

# **TEXT MINING OF SCIENTIFIC PUBLICATIONS**

**A PROJECT REPORT**

*Submitted by*

**Srinivas T R 2017115601**

**Rithvik A V S 2017115579**

*submitted to the Faculty of*

**INFORMATION AND COMMUNICATION ENGINEERING**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY**

**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY**

**CHENNAI 600 025**

**ANNA UNIVERSITY**  
**CHENNAI - 600 025**  
**BONA FIDE CERTIFICATE**

Certified that this project report Text mining of Scientific Publications is the bona fide work of Srinivas T R, Rithvik A V S who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

**PLACE:CHENNAI**

**DATE:23/11/2020**

**Dr.Saswathi Mukherjee**

**PROFESSOR**

**PROJECT GUIDE**

**DEPARTMENT OF IST, CEG**

**ANNA UNIVERSITY**

**CHENNAI 600025**

**COUNTERSIGNED**

**Dr. SASWATI MUKHERJEE**

**HEAD OF THE DEPARTMENT**

**DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY**

**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY**

**CHENNAI 600025**

## ABSTRACT

The amount of research work taking place in all streams of Science, Engineering, Medicines, etc., is growing rapidly and hence the research articles that are being published are increasing day by day. In this rapidly growing environment, it is very exhaustive to identity, maintain and manually classify a large volume and collection of articles. One of the main concern of many students, researchers and professors is finding scientific papers and articles relevant to a particular area of research or category. Classification of papers into the relevant categories makes the process easier. This task can be done manually by asking authors of journals to assign one or more categories during publishing time. However, classifying a large collection of scientific articles manually is a time consuming process. In automatic methods, by default a keyword-based search is used for the subject term in paper's title, or text which is a naive strategy. Though this approach works in some cases it fails to identify articles which contain semantically equivalent term but not exactly the same words. Also, processing the whole text of a research paper takes a long time. This project proposes an approach for classifying such huge volume of articles by using deep learning classification methods and recommending similar research papers using content based and collaborative filtering. Since one research paper might belong to different categories simultaneously, the need for multi label classification arises. Various approaches to word embeddings, neural networks and multi label classification using different methods are studied and implemented.

## TAMIL ABSTRACT

அறிவியல், பொறியியல், மருந்துகள் போன்ற அனைத்து நீரோடைகளிலும் நடைபெற்று வரும் ஆராய்ச்சிப் பணிகளின் அளவு வேகமாக வளர்ந்து வருகிறது, எனவே வெளியிடப்படும் ஆய்வுக் கட்டுரைகள் நாளுக்கு நாள் அதிகரித்து வருகின்றன. வேகமாக வளர்ந்து வரும் இந்த சூழலில், இது அடையாளத்திற்கு மிகவும் முழுமையானது, ஒரு பெரிய தொகுதி மற்றும் கட்டுரைகளின் தொகுப்பை பராமரித்தல் மற்றும் கைமுறையாக வகைப்படுத்துதல். பல மாணவர்கள், ஆராய்ச்சியாளர்கள் மற்றும் பேராசிரியர்களின் முக்கிய கவலைகளில் ஒன்று, ஒரு குறிப்பிட்ட பகுதி ஆராய்ச்சி அல்லது வகைக்கு பொருத்தமான அறிவியல் ஆவணங்கள் மற்றும் கட்டுரைகளைக் கண்டுபிடிப்பது. ஆவணங்களை தொடர்புடைய வகைகளாக வகைப்படுத்துதல் செயல்முறையை எளிதாக்குகிறது. வெளியீட்டு நேரத்தில் ஒன்று அல்லது அதற்கு மேற்பட்ட வகைகளை ஒதுக்குமாறு பத்திரிகைகளின் ஆசிரியர்களைக் கேட்பதன் மூலம் இந்த பணியை கைமுறையாகச் செய்ய முடியும். ஆனால், ஒரு பெரிய அறிவியல் கட்டுரைகளை கைமுறையாக வகைப்படுத்துவது நேரத்தை எடுத்துக்கொள்ளும் செயல்முறையாகும். தானியங்கி முறைகளில், இயல்புநிலையாக ஒரு முக்கிய சொல் அடிப்படையிலான தேடல் என்பது காகிதத்தின் தலைப்பில் உள்ள பொருள் காலத்திற்கு அல்லது ஒரு அப்பாவிடாக இருக்கும் உத்தி ஆகும். இந்த அணுகுமுறை சில சந்தர்ப்பங்களில் செயல்பட்டாலும் அது தோல்வியடைகிறது சொற்பொருளுக்கு சமமான சொற்களைக் கொண்ட கட்டுரைகளை அடையாளம் காணுங்கள், ஆனால் அதே சொற்களைக் கொண்டிருக்கவில்லை. மேலும், ஒரு ஆய்வுக் கட்டுரையின் முழு உரையையும் செயலாக்குவதற்கு நீண்ட நேரம் எடுக்கும். ஆழ்ந்த கற்றல் வகைப்பாடு முறைகளைப் பயன்படுத்தி பரிந்துரைப்பதன் மூலம் இத்தகைய பெரிய அளவிலான கட்டுரைகளை வகைப்படுத்துவதற்கான அணுகுமுறையை இந்த திட்டம் முன்மொழிகிறது. உள்ளடக்க அடிப்படையிலான மற்றும் கூட்டு வடிகட்டலைப் பயன்படுத்தி ஒத்த ஆய்வுக் கட்டுரைகள். ஒரு ஆய்வுக் கட்டுரை ஒரே நேரத்தில் வெவ்வேறு வகைகளைச் சேர்ந்ததாக இருப்பதால், பல லேபிள் வகைப்பாட்டின் தேவை எழுகிறது. சொல் உட்பொதிப்புகள், நரம்பியல் நெட்வொர்க்குகள் மற்றும் பல்வேறு முறைகளைப் பயன்படுத்தி பல லேபிள் வகைப்பாடு ஆகியவற்றிற்கான பல்வேறு அணுகுமுறைகள் ஆய்வு செய்யப்பட்டு செயல்படுத்தப்படுகின்றன.

