix(TF-IDF)*

1) Introduction: Tf-idf is available in a variety of flavours, some more complicated than others. The basic mathematical operations to learn are addition, multiplication, and division. At some point, we'll need to calculate the natural logarithm of a variable, but most online calculators and calculator mobile apps can do it for us. The raw term counts for the first thirty words of Bly's obituary are listed below in alphabetical order, but this version contains a second column indicating the number of periodicals in which each phrase may be found.

2) Process: The document frequency (df) is a tally of how many times each word appears in the corpus. (Document frequency for a specific word is denoted by dfi.)

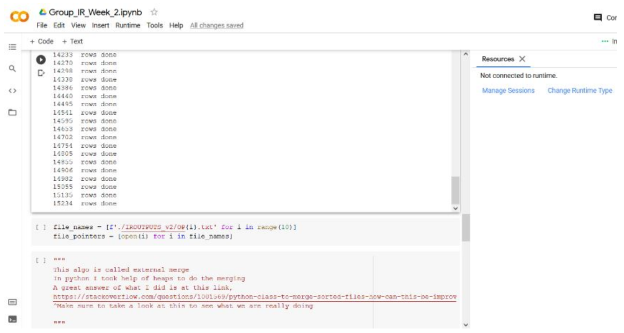
N/dfi is the simplest formula for determining inverse document frequency for each term, where N is the total number of documents in the corpus. Many implementations, however, require extra processes to normalise the results. In TF-IDF, normalisation is used in two ways: first, to prevent term frequency bias from words in shorter or longer documents, and second, to calculate the idf value for each word (inverse document frequency). The method in Scikit-Learn, for example, represents N as N+1, calculates the natural logarithm of (N+1)/dfi, and then adds 1 to the final result. In the section below labelled "Scikit-Learn Settings," we'll return to the concept of normalisation.

The following equation can be used to express Scikit-idf Learn's transformation:

idfi = In[(N+1)/dfi] +1

After calculating idf;, tf-idf; is tf multiplied by idf.

tf-idfi = tfi x idfi


Group_IR_Week_2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Resources
Not connected to runtime.
Manage Sessions
Change Runtime Type

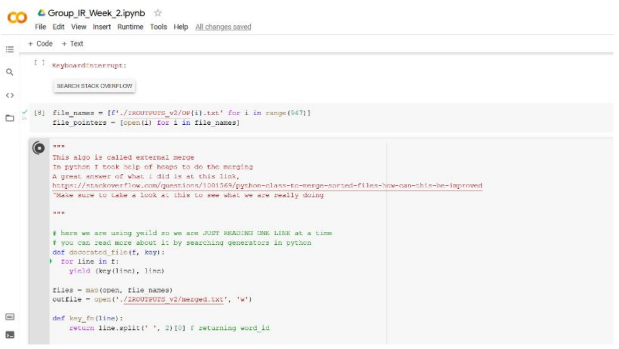
```
14233  rows done
14270  rows done
14298  rows done
14338  rows done
14386  rows done
14440  rows done
14495  rows done
14541  rows done
14595  rows done
14653  rows done
14702  rows done
14754  rows done
14805  rows done
14855  rows done
14906  rows done
14982  rows done
15055  rows done
15135  rows done
15234  rows done
```

```
file_names = [f'./IROUTPUTS_v2/OP{i}.txt' for i in range(10)]
file_pointers = [open(i) for i in file_names]
```

```
"""
This algo is called external merge
In python I took help of heaps to do the merging
A great answer of what I did is at this link,
https://stackoverflow.com/questions/1001569/python-class-to-merge-sorted-files-how-can-this-be-improv
^Make sure to take a look at this to see what we are really doing

"""
```

Fig. 17. Processing database and creating block sorted based index files

If you've worked with them before, they can provide a more lucid overview of an algorithm's processes than any well-written prose. I've added two new columns to the previous terms frequency table to make the idf and tf-idf equations more obvious. To calculate the final tf-idf score, the first new column represents the derived idf score, while the second new column multiplies the Count and Idf columns.



```
KeyboardInterrupt:
```

```
[8] file_names = [f'./IROUTPUTS_v2/OP{i}.txt' for i in range(947)]
    file_pointers = [open(i) for i in file_names]
```

```
"""
This algo is called external merge
In python I took help of heaps to do the merging
A great answer of what I did is at this link,
https://stackoverflow.com/questions/1001569/python-class-to-merge-sorted-files-how-can-this-be-improved
^Make sure to take a look at this to see what we are really doing

"""

# here we are using yeild so we are JUST READING ONE LINE at a time
# you can read more about it by searching generators in python
def decorated_file(f, key):
  for line in f:
    yield (key(line), line)

files = map(open, file_names)
outfile = open('./IROUTPUTS_v2/merged.txt', 'w')

def key_fn(line):
    return line.split(' ', 2)[0] # returning word_id
```

