International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-07, Issue-03, JUNE 2021

Performance Efficient IR using Language Filters

Kathakali Mitra, Department of Computer Science and Engineering, Birla Institute of Technology and Science , Pilani , India. kathakali1595@gmail.com

ABSTRACT - Throughout the technological domain , Information Retrieval systems play a critical role in gathering unstructured text to satisfy the user requirements. These IR systems allow users access the recent and most relevant information in an efficient way and a lesser time .This paper aims to extract the top relevant documents of a specified language for a given free text query in an efficient and optimized way. In the proposed work , NLP based framework is used to identify the language , index it and extract the relevant documents. Language Identification is done by running a pipeline of TF-IDF Vectorizer on Bag of Words and by building an appropriate classifier model. Among the 5 popularly used classifiers namely being SVM , Naïve Bayes , Decision Trees and KNN , the dataset worked best with Random Forest Classifier for Language Classification with an accuracy of 91%. Indexing is done using a relevant data structure to invoke only the corresponding dataset pertaining to the same language. A modified vector space modelling accompanied by an updated TF-IDF weighting technique to consider the freshness and number of votes is proposed for extracting top relevant trending documents.

Keywords: TF-IDF Vectorizer , Random Forest Classifier , Cosine Similarity , Lemmatization ,Jacard Similarity , Vector Space Model

I. INTRODUCTION

With significant rise in data worldwide , users want to obtain the data which likely resembles the free text query in a faster time. The simplest form of document retrieval is for a machine to do a linear scan through all the documents i.e grepping the text. With evolution of technology and production of huge data worldwide , linear scan was not the preferrable technique of document retrieval. IR systems were hence introduced to process large documents very quickly , allow more flexible matching operations and facilitate ranked retrieval. Information Retrieval domain closely works in extracting the most relevant and useful data to the user which matches their requirements. Different algorithms , models and tools have evolved with time which tried to extract the best possible match to the user query. The World Wide Web contains a vast amount of information and is a rich source for data extraction, but manually extracting data from the Web is a tedious and time-consuming process, especially when a large amount of data matches the extraction criteria.

Indexing supported Boolean Queries given by Boolean Retrieval Model. With huge production of data from various sources , the resulting documents can far exceed the linear scanning through indexing. It is essential for a search engine to rank documents matching a free-text-query. This is the reason for adoption of different weighting and scoring techniques to rank a document.

The key task of different modelling technique is a faster and an accurate retrieval policy. There were different challenges in terms of time complexity and faster access of relevant trending and most frequently accessed documents. The proposed work intends to solve this challenge by working with a lesser time complexity, optimized storage capacity and providing accurate results. The algorithm provides the most relevant domain specific , language specific , current trending and most frequently accessed documents which matches the user specified query.

II. RELATED WORK

The widely used models of IR from the time of evolution included Boolean Retrieval Model, TF-IDF Vectorizer, Vector Space Model, n-Gram Modelling etc. Boolean Retrieval Model used Term Incidence Matrix for indexing operations but was not widely accepted because of certain limitations. Inverted Index was used for storing the posting list for this method. The top limitations being its sparse nature and inability to handle similarity queries. Wildcard characters also played a critical part in IR systems. Tolerant Retrieval, Asymmetric Query Expansion, Normalization, Equivalence Class are used for converting words that have different representations in different documents into the same format. Stemming and Lemmatization was used for converting words expressed in different parts of speech into the same inflectional form. [1] Phrases were often used as queries while searching relevant documents on the search engine. These phrase queries use bi-word indexes for Information Retrieval asks. Bi-word indexes generate a huge number of false positives and are not preferred as the ideal model for IR.