## ACKNOWLEDGEMENT

First and foremost, we would like to express our deep sense of gratitude to our guide **Dr.Saswati Mukherjee**, Professor Department of Information Science and Technology, Anna University for their excellent guidance,counsel,continuous support and patience. They helped us with this topic and guided us in the development of this project. They gave us the moral support to finish out creative and innovative project in a successful manner.

We would also like to express our gratitude to **Dr.Saswati Mukherjee**, Head of the Department, Department of Information Science and Technology,Anna University,for supporting us with the technical resources required for our project. We express our heartiest thanks to all the other teaching and non teaching staffs who have helped us in the successful completion of the project. We would also like to thank our parents and friends for their indirect contribution in the successful completion of the project.

**SRINIVAS T R**  
**RITHVIK A V S**

# TABLE OF CONTENTS

<b>ABSTRACT</b>	iii
<b>ACKNOWLEDGEMENT</b>	v
<b>LIST OF FIGURES</b>	ix
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Text Mining	1
1.2 Multi Label Classification	1
1.3 Recommendation System	2
1.4 Motivation	3
1.5 Problem Statement	3
1.6 Objective	3
1.7 Organisation of the report	3
<b>2 LITERATURE SURVEY/RELATED WORK</b>	<b>5</b>
2.1 Text Classification	5
2.1.1 Attention-Based Bidirectional Long Short-Term	
Memory Networks for Relation Classification	5
2.1.2 Convolutional Neural Networks for Sentence	
Classification	5
2.1.3 BERT: Pre-training of Deep Bidirectional Transformers	
for Language Understanding	6
2.2 Recommendation System	6
2.2.1 A collaborative approach for research paper	
recommender system	6
2.2.2 Paper Recommendation System	7
2.2.3 A Collaborative Filtering based Recommender	
System for Suggesting New Trends in Any Domain	
of Research	7
<b>3 DESIGN</b>	<b>9</b>
3.1 Multilabel text classification	9
3.2 Data Wrangling	10
3.3 Text Preprocessing	10
3.4 Tokenization and padding	11

3.5	Word Embeddings	11
3.5.1	GloVe	11
3.5.2	Fasttext	12
3.6	Multi Label text Classification using deep learning	12
3.6.1	Long Short Term Memory Networks	13
3.6.2	Bi-directional LSTM	14
3.6.3	Bi-directional LSTM with self attention	14
3.6.4	TextCNN	15
3.6.5	CNN-BiLSTM	15
3.6.6	BiLSTM-CNN	16
3.6.7	Encoder Decoder with teacher forcing	16
3.6.8	Bidirectional Encoder Representations from Transformers	16
3.7	Recommendation System	17
3.7.1	Collaborative Filtering	17
3.7.2	Content Based Filtering	17
<b>4</b>	<b>IMPLEMENTATION</b>	<b>21</b>
4.1	Tools Used	21
4.1.1	Pandas	21
4.1.2	Dask	21
4.1.3	NLTK	21
4.1.4	Swifter	21
4.1.5	Google Colab Notebooks	22
4.1.6	Tensorflow	22
4.1.7	Fasttext	22
4.2	Dataset Used	22
4.3	Data Wrangling	23
4.3.1	Format Conversion	23
4.3.2	Label Cleaning	23
4.3.3	Sampling	24
4.4	Data Preprocessing	24
4.5	Tokenization and Padding	26
4.6	Word Embeddings	27
4.7	Multi label text classification using deep learning	27
4.7.1	LSTM	28
4.7.2	Bidirectional LSTM	28

4.7.3 Bidirectional LSTM with self attention	29
4.7.4 TextCNN	31
4.7.5 CNN-LSTM	31
4.7.6 LSTM-CNN	32
4.7.7 Encoder Decoder with teacher forcing	33
4.7.8 BERT	33
4.8 Recommendation System	35
4.8.1 Dataset	35
4.8.2 Content Based Recommendation	37
4.8.3 Collaborative Based Recommendation	38
<b>5 RESULTS</b>	<b>39</b>
5.1 Results for multi label text classification	39
5.1.1 Metrics for evaluating multi label classification	39
5.1.1.1 Hamming Loss	39
5.1.1.2 Subset Accuracy	40
5.1.1.3 Inference	40
5.2 Results for recommendation system	41
<b>REFERENCES</b>	<b>43</b>