Fig. 18. Merging the sorted files in one

C. *Conclusion*

It's worth noting that the idf score is higher if the phrase appears in fewer papers, although the visible idf score ranges from 1 to 6. Different scales would result from different normalisation approaches.
Link to code folder

IV. LAB-6 | Latent Semantic Indexing

A. *Introduction*

Latent semantic indexing (also known as Latent Semantic Analysis) is a way of evaluating a series of documents to find statistical co-occurrences of words that appear together, which subsequently provides information about the themes of those words and documents.[8]

Two well-studied machine learning algorithms to text classification are Support Vector Machines (SVM) and k-nearest neighbours (kNN). They're both used on a vector space model that's defined over a collection of words (BOW). Documents are represented as vectors across a set of dimensions in this approach, with each dimension representing to a word in the BOW.

B. *Processing*

LSI's goal is to extract a smaller number of dimensions that are more reliable markers of meaning than single phrases. To get at these latent dimensions, LSI employs the Singular Value Decomposition (SVD). In theory, SVD does a two-mode factor analysis, placing both terms and documents in a single space specified by the extracted dimensions. The original term document matrix is broken down into three matrices by SVD, two of which indicate the altered representations of words and documents in terms of the additional dimensions, while the third assigns "weights" to these dimensions. The theory underpinning LSI is that due to word-choice unpredictability, less important dimensions correspond to "noise." By removing these noisy dimensions, a lower rank approximation to the original matrix is created. This approximation is a smoothed (blurred) version of the original, and it is supposed to reflect relationships between terms and documents more precisely in terms of the underlying concepts. The generated approximation is the closest matrix of its rank to the original in the least-squares sense, which is an intriguing property of SVD.[16] While LSI has proven to be effective in retrieval tasks, it has several drawbacks when it comes to categorization. As previously stated, because LSI is blind to training document class labels, the extracted dimensions are not always the best in terms of class discrimination. In addition, LSI may filter out rare terms with strong discriminatory power.

- SVD minimises the "distance" between the two matrices as measured by the 2-norm by representing A in a reduced dimensional space:
$$\triangle = \|A - \hat{A}\|_2$$
- The SVD projection is computed by decomposing the document-by-term matrix At×d into the product of three matrices, Tt×n , Sn×n , Dd×n
$$A_{t*d} = T_{t*n} S_{n*n} (D_{d*n})^T$$

- where t is the number of terms, d is the number of documents, n = min(t,d), T and D have orthonormal columns, i.e. $TT^T = D^TD = I, rank(A) = r, S = diag(\sigma_1, \sigma_2, ..., \sigma_n,), \sigma_i > 0 for 1 \leq i \leq r, \sigma_j = 0 for j \geq r+1$.

C. Queries

- For purposes of information retrieval, a user's query must be represented as a vector in k-dimensional space and compared to documents. A query (like a document) is a set of words. For example, the user query can be represented by

$$\hat{q} = q^T T_{t*k} S_{k*k}^{-1}$$

- where q is simply the vector of words in the users query, multiplied by the appropriate term weights. The sum of these k-dimensional terms vectors is reflected by the term $q^T T_{t*k}$ in the above equation, and the right multiplication by S_{k*k}^{-1} differentially weights the separate dimensions.

D. Updating

- Updating refers to the general process of adding new terms and/or documents to an existing LSI-generated database. Updating can mean either folding-in or SVD-updating.
- Folding-in terms or documents is a much simpler alternative that uses an existing SVD to represent new information.
- Recomputing the SVD is not an updating method, but a way of creating an LSI-generated database with new terms and/or documents from scratch which can be compared to either updating method.

The equation for folding documents into the space can again be derived from the basic SVD equation:

$$A = TSD^T$$
$$T^TA = T^TTSD^T$$
$$T^TA = SD^T$$

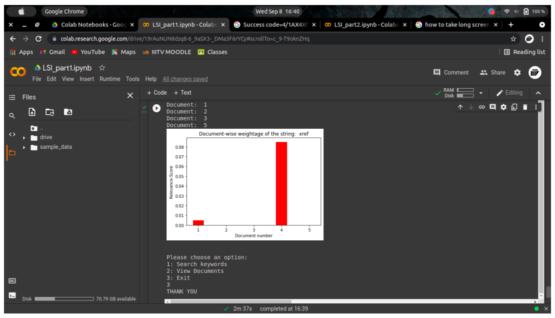
E. Lab Practical

link to code

F. Conclusion

From the above experiment we can conclude some advantages of this model.[4]