Integration of wild cards in our queries can generally reduce the number of queries that must be issued, hence simplifying the extraction task. A problem in querying natural language text is that a free text query specified by a user may not

 DOI : 10.35291/2454-9150.2021.0334

retrieve enough exact matches. Unlike term queries which can be relaxed by removing some of the terms removing terms from a wild card query without ruining its meaning is more challenging. There are various ways to deal with wildcard queries the most popular being the Permuterm Index. Example : mon*day can be rotated unless the * appears at the end of the query.[3]

With an increase in the amount of data from different data sources, ranking of documents is a critical task for providing the best possible match to the user-specified query. Different weighting mechanisms are available the most popular being TF-IDF, Raw Term Frequency Weighting Scheme. The TF-IDF converts the text document into a sparse matrix of a Bag of Words of corpus and then performs different operations and modeling on the obtained matrix. The TF-IDF Vectorization concept is used in text summarization, domain classification etc. [1]

Different Similarity Measures were used to compare the best-suited document to the query in the vector space modeling. Cosine Similarity, Inner Product, Dice Coefficient, Jaccard Similarity was used to calculate the distance between the document and the free-text query. Top n documents matching the query with the shortest distance would be retrieved as a part of the IR.

III. SPECIFICATIONS

In order to accomplish the objectives of the paper , Python has been used as a base language and Natural Language Toolkit (NLTK) and Sklearn libraries have been used.

IV. PROPOSED WORK

This section represents our proposed work and focuses a strategy on retrieving n relevant documents from a free text query with reduced time complexity. The development method is broadly divided into 2 main parts - Language Prediction and Relevant Documents Retrieval. To perform Language Prediction , the data is extracted from varying language-data dictionaries and by web-scraping. Data is then pre-processed through different techniques. TF-IDF Vectorizer Model generates a sparse matrix converting text data into numerical data for better modelling. The vectorized matrix with the unique terms in the dataset as its columns along with message length and punctuation count are treated as the parameters/features for building the classification model. Random Forest Classifier has been used which uses ensemble learning and bagging technique to predict the free text query's language. To perform relevant documents retrieval , a language hash map is used which stores the Language as keys and Corresponding Dataset pertaining to different domains as values in a key-value pair format. The predicted language is passed to the data Hash Map for importing the expected dataset. Data pre-processing is done for making the data suitable for higher accuracy and better modelling. Hash Map technique is proposed for an optimized time complexity and less iterative heavy threads. TF-IDF Vectorizer is modified to consider the Unique Terms along with the hits and timestamp/freshness. The TF-IDF values are then used to calculate the score and cosine similarity between the other vectors/documents and the free text query vector. The cosine values are then sorted in a descending order to retrieve the top n relevant documents from a huge dynamic dataset.

The architectural overview describing an overall process of Information Retrieval which is shown in Fig. 1.