## LIST OF FIGURES

3.1 Overall flow diagram	10
3.2 Content Based Filtering	18
3.3 Collaborative Based Filtering	19
4.1 Dataset	24
4.2 Count of categories	25
4.3 Sampled data	26
4.4 Clean Abstract	26
4.5 Tokenized data	27
4.6 Matrix generated using Glove Embeddings	28
4.7 Matrix generated using Fasttext Embeddings	29
4.8 Distribution of sentence lengths	29
4.9 Summary of the LSTM model	30
4.10 Summary of the Bi-LSTM model	30
4.11 Summary of the Bi-LSTM Attention model	31
4.12 Summary of the TextCNN model	32
4.13 Summary of the CNN-LSTM model	33
4.14 Summary of the LSTM-CNN model	34
4.15 Flow diagram of the Encoder decoder model	34
4.16 Flow diagram of BERT	35
4.17 TF-IDF Scores of TensorFlow	37
4.18 Popularity Scores of Tensorflow	38
5.1 Results	40
5.2 Collaborative Based Recommendation System	41
5.3 Content Based Recommendation System	42

# **CHAPTER 1**

## **INTRODUCTION**

This chapter gives an overview of the text mining, multi label classification, recommendation system and the research that has been done over the last few years. Furthermore this briefly explains the motive, challenges faced in developing this method.

### **1.1 Text Mining**

Research in text mining has become one of the most widespread fields in analysing the natural language documents. Almost all data of different institutions across the world is stored in electronic documents in the form of semi structured data. Thus, the need for text mining arises which is different from data mining. Data mining is focused on discovering interesting patterns and statistical information from large databases that contain numerical data rather than information in the form of text. Text mining intends to detect the information that was not recognized before through extracting it automatically from various text-based sources. Structured data can be handled through data mining tools and methods while unstructured or semi-structured datasets like full-text documents, emails, and HTML files can be efficiently handled through text mining. Text refining and knowledge distillation are two consecutive stages involved in text mining. In text refining, free-form text documents are converted into an intermediate form, whereas in knowledge distillation, patterns or knowledge are derived from intermediate form.

## **1.2 Multi Label Classification**

In Machine Learning, and particularly in supervised learning, classification is one the most important learning techniques. In the traditional task of singlelabel classification each example is associated with a single class label. A classifier learns to associate each new test example with one of these known class labels. When each example may be associated with multiple labels, this is known as multi-label classification. Multilabel classification is a challenging research problem that emerges in several modern applications such as music categorization, protein function classification and semantic classification of images. In the past, multilabel classification has mainly engaged the attention of researchers working on text categorization, as each member of a document collection usually belongs to more than one semantic category. Multilabel classification methods can be categorized into two different groups

- Problem transformation methods
- Algorithm adaptation methods.

The first group of methods are algorithm independent. They transform the multilabel classification task into one or more single-label classification, regression or label ranking tasks. The second group of methods extend specific learning algorithms in order to handle multilabel data directly.

## **1.3 Recommendation System**

It is important that not only Research Papers are classified but also be able to recommend the right paper to the user. In order to recommend the most suitable papers upon users search, two different approaches are to be explored.

The Content Based and Collaborative based filtering are to be explored for the Recommendation of Research Papers.

#### **1.4 Motivation**

There has been a limitation in categorizing research articles automatically. Most of the categorization is based on just the keywords present in the research papers. These are very inefficient. And also, classifying archived research papers into categories manually is very inefficient and time consuming task.

#### **1.5 Problem Statement**

A research paper may belong to more than one category at the same time. In existing literature, many text mining and machine learning classification methods have been applied for classifying research documents. But there is a limitation in addressing multi label classification of scientific documents. Recommendation of Research Papers plays a vital role in transfer of knowledge to user, hence the need to develop methods that is able to achieve the same.

#### **1.6 Objective**

The project aims to find an appropriate and efficient method for multi label text classification of scientific documents and build a recommendation system.

#### **1.7 Organisation of the report**

This report explains the proposed methodologies, tools and the resultant outcome of the project.

## **CHAPTER 2**

### **LITERATURE SURVEY/RELATED WORK**

Following papers have been referred to gain insights into the existing methods of multi-label classification and recommendation systems.

#### **2.1 Text Classification**

##### **2.1.1 Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification**

Relation classification is an important semantic processing task in the field of natural language processing (NLP). Even now state-of-the-art systems rely on lexical resource, WordNet, or NLP systems like dependency parser and named entity recognizers (NER) to obtain high-level features. Another important challenge is the ability to identify the right words as they could be in any position of the sentences. Therefore in order to tackle those problems, they proposed Attention-Based Bidirectional Long Short-Term Memory Networks (Att-BLSTM) to capture the most important semantic information in a sentence[1]. Their experimental results on the SemEval-2010 relation classification task showed that the method they developed outperformed most of the existing methods, with only word vectors.