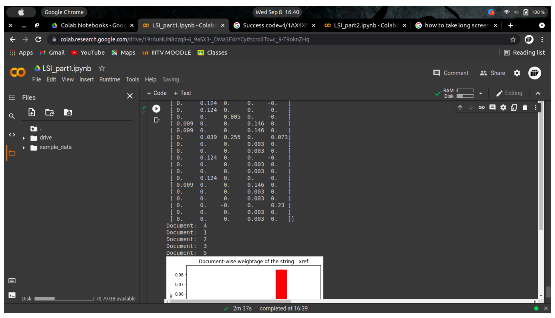
1. The Assumption in LSI is that the new dimensions are a better representation of documents and queries.



```
Document:  1
Document:  2
Document:  3
Document:  5
```

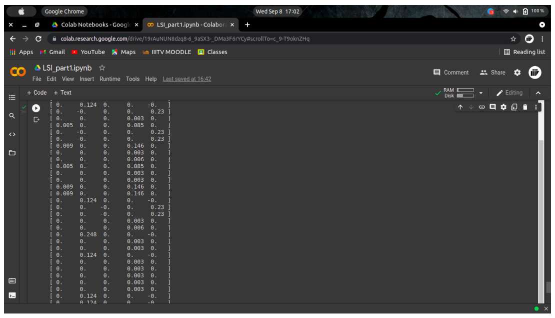
```
Please choose an option:
1: Search keywords
2: View Documents
3: Exit
3
THANK YOU
```

Fig. 19. 1



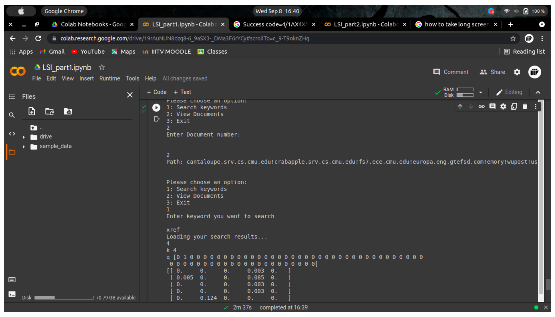
```
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.     0.805  0.    -0.   ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.     0.039  0.255  0.     0.073]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.    -0.     0.     0.23 ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]]
Document:  4
Document:  1
Document:  2
Document:  3
Document:  5
```

Fig. 20. 2



```
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.    -0.     0.     0.     0.23 ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.005  0.     0.     0.085  0.   ]
 [ 0.    -0.     0.     0.     0.23 ]
 [ 0.    -0.     0.     0.     0.23 ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.006  0.   ]
 [ 0.005  0.     0.     0.085  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.009  0.     0.     0.146  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.    -0.     0.     0.23 ]
 [ 0.     0.    -0.     0.     0.23 ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.006  0.   ]
 [ 0.     0.248  0.     0.    -0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
```

Fig. 21. 3



```
Please choose an option:
1: Search keywords
2: View Documents
3: Exit
2
Enter Document number:


2
Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!europa.eng.gtefsd.com!emory!wupost!us


Please choose an option:
1: Search keywords
2: View Documents
3: Exit
1
Enter keyword you want to search

xref
Loading your search results...
4
k 4
q [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[ 0.     0.     0.     0.003  0.   ]
 [ 0.005  0.     0.     0.085  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.     0.     0.003  0.   ]
 [ 0.     0.124  0.     0.    -0.   ]
```

Fig. 22. 4

2. The same underlying concept can be described using different terms.

3. Polysemy describes the availability of the words that have more than one meaning, which is common property of language.

This model also have some disadvantages as following

1. It becomes very challenging and time-consuming to handle very much bigger matrix and it becomes exponentially more time consuming with the increment of unique words.

2. we can say that the SVD representation is more dense. Many documents have more than 200 unique terms. So the sparse vector representation will take up more storage space than the compact SVD representation if we reduce to 200 dimensions.

V. LAB - 8 | Support Vector Machine

A. Introduction

Support vector machines use hyperplanes or sets of hyperplanes in a high-dimensional or infinite-dimensional space for classifying, regressing, or detecting outlines. According to this premise, the hyperplane that has the shortest distance to the nearest training-data point of any class is considered to be a good indicator of separation, since the larger the margin, the lower the generalization error of the classifier.[14]

B. Kernels

Due to their kernel, SVMs are also referred to as kernelized SVMs because they convert the input data space into a higher-dimensional space.[11]

$$\phi(x_i)$$

Contains the kernel function that turns the input space into a higher-dimensional space, so that not every data point is explicitly mapped. Examples of kernel functions in scikit-learn include:

1) Linear Function
2) Polynomial Function
3) Radial Basis Function (RBF)
4) Sigmoid Function

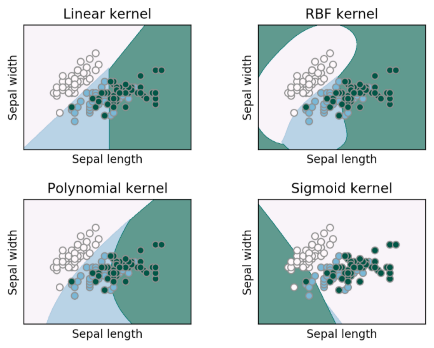


Fig. 23. Visualization of the different kernel functions

C. Functional Margin and Geometric Margin

An area in which the line is separated from the closest class points is called a margin. A good margin is one where this separation is greater for both classes. The example below shows the difference between a good and a bad margin. Points can remain in their classes without crossing into another class with a good margin.