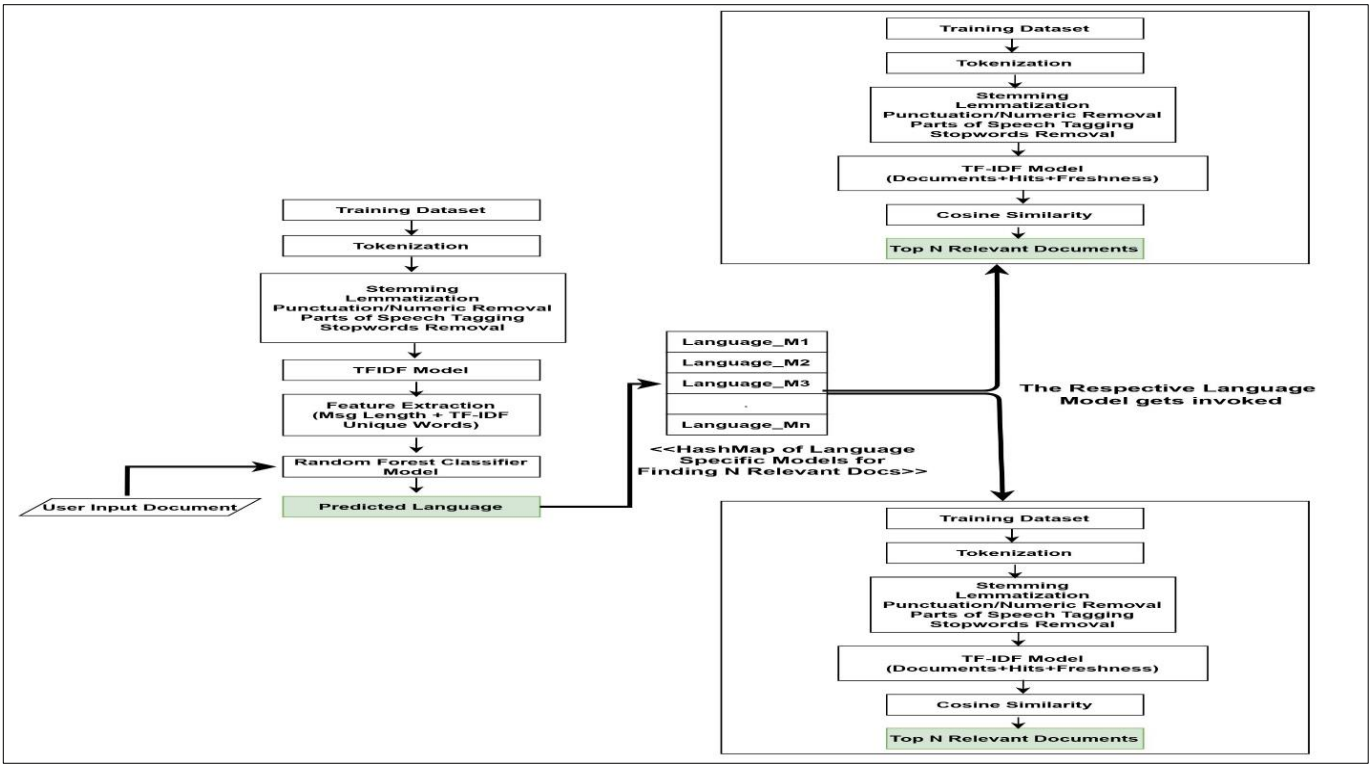


Fig.1 Architecture Diagram

 DOI : 10.35291/2454-9150.2021.0334

Algorithm 1

Input : New Test Document

Output : Predict the Language of Test Document

Method :

1. Import a Training Dataset (Format : Documents and Language)
2. For EachDoc in Training Data:
   Docs ← Tokenize(EachDoc)
   Docs ←RemoveStopwords(EachDoc)
   Docs ←CleanDocs(EachDoc)
   Docs ←WordNetLemmatizer(EachDoc)
3. Vectorizer ← TFIDF Vectorizer()
4. Extract the features for Classification
   X_Features = Unique Terms of the TFIDF Model + Message Length + Punctuation Count
   Y_Original = Given Language in the Dataset
5. Split the Dataset into Training and Testing Data
6. Model ← RandomForestClassifier(Training Data)
7. Language_Prediction ← Pickle the Model.

Algorithm 2

Input : Given a User Query

Output : Retrieve n relevant documents pertaining to the same domain

Method:

1. Language_HashMap = {"Language" : "Training Dataset from different domains"}
   //Training Data Comprising of Data , PageHits , Freshness
2. Storing the Training Dataset in LinkedList because of the dynamic pattern.
3. Passing the New User Query into the Language Prediction Model
   2.1 Predicted_Language = Language_Prediction(User Query) //Invoke the Pickled Model
   2.2 Training_Dataset = Language_HashMap(Predicted_Language)
4. Df ← Retrieve the Training Dataset
5. For EachDoc in Training Data:
   Docs ← word_tokenize()
   Docs ← RemoveStopwords(EachDoc)
   Docs ← CleanDocs(EachDoc)
   Docs ← WordNetLemmatizer(EachDoc)
6. Df ← Df.append(User Query)
7. Vectorizer ← TFIDF Vectorizer(Df(n_terms , page_hits , timestamp))
8. X ← Vectorizer.fit_transform(Df)
9. for col in Vectorizer.nonzero()[1]:
   print(Df[col], ' - ', Vectorizer [0, col])
10. cosine = cosine_similarity(Vectorizer, X)
11. for x in cosine:
    valuessort = x.argsort()[-n:][::-1]