##### **2.1.2 Convolutional Neural Networks for Sentence Classification**

A series of experiments with convolutional neural networks (CNN) were trained on top of pre-trained word vectors for sentence-level classification

tasks. A simple CNN with little hyperparameter tuning and static vectors achieved excellent results on multiple benchmarks. Learning task-specific vectors through fine-tuning offers further gains in performance[2]. They additionally proposed a simple modification to the architecture that allows for the use of both task-specific and static vectors. The CNN models improved upon the state of the art on 4 out of 7 tasks, that included sentiment analysis and question classification.

### **2.1.3 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**

BERT, which stands for Bidirectional Encoder Representations from Transformers is new language representation model. BERT was designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers[3]. Therefore, the pre-trained BERT model could be fine tuned with just one additional output layer to create models for a wide range of tasks, that includes question answering and language inference, without substantial task specific architecture modifications. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement),

## **2.2 Recommendation System**

### **2.2.1 A collaborative approach for research paper recommender system**

The hidden associations between a research paper and its references and citations using paper-citation relations were brought in this paper. The rationale behind their approach is that, if two papers are significantly co-occurring

with the same citing paper(s), then they should be similar to some extent. Their method outperformed the baseline methods in measuring both the overall performance and the ability to return relevant and useful research papers[4]. Their proposed approach has also recorded significant improvements over the baseline methods in providing relevant and useful recommendations at the top of the recommendation list based on mean average precision (MAP) and mean reciprocal rank (MRR).

### **2.2.2 Paper Recommendation System**

A hybrid content and collaborative based recommendation system that recommends the best suitable research paper to the researchers on the basis of content similarity, popularity metrics and the collaborative nature of citations and references was proposed in this paper. The system implemented is also deployed in google cloud to facilitate user interface that could be used by the visitors to find relevant research paper easily. The system was also evaluated by comparing its performance with the recommendation system of the webpage where the data was obtained performed efficiently than the existing recommender system.

### **2.2.3 A Collaborative Filtering based Recommender System for Suggesting New Trends in Any Domain of Research**

The recommender system proposed here uses three major factors used for building this system which includes datasets, prediction rating based on users and cosine similarity. The ratings are made by user which will be determined by the number of accurate ratings they provide. The results are then sorted by using cosine similarity[5]. A research-paper recommender system using collaborative filtering approach to recommend a user with best research papers in their domain according to their queries and based on the similarities



found from other users on the basis of their queries, which will help in avoiding time consuming searches for the user.

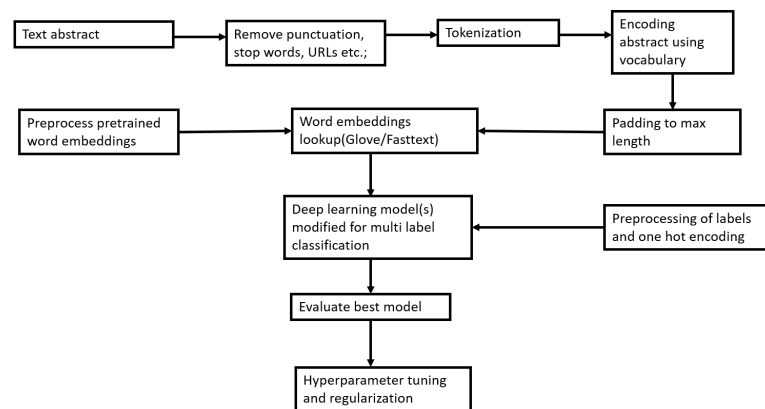
## CHAPTER 3

### DESIGN

#### 3.1 Multilabel text classification

For multilabel text classification the following modules are used.

- Data Wrangling
- Text preprocessing
- Tokenization and padding
- Preprocessing word embeddings
- Constructing Word embeddings
- Multi label classification using deep learning
- Hyperparameter tuning and regularization



**Figure 3.1: Overall flow diagram**

Fig 3.1 shows the overall flow diagram for multi-label classification of scientific publications.

## 3.2 Data Wrangling

In the data wrangling phase, the data is acquired, required fields are chosen and converted into a comma separated value file format. The dataset is also sampled.

## 3.3 Text Preprocessing

Text preprocessing consists of the following stages:

- Removing punctuations, whitespaces, URLs,numbers etc;
- Tokenization
- Stopwords removal

A stop word is a commonly used word such as “the”, “a”, “an”, “in” etc;. These are removed from the sentence after tokenization. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but the words

produced after lemmatization are present in the vocabulary which is not the case for stemming.

### **3.4 Tokenization and padding**

Before feeding the text into the neural network it should be converted into a numerical form. Tokenization splits the word into tokens and maps the tokens to numbers. Padding is done so that all sentences have the same length in a particular batch.

### **3.5 Word Embeddings**

Word embeddings is a form of learned text representation. Each word in the vocabulary is converted into a vector of specified dimension. Words that have a similar meaning share a similar representation in a vector space when they are plotted. Usually, word embeddings are learnt using neural networks. A particular word is masked and it is predicted by using the neighbouring words. The resultant weights of the neural network is considered as the embedding of the word. Transfer learning which is the use of pretrained models for machine learning tasks is very effective for text representation. For the project, two pretrained word embeddings - GloVe: Global Vectors for Word Representation[6] and fasttext[7] are used.