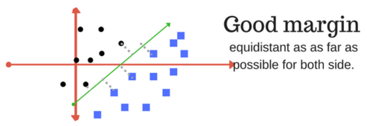


Fig. 24. Good Margin

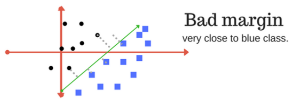


Fig. 25. Relatively Bad Margin

1) *Functional Margin:* As our training dataset has labeled classes for each point, we can compare the classifier's prediction $w^T x + b$ with the actual class y_i. The functional margin is defined as:

$$\gamma = y_i * (w^T x + b)$$

When $y_i = 1$ we would require an enormous positive number for $w^T x + b$ to have a huge practical edge, while if $y_i = 1$ we would require a huge negative number for $w^T x + b$ to have a huge utilitarian edge. Also, our classifier plays out a right order for all qualities where $_1$ is positive.[12]

One method for amplifying this utilitarian edge is to discretionarily scale w and b, since grouping with $h_{w,b}$ just relies upon the sign, not the extent, of the brought esteem back. In any case, this offers little benefit and is a shortcoming of useful edges, as we have no valuable method for expanding this articulation.

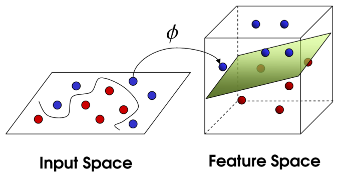


Fig. 26. Visualization of the SVM

2) *Geometric Margin:* The mathematical edge is characterized as the separation from an information highlight the choice limit. How should this be characterized? How about we take a gander at the information point named "A" in the figure 29. Our edge, $_1$, characterizes the scalar length from our informative element (x_1) to the choice limit while w|w| addresses the unit bearing ordinary to the choice limit (ie. the course of the most brief way from A to B). Utilizing vector deduction we can characterize point B as:

$$\vec{A} - \gamma_i(\frac{W}{|\vec{W}|})$$

Or in other words:

$$\vec{A} - margin = \vec{B}$$

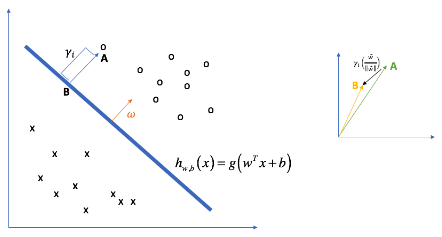


Fig. 27. Geometric Margin

D. Applications

- As a result of their application, SVMs can be useful in text and hypertext categorization as it reduces the need for labeled training instances, both in inductive and transductive settings.[7] Support vector machines can also be used for shallow semantic parsing.
- SVMs may also be used for picture classification. SVMs offer much greater search accuracy than typical query refinement techniques after only three to four rounds of relevance feedback, according to experimental data. This is also true for image segmentation systems, even those that use a modified version of SVM that employs Vapnik's privileged method.
- Classification of satellite data like SAR data using supervised SVM.
- Hand-written characters can be recognized using SVM[6]

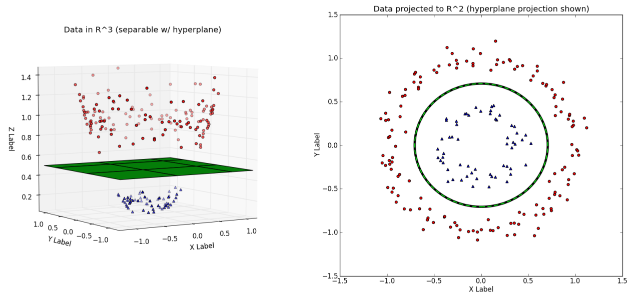


Fig. 28. Visulization of SVM

E. Conclusions

1) Pros:

- It works really effectively when there is a clear margin of separation.
- is useful when the number of dimensions exceeds the number of samples.
- It is also memory efficient since it employs a subset of training points in the decision function (called support vectors).[6]
- It is effective in high dimensional spaces.