I. Language Identification

Language Identification becomes a crucial pre-requisite in Information Retrieval systems where it is common to index documents in a multilingual collection by the language they are written in. This feature helps in reducing the time complexity with the indexing property in IR systems. The main steps involved in Language Identification are : Data Extraction ,Data Tokenization , Data Preprocessing , TF-IDF Vectorization and Classification.

Data Extraction is an important aspect of any machine learning algorithms since it forms the base for building efficient models and deriving better accuracy scores. Data is gathered from different sources , data dictionaries and through web scraping. [1]. NLTK library is used to perform the text pre-processing. The documents are broken into tokens using the word_tokenize() package supported by NLTK. Stopwords are the most frequently used words in every language and do not add in important information in terms of performing any analytical tasks or cater to a little value in helping select documents matching a user need, so they are removed as a part of text pre-processing. Numeric characters and special characters are often present in any

 DOI : 10.35291/2454-9150.2021.0334

documents but they additionally don't contribute to any efficiency of the model and are hence removed. Lemmatization refers to doing things properly with the use of a vocabulary and morphological analysis of words , normally aiming to remove the inflectional endings only and to return the base form of a word/lemma.

Classifiers cannot be directly built on text data. Hence different weighting schemes convert the text data into numerical form for different models to work on. We have used TF-IDF Vectorization which is a product of Number of times a term occurring in document d and Number of documents in the collection that contain the term t.

Model_Pipeline(TF-IDF Vectorizer , Classifier)

Any document in a vector space model is expressed by :

D1 : <T1 T2 T3 T4 T5 T9 T2 T3 T5>

D2 : <T2 T5 T6 T7 T8>

	T1	T2	T3	T4	T5	T6	T7	T8	T9
D1	1	2	2	1	2	0	0	0	1
D2	0	1	0	0	1	1	1	1	0

This is a count vectorizer model which considers the Raw Term Frequency in a document. We run into certain challenges like bias towards more stop words/more frequently occurring terms in the document , hence the proposed and widely used solution is Term Frequency – Inverse Document Frequency.

TF-IDF Weighting Technique is given by the mathematical formula.

$$Tf\text{-}idf(t,d) = (1+\log(tf)) * \log(N/df)$$

where,

tf : number of times a term t appears in a document

N : total number of documents in the collection

df : number of documents that contains the term t

After applying the TF-IDF Vectorizer , the corpus now contains Bag of Words (all unique terms in any given order).

For feature selection, we add 2 more factors like document length and punctuation count.

X (Dependent Variable)= Document Length(x1) + Punctuation Count(x2) + Bag of Words (x3…xi)

Y (Target Variable) = Language

The corpus is then classified into training and testing data and different classifiers are built on the training dataset. Models are fit on the training data and prediction is done on the test data. Different Accuracy Scores are generated from applying different modelling techniques. The Accuracy chart is given in the Fig.2 and is proved that Random Forest Classifier worked best on our dataset.