#### **3.5.1 GloVe**

GloVe stands for “Global Vectors”. GloVe captures both global statistics and local statistics of a corpus, to form embeddings. GloVe constructs a co-occurrence matrix to come up with semantic relations between words.

The following are the steps in GloVe algorithm:

- A co-occurrence matrix is constructed based on the text. Every element  $X_{ij}$  denotes the number of times  $i$  appears in the context of  $j$ .
- Soft constraints are defined for each pair where  $W$  denotes the weight for  $i, j$  and  $b$  the bias term.

$$w_i^T w_j + b_i + b_j = \log(X_{ij}) \quad (3.1)$$

- The cost function is finally defined as

$$J = \sum_{i=1}^V \sum_{j=1}^V f(X_{ij}) (w_i^T w_j + b_i + b_j - \log X_{ij})^2 \quad (3.2)$$

- $f$  is a function chosen by the authors of Glove denoted by

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_{max}}\right)^\alpha & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases} \quad (3.3)$$

### 3.5.2 Fasttext

FastText is a library written in c++ for efficient learning of word representations and sentence classification and was released by facebook. In fasttext, each word is represented as a bag of character n-grams in addition to the word itself. Thus, for out of vocabulary words the word representations are found by averaging the character level n-grams.

## 3.6 Multi Label text Classification using deep learning

To achieve multi-label classification the final layer of the neural network is configured with number of neurons equal to the total number of distinct labels. Each neuron is activated with a sigmoid function.

$$h_{\theta}(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function outputs a number between 0 and 1 which is rounded off. 0 denotes the absence of a particular label and 1 denotes the presence. Binary cross entropy loss is calculated for each neuron and the average across all the neurons is considered as the cost which is optimized. The binary cross entropy loss is given by:

$$Loss = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3.4)$$

where  $p$  denotes the prediction and  $y$  the ground truth. The following neural networks are studied and implemented model for multi label text classification.

- Long Short Term memory networks (LSTM)
- Bidirectional LSTM
- Bidirectional LSTM with Self attention
- TextCNN
- CNN-BiLSTM
- BiLSTM-CNN
- LSTM based Encoder decoder for sequence generation of labels
- Bidirectional Encoder Representations from Transformers(BERT)

### 3.6.1 Long Short Term Memory Networks

LSTM [8] was proposed to solve the problem of vanishing and exploding gradient that occurred in recurrent neural networks during backpropagation. Hidden states are passed between successive time steps similar to RNNs. LSTMs use three different gates namely:- input gate, forget gate and output gate. These gates decide which information to pass and discard between successive time steps.  $X^t$  denotes the input to the LSTM cell and  $i, f$  and  $o$  denotes input gate, forget gate and the output gate respectively.

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t) \quad (3.5)$$

$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t) \quad (3.6)$$

$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t) \quad (3.7)$$

$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t) \quad (3.8)$$

### 3.6.2 Bi-directional LSTM

In a traditional LSTM information is propagated only in one direction. During text classification in an LSTM, each cell can look only at words that are before it but not ahead of it. Bi-directional LSTMs [9] solve this problem by using two sequences of LSTM cells, one running in the forward directions from start of the sentence to end of sentence and the other vice versa. Results from both the LSTM layers are concatenated at the end.

### 3.6.3 Bi-directional LSTM with self attention

Given a sentence, not all words have an equal contribution in predicting

the class. The self attention mechanism is used to extract these more important words by giving them a higher weight to increase their importance by calculating the attention score for each word. The BiLSTM layer concatenates information from two opposite directions of a sentence through two independent LSTMs. The self-attention layer [10] captures the more important parts of a sentence. The self attention mechanism is used to extract these more important words by giving them a higher weight to increase their importance by calculating the attention score for each word. A weighted dot product is taken with the hidden vector  $h_t$  and a bias vector is added before applying tanh activation. This is the attention weight that is calculated during backpropagation. Softmax activation is applied on the resultant vector after which it is concatenated with the hidden vector to produce the final context vector.

$$u_t = \tanh(Wh_t + b) \quad (3.9)$$

$$\alpha_t = \text{softmax}(v^T u_t) \quad (3.10)$$

$$s = \sum_{t=1}^M \alpha_t h_t \quad (3.11)$$

#### 3.6.4 TextCNN

Convolutional layers followed by maxpooling with multiple filter widths are used on top of the embedding layer to produce new features. For an image the filters are moved horizontally and vertically, but for text the kernel size is fixed to Filter Size x Embedding size. Max Pooling operation is applied on the resultant features to reduce the dimension. The filter width denotes the number of words that the filter can look at. [11]

#### 3.6.5 CNN-BiLSTM



In a CNN-LSTM, convolution is performed in a sequential fashion over the word embedding and the resultant features are fed into a BiLSTM layer.

### **3.6.6 BiLSTM-CNN**

This model differs from CNN-BiLSTM in the way the filters are applied. In a CNN-BiLSTM, filters are applied sequentially on word embeddings. In BiLSTM-CNN, filters of different sizes are applied parallelly over the output of BiLSTM and concatenated at the end.