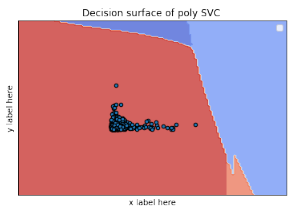


Fig. 29. Output

2) *Cons:*

- When we have a huge data set, it does not perform well since the needed training time is longer.
- It also does not perform well when the data set has more noise, i.e. target classes overlap.
- SVM does not immediately offer probability estimates; these are obtained through a costly five-fold cross-validation procedure. It is similar to the SVC technique of the Python scikit-learn module.[6]

Code of Support Vector Machine : Link

VI. LAB - 9 | GOOGLE PAGERANK ALGORITHM

A. *Introduction*

Google PageRank algorithm was developed by Lary Page, the CEO and founder of Google. It was developed to rank the different webpages according to their content and demand on the world wide web. After that, the algorithm is used for ranking users in social media and many more. In this article, we will provide the theory of pagerank algoritm and its working.

B. *Original Summation Algoritm For PageRank*

The inventor began with a simple summation equation, the roots of which actually derive from bibliometrics research, the analysis of the citation structure among academic papers. The PageRank of a page Pi, denoted r(Pi), is the sum of the PageRanks of all pages pointing into Pi where B(Pi) is the set of pages pointing into Pi (backlinking to Pi in Brin and Page's words) and |Pj | is the number of outlinks from page Pj .

$$r(P_i) = \sum_{P_j \in B_{P_j}} \frac{r(P_j)}{|P_j|}$$

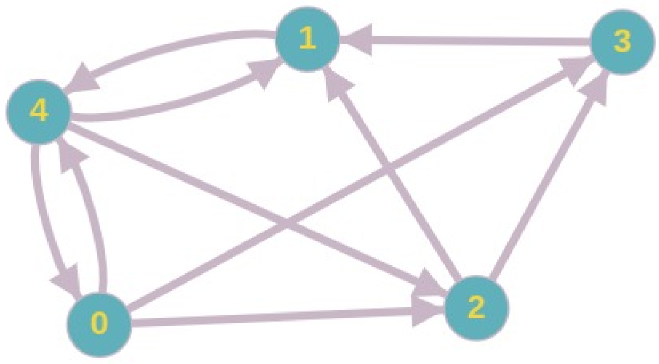


Fig. 30. Directed Graph

From the above formula and the directed graph given, we can write the formula as follows.

$$r_0 = \frac{r_4}{3}$$

$$r_1 = \frac{r_2}{2} + \frac{r_4}{3} + r_3$$

$$r_2 = \frac{r_0}{3} + \frac{r_4}{3}$$

$$r_3 = \frac{r_2}{2} + \frac{r_0}{3}$$

$$r_4 = \frac{r_0}{3} + r_1$$

Here we can see that, a system of linear equations is created and it can be solved using Gaussian Elimination method. But this formula can be accurate for small graphs. If we have more edges and nodes then it can not be a good formula.

C. Markov Chain and PageRank Matrix

1) Markov Chain Matrix: Since a Markov Chain is characterized by an underlying circulation and a change in matrix, the above graph can be viewed as a Markov chain with the following transition matrix:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 31. Markov Matrix

Here we can see that the P transpose is row stochastic from which is the basic condition for markov chain theorem. Lets consider the below given equation for initial condition.

$$\pi = (\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}, \frac{1}{n})[2]$$

where n represents total number of nodes. This implies that the irregular walker will pick randomly the underlying node from where it can arrive at any remaining nodes.

At each progression, the arbitrary walker will jump to one more node as per the transition matrix. The likelihood appropriation is then registered for each progression. This distribution lets us know where the arbitrary walker is probably going to be after a specific number of steps. The likelihood circulation is computed utilizing the following condition:

$$\pi^{(t+1)} = P\pi^{t}$$

A fixed circulation of a Markov chain is a likelihood conveyance with = P. This implies that the circulation won't change after one stage. Note that not all Markov chains concede a fixed dispersion.

In the event that a Markov chain is emphatically associated, which implies that any node can be reached from some other node, then, at that point, it concedes a fixed conveyance. It is the situation in our concern. In this way, after a boundlessly long walk, we realize that the likelihood dissemination will join to a fixed circulation .

2) Frobenius-Perron Thereom: We notice that is an eigenvector of the matrix P with the eigenvalue 1. Rather than figuring all eigenvectors of P and select the one which relates to the eigenvalue 1, we utilize the Frobenius-Perron theorem.

As indicated by Frobenius-Perron theorem, if a matrix A will be a square and positive network (every one of its entrances are positive), then, at that point, it has a positive eigenvalue r, for example, || < r, where is an eigenvalue of A. The eigenvector v of A with eigenvalue r is positive and is the one of a kind positive eigenvector.