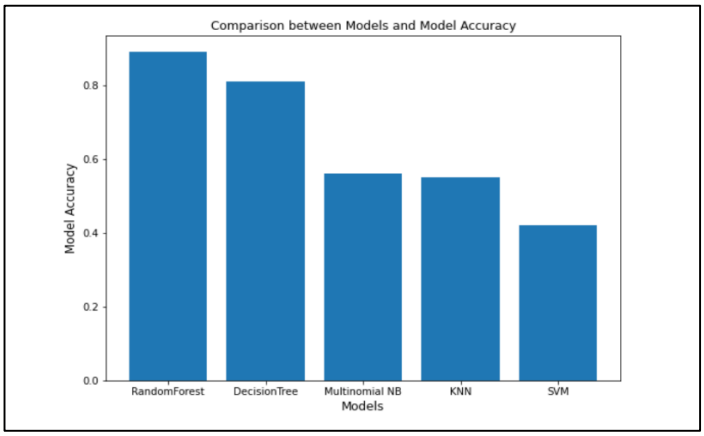


Fig.2 Accuracy Report

The Classification Report is generated for all the predictor variables/languages to check their precision , recall , f1-score and support. The report is given in Fig.3

 DOI : 10.35291/2454-9150.2021.0334

	precision	recall	f1-score	support
Chinese	0.98	0.98	0.98	2262
Danish	0.80	0.86	0.83	243
Dutch	0.88	0.70	0.78	213
English	0.78	0.85	0.81	365
French	0.97	0.92	0.94	323
German	0.85	0.94	0.90	429
German	0.00	0.00	0.00	0
Italian	0.94	0.91	0.92	262
Japanese	1.00	0.97	0.99	153
Korean	1.00	0.84	0.91	282
Norwegian	0.97	0.47	0.63	141
Portuguese	0.77	0.94	0.85	770
Spanish	0.96	0.76	0.85	439
accuracy			0.91	5882
macro avg	0.84	0.78	0.80	5882
weighted avg	0.92	0.91	0.91	5882

Fig.3. Accuracy Report of Random Forest Classifier

The Language Identification Pipeline through TF-IDF Vectorization and Classification was able to successfully predict the languages in the validation dataset or a free text query with a model accuracy of 91%.

II. Dynamic Indexing

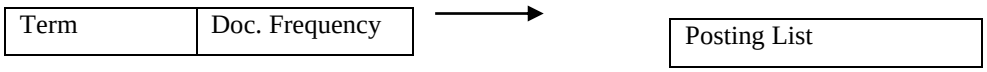
All Information Retrieval systems work with huge terabytes of data which is non-static and gets updated frequently or at periodic intervals. Fault tolerant systems are used for building indexes in IR. The preferred data structure used is B+ Tree or a Linked List because the data size length is not fixed. Most collections are modified frequently with documents being added, deleted, and updated , either new terms need to be added to the dictionary or postings lists need to be updated for existing terms. We have 2 main indexes , Big Index and Small Index(Auxiliary Index) through which we overcome the challenge of dynamic data. A HashMap is maintained in terms of a key-value pair of language and the corresponding varying domain dataset. The identified language of the free text query obtained from the Language Identification Process is then iterated through the HashMap for extracting the corresponding dataset which will be required for the next processing tasks.

III. Relevant Documents Retrieval

Information Retrieval systems operate on the underlying algorithm of finding documents of an unstructured nature that satisfies the an user requirement. To process large documents in an optimal and a shorter time , allow more flexible matching operations and allow ranked retrieval , linear scanning of texts is not a possible best solution. Hence Ranked Retrieval models have been introduced to cater to a free text query with better accuracy and a shorter turn around time. Text Pre-processing is done on the language specific dataset that has been transferred in the Main Memory. Pre-processing involves tokenization of documents into tokens/terms followed by stop word/punctuation removal followed by lemmatization.[6].

In Boolean Retrieval Model , the documents are represented as vectors of indexed terms. Retrieval is based on membership in sets through binary retrieval function i.e 0 when the term doesn't appear in the document , 1 when the term appears in the document. Boolean Retrieval is closely associated with Inverted Index Construction. Inverted Index is represented as Term, Document Frequency and a Posting List. Boolean Retrieval Systems often create a large sparse inverted matrix and is computationally weak and takes a lot of time for processing tasks.

Structure of Inverted Index:



When a free text query is entered by a user , the documents are expected to be retrieved in a Ranked Order , so different scoring and weighting techniques are used.