### **3.6.7 Encoder Decoder with teacher forcing**

An encoder decoder model consists of two parts - An encoder and a decoder. Both the encoder and decoder are made up of a sequence of LSTM cells. In the above models classification was performed by configuring the last layer of the neural network with number of neurons equal to that of the number of distinct labels. In an Encoder decoder model, the text sequence is encoded by a sequence of LSTM cells and the decoder performs classification by predicting the labels one step at a time [12].

### **3.6.8 Bidirectional Encoder Representations from Transformers**

Bidirectional Encoder Representations from Transformers is a learning technique that is based on the transformer architecture. Although LSTM solved the problem of vanishing and exploding gradients it was not capable of handling long sentences well. Since every LSTM cell had to depend on the previous lstm cell for the input parallelism was not possible. BERT looked to solve these problems by using a pure attention based model to improve parallelism and residual connections to avoid vanishing and exploding gradients. BERT uses a

variant of self attention called multi head attention which improves the accuracy of attention computation.

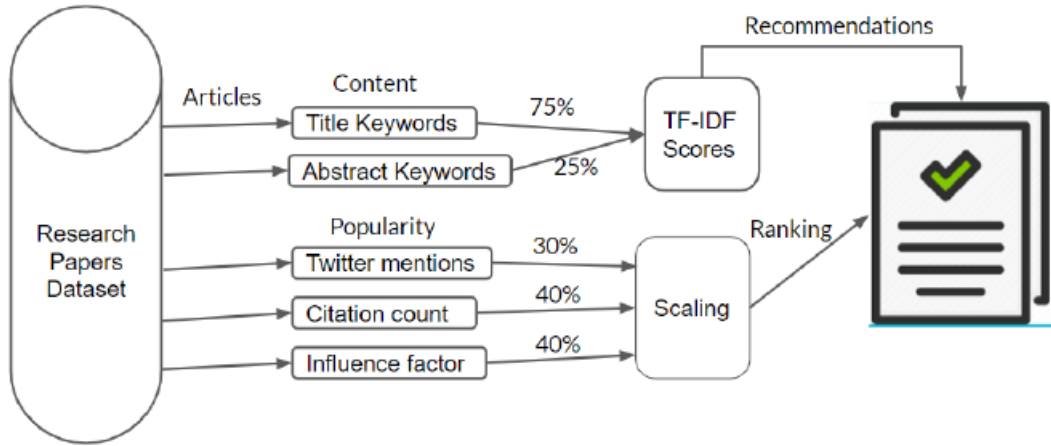
### **3.7 Recommendation System**

In this section, the design of the models for recommendation system is explained. The strengths of both Content-based approach and Collaborative Filtering are tested pipeline to give the best recommendations to the user.

#### **3.7.1 Collaborative Filtering**

In this model, item-based filtering is performed to obtain recommendations. The model doesn't use any user data for performing item-based filtering, but uses citation analysis to find the similarity between documents. The base paper for which we look for similar papers is obtained through user's click(implicit).  
 1.(base paper) 2.(Potential recommendable paper) 3. (Another paper) If ((2 has cited papers cited in 1) (3 has cited both 1 and 2)) then (use 2 to recommend for those who view 1).

We first check all the potential recommendable papers from the corpus based on the first condition here in the if check. To do this we match every document's references in the corpus with the references of the base paper to identify documents which have at least one reference in common. Once we have the list of potential recommendable papers, we further check the second condition in the if check to obtain the final list of papers that can be recommended. Through this citation match analysis we try to find the best close documents which could be used together or are related to one another. The more papers that have cited them together the better close they are.



**Figure 3.2: Content Based Filtering**  
Flow of content based filtering method

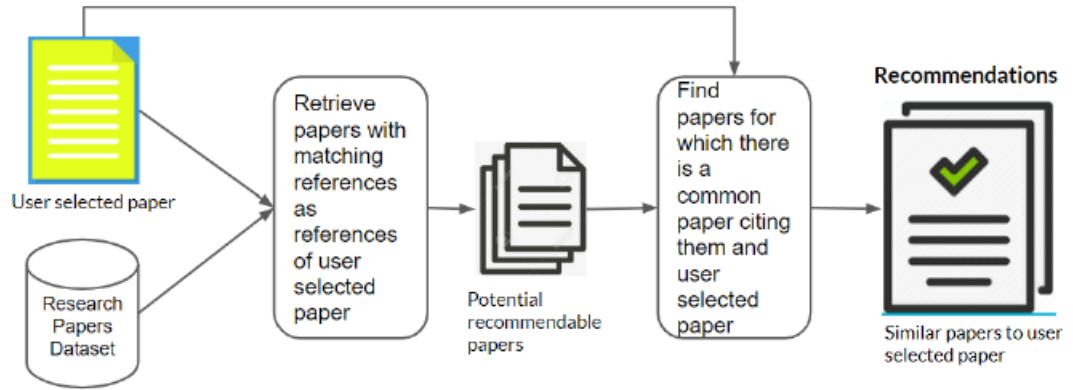
### 3.7.2 Content Based Filtering

In this model we have used the TF-IDF (term frequency-inverse document frequency) for information retrieval and text mining. The text from the user search is fetched and stop words are removed to find keywords from it. We then proceed to find the frequency score for each of the words from keywords in the available corpus. Title and abstract of the articles are used to calculate the TF-IDF scores as our dataset doesn't contain the whole document text. To provide more importance to the title over abstract we have calculated weighted sum of the TF-IDF score, giving title 75 percent and abstract 25 percent weightage respectively.