D. Subdominant Eigenvalue of the Google Matrix

For the google Matrix

$$G = \alpha S + (1-\alpha)1/nee^{T}, |\lambda_2(G)| \leq \alpha.$$

• For the case when $|\lambda_2(S)| = 1$ (which occurs often due to the reducibility of the web graph), $|\lambda_2(G)| = \alpha$. Therefore, the asymptotic rate of convergence of the PageRank power method of equation (4.6.1) is the rate at which $\alpha K \rightarrow 0$.[7]

E. Teleportation Matrix

In the web graph, for instance, we can find a site page I which alludes just to page j and j alludes just to I. This is the thing that we call insect **spider trap problem**. We can likewise find a site page which has no outlink. It is normally named Dead end.

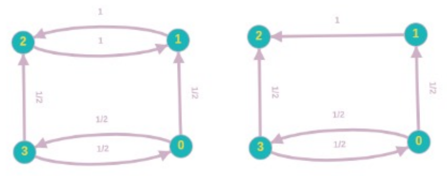


Fig. 32. Dead ends and spider traps illustration

In this case, you can see that node 1 and node 2 are link with each other. From node 1 and 2 we are unable to go node 3 or 0. This is called spider trap or dead end. To solve this problem, **teleportation** concept is introduced.

Teleportation comprises of associating every node of the graph to any remaining nodes. The diagram will be then finished. The thought is with a specific likelihood , the irregular walker will leap to one more node as per the progress grid P and with a likelihood (1-)/n, it will bounce randomly to any node in the diagram. We get then the new change grid R:

$$R = \beta P + (1-\beta) v e^T$$

$$e^T = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$$

$$v = (1, ..., 1)^T.$$

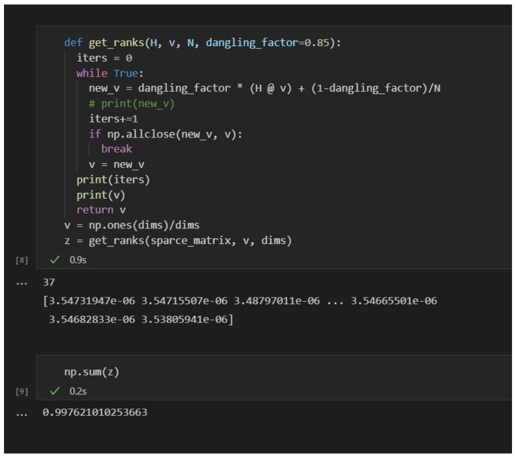
where v is a vector of ones, and e a vector of 1/n. is commonly defined as the **damping factor**. In practice, it is advised to set to 0.85.

By applying teleportation in our example, we get the following new transition matrix:

$$R = \begin{pmatrix} \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{3}\beta + \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1}{2}\beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \\ \frac{1}{3}\beta + \frac{1-\beta}{5} & \beta + \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} & \frac{1-\beta}{5} \end{pmatrix}$$

Fig. 33. Our new transition matrix

F. Output Of The Code



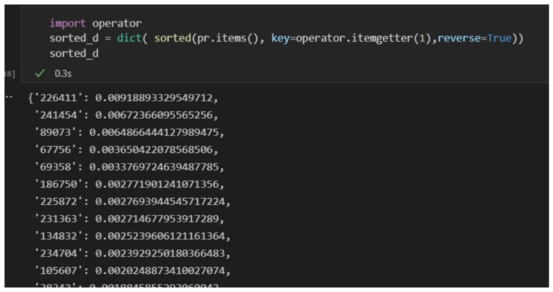
```
def get_ranks(H, v, N, dangling_factor=0.85):
  iters = 0
  while True:
    new_v = dangling_factor * (H @ v) + (1-dangling_factor)/N
    # print(new_v)
    iters+=1
    if np.allclose(new_v, v):
      break
    v = new_v
  print(iters)
  print(v)
  return v
v = np.ones(dims)/dims
z = get_ranks(sparce_matrix, v, dims)
```

```
[8]  ✓ 0.9s
...  37
     [3.54731947e-06 3.54715507e-06 3.48797011e-06 ... 3.54665501e-06
      3.54682833e-06 3.53805941e-06]
```

```
np.sum(z)
```

```
[9]  ✓ 0.2s
...  0.997621010253663
```

Fig. 34. ranking and verification by adding it



```
import operator
sorted_d = dict( sorted(pr.items(), key=operator.itemgetter(1),reverse=True))
sorted_d
```

```
8]  ✓ 0.3s
..  {'226411': 0.009188933295497l2,
     '241454': 0.00672366095565256,
     '89073': 0.006486644127989475,
     '67756': 0.003650422078568506,
     '69358': 0.003376972463948778 5,
     '186750': 0.002771901241071356,
     '225872': 0.0027693944545717224,
     '231363': 0.002714677953917289,
     '134832': 0.0025239606121161364,
     '234704': 0.0023929250180366483,
     '105607': 0.0020248873410027074,
```

Fig. 35. List of documents id in sorted manner of rank

G. Conclusion

The matrix R has similar properties than P which implies that it concedes a fixed appropriation, so we can utilize every one of the hypotheses we saw beforehand. All the algorithm are updated as research goes on but the basic formation of page ranking is same.