Jacard Coefficient Similarity between two dataset is the ratio between the number of terms common to all and all the unique terms.

Jacard Similarity (Query , Document) = (Query AND Document) / (Query UNION Document)

After certain experiments , it has been observed that Jaccard Coefficient is bias towards shorter length documents and does not consider the Term Frequency in a document. Sometime rare words in a collection are more informative than frequent terms which is not considered by Jaccard Coefficient. Vector Space Modelling is a widely used technique in Information Retrieval Systems. In Vector Space Modelling , the terms are represented as basis vectors which are linearly independent and orthogonal to each other. Basis vectors correspond to dimensions or directions in the vector space. Vector Coefficients are represented in the following fig.4

 DOI : 10.35291/2454-9150.2021.0334

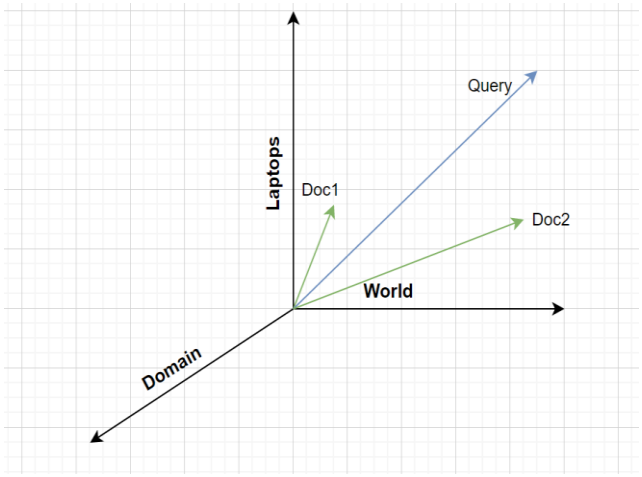


Fig.4 Vector Space Modelling

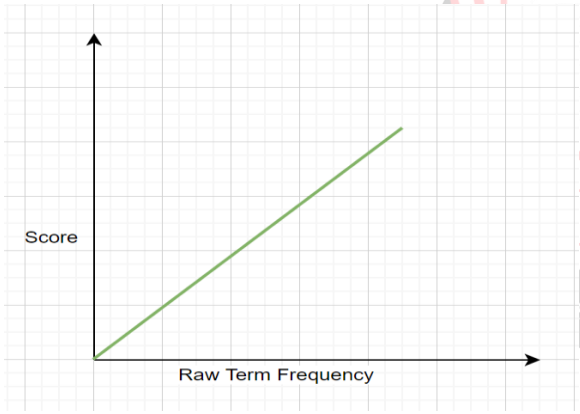
Doc1: Laptops Domain World <1 1 1>

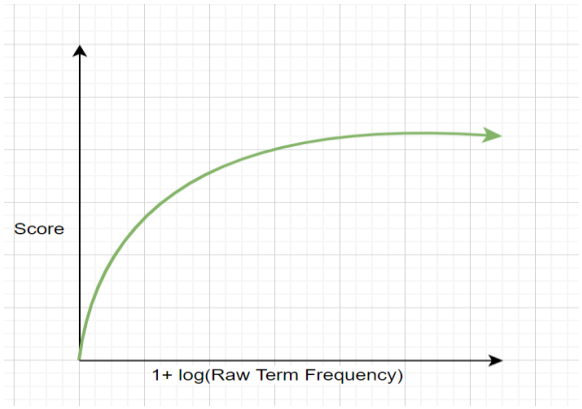
Doc2: World Laptops Laptops Domain Domain Domain <1 2 3>

Query: World Laptops Laptops Domain Domain<1 2 2>

Vector Coefficients represent the importance of a term but not help assign weights to the term.