Typically, the tf-idf weight is composed by two terms-

- Normalized Term Frequency (tf)
- Inverse Document Frequency (idf)

$$tf(t, d) = N(t, d) \quad (3.12)$$



**Figure 3.3: Collaborative Based Filtering**  
Flow of Collaborative based filtering method

wherein  $tf(t, d)$  = term frequency for a term  $t$  in document  $d$ .  $N(t, d)$  = number of times a term  $t$  occurs in document  $d$ .

$$tf(t, d) = N(t, d) / ||D|| \quad (3.13)$$

wherein,  $D$  = Total number of terms in the document

$$df(t) = N(t) \quad (3.14)$$

where  $df(t)$  = Document frequency of a term  $t$  and  $N(t)$  = Number of documents containing the term /

$$idf(t) = N / df(t) = N / N(t) \quad (3.15)$$

$$idf(t) = \log(N / df(t)) \quad (3.16)$$

$$tfidf(t, d) = tf(t, d) * idf(t, d) \quad (3.17)$$

where,  $N$  is the number of recommendable documents and  $n(i)$  is the number

of documents from  $N$  in which the keyword  $i$  appears. We then arrive with recommendations from the content similarity from user search to the available papers in corpus. However, to enhance the recommendations we ranked them using some popularity measures available in the corpus. Citation Score, Influence factor (highly influenced papers score), Twitter mentions data available in the corpus is used with a weightage to provide score to the recommendations in order to rank them. The weightages provided to citation score, influence factor and twitter mentions are 40, 40 and 30 percentage respectively. The weightage is decided based on multiple iterations to obtain best recommendations.

## **CHAPTER 4**

### **IMPLEMENTATION**

#### **4.1 Tools Used**

The following tools, libraries and environments are used in this project

##### **4.1.1 Pandas**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license [\[13\]](#).

##### **4.1.2 Dask**

Dask is an open source library for parallel computing written in Python [\[14\]](#).

##### **4.1.3 NLTK**

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. [\[15\]](#)

##### **4.1.4 Swifter**

Swifter is a package which efficiently applies any function to a pandas dataframe or series in the fastest available manner.

#### **4.1.5 Google Colab Notebooks**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education

#### **4.1.6 Tensorflow**

Tensorflow[16] is an open source library for implementing neural networks. Tensorflow is based on dataflow and differentiable programming

#### **4.1.7 Fasttext**

FastText[7] is a library for learning word embeddings and text classification created by Facebook’s AI Research lab. The model allows to create an unsupervised learning or supervised learning algorithm for obtaining vector representations for words.

### **4.2 Dataset Used**

Arxiv is a repository of 1.7 million articles, with relevant features such as article titles, authors, categories, abstracts, full text PDFs, and more [17]. This dataset is a mirror of the original ArXiv data. Because the full dataset is rather large (1.1TB and growing), this dataset provides only a metadata file in the json format. This file contains an entry for each paper, containing:

- id: ArXiv ID (can be used to access the paper, see below)
- submitter: Who submitted the paper
- authors: Authors of the paper
- title: Title of the paper
- comments: Additional info, such as number of pages and figures
- journal-ref: Information about the journal the paper was published in
- doi: [<https://www.doi.org>](Digital Object Identifier)
- abstract: The abstract of the paper
- categories: Categories / tags in the ArXiv system
- versions: A version history

The dataset has 1704576 instances in total.

## **4.3 Data Wrangling**

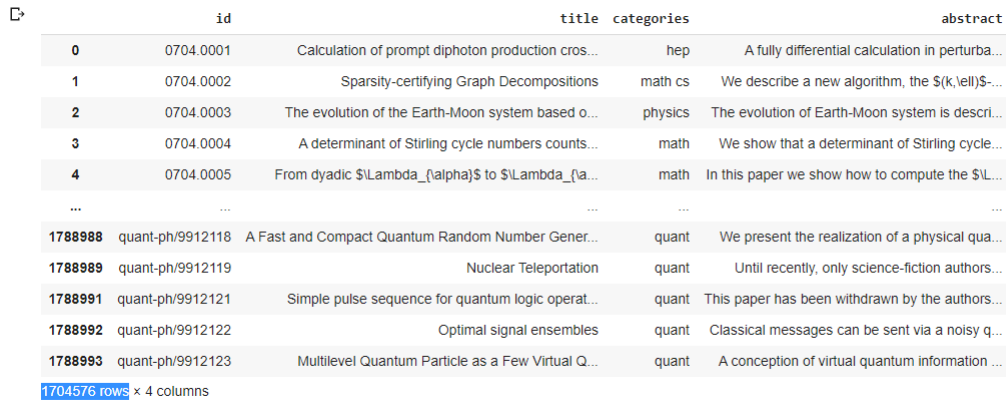
### **4.3.1 Format Conversion**

Data Wrangling is done using dask. Dask is a flexible library for parallel computing in Python. The dataset is acquired in Javascript Object Notation format. Since loading entire JSON object takes a huge amount of RAM dask is used. Dask splits the dataset into various chunks and only loads the necessary chunk into memory. The following fields are chosen and the resultant data is converted into comma separated value file format. This has 1704576 rows.