Code of Page Rank : Link1 and Link2

VII. Lab - 10 | Concept Lattice

A. Introduction

People typically depict and perceive objective things from various levels and diverse granularity. There is a course of developing the trait qualities of genuine things, and thus the information ideas at various levels or at various granularity are gotten. Granular figuring is a sort of helpful numerical strategy for handling complex design information, and the possibility of granular registering fits impeccably with the progressive and granular considering mode "from coarse to fine, from entire to part" during the time spent human insight.

Idea cross section is the critical information construction of formal setting investigation, and it is additionally a sort of essential information structure reasonable for information portrayal and information disclosure. Idea grid models and their expansion models are set up on the connection among items and properties, they handle just the basic kind of information, and there is no model that has had the option to deal with information with complex designs.[17]

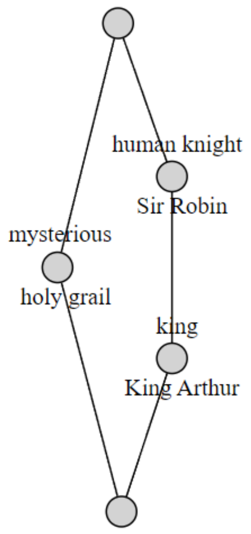


Fig. 36. Introduction

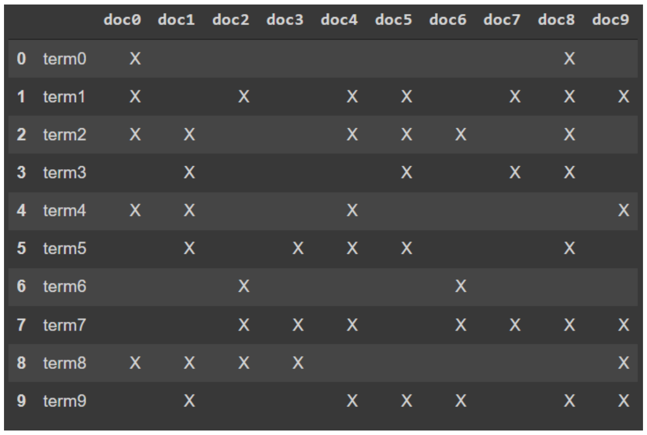
B. Time complexity

Concept Lattice can be implemented with the help of these two algorithm.

1) The amount of layered qualities and lower attributes in the layered context determine the time complexity of the Roll-up building process. In the following, we'll suppose that K has l layered attributes, C(K) has s nodes, and that the maximum number of lower attributes of all layered attributes is r. As a result, the time Complexity is O((r+r*log2r + s+ (s-r)) * l) = O(l(2s+r*log2 r)).
2) The Drill-down method's temporal complexity is explored more below. In the following, we'll assume L for the number of layered attributes in K, s for the number of node concepts in C(K), and r for the maximum number of lower attributes. As a result, this algorithm's time complexity is O((|G|r2 +s*r+ r2)l) = O(l((|G| +1) r2 +s*r)).[17]

C. Code Explanation

We have Built a Lattice of size 10X10 by code.



		doc0	doc1	doc2	doc3	doc4	doc5	doc6	doc7	doc8	doc9
0	term0	X								X	
1	term1	X		X		X	X		X	X	X
2	term2	X	X			X	X	X		X	
3	term3		X				X		X	X	
4	term4	X	X			X					X
5	term5		X		X	X	X			X	
6	term6			X				X			
7	term7			X	X	X		X	X	X	X
8	term8	X	X	X	X						X
9	term9		X			X	X	X		X	X

Fig. 37. Code Output-1

1) atrsize = 20
2) arr = np.random.randint(2, size=(atrsize, atrsize))
3) newcol = [f'termi' for i in range(atrsize)]
4) df[df == 1] = 'X'
5) df[df == 0] = ''
6) df.insert(loc=0, column='', value=newcol)
7) df.tocsv('./dummy.csv', index=False)

In the first line, we define a square matrix of size 20. In the next line, we define the variable arr, which can take one of two values: 0 or 1. In the next line, we insert data(0 or 1) in each of the 20 columns of size 20 and name the column doc(i) where i=0 to 19. In the next line, we created a column called term(i), where I ranges from 0 to 19. In the next line, we substituted 1 with 'X' in the column data. In the next line, we substituted 0 with '' in the column data. In the next line, we insert a serial number column. In the final line, we save this matrix to a csv file.