Commonly used coefficients are Raw Term Frequency ,Term-Frequency—Inverse Document Frequency etc. Raw Term Frequency denotes the occurrence of frequency of terms in a document and specifies the relative importance of a word in a document. The challenge with Raw Term Frequency is about stop words or different parts of speech which are frequently occurring but might not be important in a document. The relation between Score and Raw Term Frequency is show in Fig.6 which shows the score and the coefficient is directly proportional to the occurrence of terms in a document. But for an ideal behavior , after the repetition of terms in a document , the score should be constant approximately. This scenario happens when 1+RTF is taken instead of RTF for calculating the score.





In this paper , the proposed technique is TF-IDF weighting scheme. This weighting scheme is modified to also consider freshness/timestamp and number of votes for each document. The rarity of a term is given by Inverse Document Frequency. DF_t

 DOI : 10.35291/2454-9150.2021.0334

is the document_frequency of *t*: the number of documents that contain *t*. We define the IDF (inverse document frequency) of *t* by $\log_{10}$ (N/ DF_t).

The proposed formula for TF-IDF in this paper to consider the changing dataset , the latest trends , freshness/timestamp and also the number of votes for each document (how other users have browsed the document) is given below.

$$\text{Tf-IDF}_{\text{updated}} = ((1+ \log(TF)) * \log_{10}(N/DF) + (1+\log_{10}(\text{Freshness}))+ (1+\log_{10}(\text{Number of Votes}))$$

where,

TF : number of times a term t appears in a document

N : total number of documents in the collection

DF : number of documents that contains the term t

The TF-IDF Weighting is calculated for all terms in the document/Bag of Words of the corpus. To retrieve the top relevant documents , the cosine similarity needs to be calculated between the documents in a collection and the free text query.

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

where,

q : free text query

d : documents in the corpus

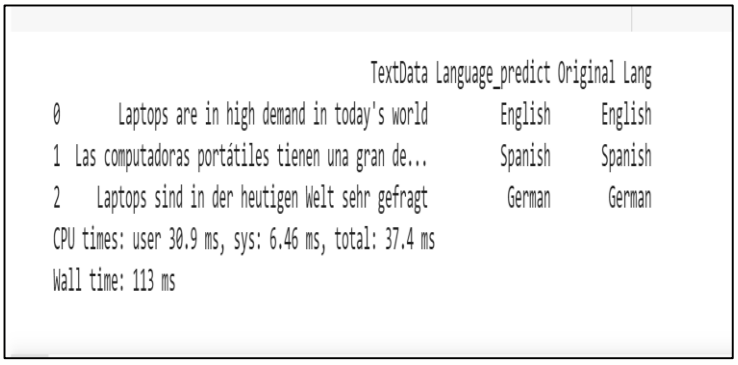
Cosine Similarity / Score is calculated by the given mathematical formula.

$$\text{Cosine\_Similarity (q , d)} = \frac{\sum q \cdot d_i}{\sqrt{\sum {d_i}^2} \cdot \sqrt{\sum q^2}}$$

Score calculated is sorted in a descending order to provide the top most relevant trending and most user recommended documents for a free text query entered by the user.

V. RESULT

From the language identification algorithm which was built using TF-IDF Vectorizer and Random Forest Classifier, the language was identified for further operations. A Validation dataset was created to predict the language using the pickled TF-IDF and Random Forest Classifier Model. The Validation report is given in Fig.5.



	TextData	Language_predict	Original Lang
0	Laptops are in high demand in today's world	English	English
1	Las computadoras portátiles tienen una gran de...	Spanish	Spanish
2	Laptops sind in der heutigen Welt sehr gefragt	German	German

CPU times: user 30.9 ms, sys: 6.46 ms, total: 37.4 ms
Wall time: 113 ms

Fig.5. Language Output Report

The predicted language is then passed through the IR modelling to retrieve the most relevant , domain-specific , current trending and most frequently accessed documents which matches the user specified query. The results are shown below for the free text query "*Covid19 vaccines*". Top n Relevant Documents were retrieved for a given free text query in Fig.6 .

 DOI : 10.35291/2454-9150.2021.0334

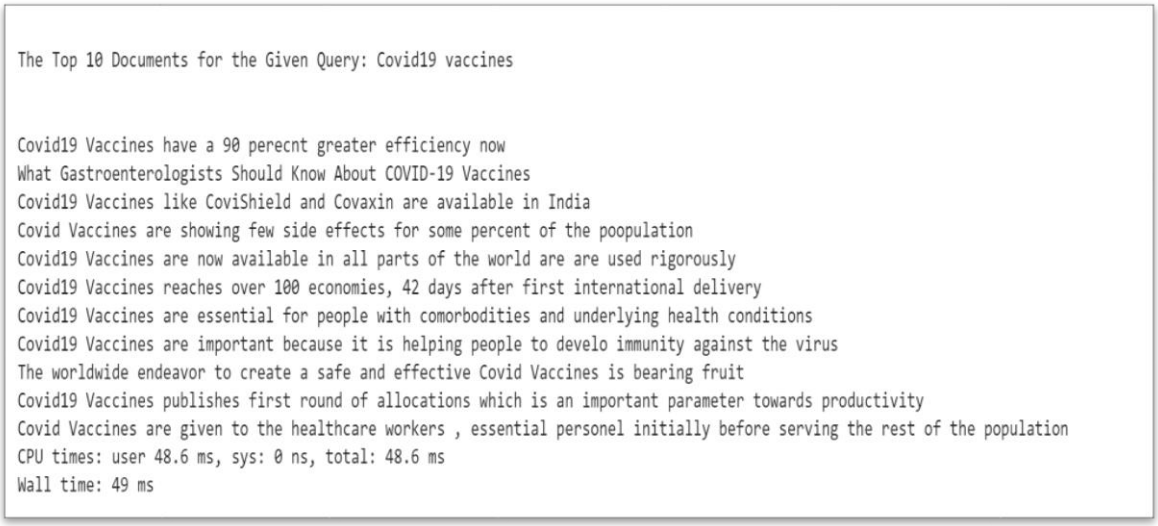


Fig.6. Top 10 Relevant Documents

VI. CONCLUSION

Although Information Retrieval systems exist and caters to the user need of providing the relevant documents , our proposed solution works in an efficient manner with a lesser time and space complexity. Our solution is built using NLTK and Scikit Learn Packages. In this paper we have developed Language Identification system using TF-IDF Vectorizer and a classifier model , followed by a modified vector space model that considers the number of votes and timestamp along with the Bag of Words for TF-IDF matrix computation and retrieves the top relevant documents to the user using cosine similarity scoring technique. The Language Classifier Model worked with an average accuracy of 91% to predict the language. The intermediate hashmap technique used between the two modelling techniques has reduced the time complexity and helped in retrieving the results faster.

REFERENCES

[1] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. *Introduction to Information Retrieval* 2008 Cambridge University Press

[2] Suphakit Niwattanakul*, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu *Using of Jaccard Coefficient for Keywords Similarity* Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong

[3] Davood Rafiei , Haobin Li *Wild Card Queries for Searching Resources on the Web* arXiv:0908.2588v1 [cs.DB] 18 Aug 2009

[4]Shahzad Qaiser , Ramsha Ali *Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents*

International Journal of Computer Applications (0975 – 8887)

[5] Qiong Ren a), Hui Cheng b) and Hai Han *Research on Machine Learning Framework Based on Random Forest Algorithm*

[6] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, Noor Akhmad *Setiawan Cosine Similarity to Determine Similarity Measure: Study Case in Online Essay Assessment* IEEE

[7] Liu, C., Sheng, Y., Wei, Z., & Yang, Y.-Q. (2018). *Research of Text Classification Based on Improved TF-IDF Algorithm.* 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)

 DOI : 10.35291/2454-9150.2021.0334