1) context = loadcsv('./dummy.csv')
2) context
3) df
4) context.lattice.graphviz()

We begin by loading a csv file and then print the file's context. Then we output the file's data frame. Then we create a matrix lattice graph.

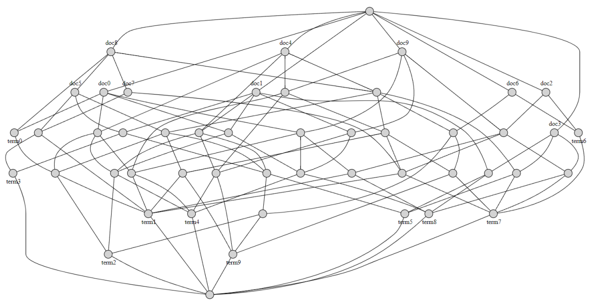


Fig. 38. Code Output-2

D. Conclusion

We can see here that as the number of properties and objects in the matrix increases, the idea lattice becomes illegible for humans. A layered idea lattice model is produced when some qualities in a formal context are made up of some sub attributes. We've seen a roll-up building algorithm that constructs the upper concept lattice from the lower concept lattice, as well as a drill-down building algorithm that constructs the lower concept lattice from the upper concept lattice. Experiments and examples indicate that the layered idea lattice model may be used to describe complicated structural data, and that the roll-up and drill-down construction algorithms are successful.

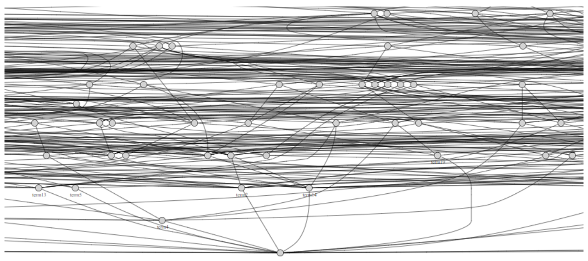


Fig. 39. Conclusion

Code of Concept Lattice : Link

References

[1] Ralf Abueg. *Elasticsearch: What It Is, How It Works, And What It's Used For*. URL: https://www.knowi.com/blog/what-is-elastic-search/.

[2] Amrani Amine. *PageRank algorithm*. URL: https://towardsdatascience.com/pagerank-algorithm-fully-explained-dc794184b4af.

[3] AngeloCatalani. *Web Information Retrieval | Vector Space Model*. URL: https://www.geeksforgeeks.org/web-information-retrieval-vector-space-model/.

[4] Raghavendran Balu. *ADV*. URL: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-Latent-Semantic-Analysis.

[5] Giovanni Pagano Dritto. *An Overview on Elasticsearch and its usage*. URL: https://towardsdatascience.com/an-overview-on-elasticsearch-and-its-usage-e26df1d1d24a.

[6] Eric Kim. *Support vector machines*. URL: https://www.jeremyjordan.me/support-vector-machines/#:~:text=Summary.

[7] AMYN. LANGVILLE and CARL D. MEYER. *Google's PageRank and Beyond*. URL: https://gi.cebitec.uni-bielefeld.de/_media/teaching/2019winter/alggr/langville_meyer_2006.pdf.

[8] Susan li. *LSI defination*. URL: https://towardsdatascience.com/latent-semantic-analysis-sentiment-classification-with-python-5f657346f6a3.

[9] Christopher D. Manning. *block sort based indexing*. URL: https://nlp.stanford.edu/IR-book/html/htmledition/blocked-sort-based-indexing-1.html.

[10] Avinash Navlani. *KNN*. URL: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn.

[11] Nicolus Rotich. *Multiclass Classification with Support Vector Machines (SVM), Dual Problem and Kernel Functions*. URL: https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02.

[12] Jinde Shubham. *SUPPORT VECTOR MACHINES (SVM)*. URL: https://medium.com/coinmonks/support-vector-machines-svm-b2b433419d73.

[13] Pavan Vadapalli. *Naive Bayes*. URL: https://www.upgrad.com/blog/naive-bayes-explained/.

[14] Wikipedia. *Support-vector machine*. URL: https://en.wikipedia.org/wiki/Support-vector_machine.

[15] wikipedia. *Rocchio Algorithm*. URL: https://en.wikipedia.org/wiki/Rocchio_algorithm.

[16] wikipedia. *SVD*. URL: https://en.wikipedia.org/wiki/Singular_value_decomposition.

[17] Xia Wu, Jialu Zhang, and Jiaming Zhong. *Layered concept lattice model and its application to build rapidly concept lattice*. June 2020. URL: https://www.hindawi.com/journals/cin/2020/5784209/.