### 4.3.2 Label Cleaning

In the dataset being used, a row may belong to more than one category. Each category also contains a subcategory which is separated by a dot. These subcategories are removed. After this phase, the dataset has 31 distinct labels.



	id	title	categories	abstract
0	0704.0001	Calculation of prompt diphoton production cros...	hep	A fully differential calculation in perturba...
1	0704.0002	Sparsity-certifying Graph Decompositions	math cs	We describe a new algorithm, the $(k, \ell)$ -...
2	0704.0003	The evolution of the Earth-Moon system based o...	physics	The evolution of Earth-Moon system is descri...
3	0704.0004	A determinant of Stirling cycle numbers counts...	math	We show that a determinant of Stirling cycle...
4	0704.0005	From dyadic $\Lambda_{(\alpha)}$ to $\Lambda_{(\alpha)}$	math	In this paper we show how to compute the $\Lambda_{(\alpha)}$
...	...	...	...	...
1788988	quant-ph/9912118	A Fast and Compact Quantum Random Number Gener...	quant	We present the realization of a physical qua...
1788989	quant-ph/9912119	Nuclear Teleportation	quant	Until recently, only science-fiction authors...
1788991	quant-ph/9912121	Simple pulse sequence for quantum logic operat...	quant	This paper has been withdrawn by the authors...
1788992	quant-ph/9912122	Optimal signal ensembles	quant	Classical messages can be sent via a noisy q...
1788993	quant-ph/9912123	Multilevel Quantum Particle as a Few Virtual Q...	quant	A conception of virtual quantum information ...

1704576 rows x 4 columns

**Figure 4.1: Dataset**  
Dataset in csv format

### 4.3.3 Sampling

Figure 4.1 shows the number of instances each category has. Each instance may belong to more than one category at the same time. The instances of belonging to labels 'acc', 'ao', 'atom', 'bayes', 'plasm' and 'supr' are removed since their occurrence count is very low compared to the other categories which may lead to class imbalance problems. Random sampling is used to sample the dataset. 300000 rows are chosen randomly from the dataset. After removing duplicates from the sample, 293992 rows are used for training and 6008 rows for testing. Since the number of rows are large a split ratio of 98-2 is used.

## 4.4 Data Preprocessing

```

{ 'acc': 49,
  'adap': 584,
  'alg': 1423,
  'ao': 17,
  'astro': 277775,
  'atom': 123,
  'bayes': 16,
  'chao': 2398,
  'chem': 251,
  'cmp': 894,
  'comp': 221,
  'cond': 296203,
  'cs': 302134,
  'dg': 732,
  'econ': 2855,
  'eess': 25993,
  'funct': 427,
  'gr': 84059,
  'hep': 300528,
  'math': 472834,
  'mtrl': 262,
  'nlin': 35203,
  'nucl': 60023,
  'patt': 650,
  'physics': 177360,
  'plasm': 38,
  'q': 45924,
  'quant': 106991,
  'solv': 1413,
  'stat': 79652,
  'supr': 175}

```

**Figure 4.2: Count of categories**

- Removal of digits is done by using regular expression replacement.
- Punctuations are replaced by constructing a lookup table.
- NLTK tokenizer is used to tokenize the data
- NLTK Wordnet Lemmatizer is used to lemmatize the words after tokenization.
- In the tokenized list, all other words except the stopwords are combined to form a string.

↗

	title	categories	abstract	clean_abstract
0	A Health Monitoring System for Elder and Sick ...	cs	This paper discusses a vision based health m...	paper discus vision based health monitoring sy...
1	On stochastic differential equations with arbi...	math	In the recent article [Jentzen, A., M\"uller...	recent article jentzen mullergronbach yaraslav...
2	Why Do Smart Contracts Self-Destruct? Investig...	cs	The Selfdestruct function is provided by Eth...	selfdestruct function provided ethereum smart ...
3	Exchange Cross Sections for Hard Binaries	astro	We present results on the exchange cross sec...	present result exchange cross section interact...
4	Statistical Reconstruction of arbitrary spin s...	quant	A method of quantum tomography of arbitrary ...	method quantum tomography arbitrary spin parti...
...	...	...	...	...
299995	cphVB: A System for Automated Runtime Optimiza...	cs	Modern processor architectures, in addition ...	modern processor architecture addition still c...
299996	Observation of surface solitons in chirped wav...	physics nlin	We report the observation of surface soliton...	report observation surface soliton chirped sem...
299997	A theory of the strain-dependent critical fiel...	cond	We propose a theory to explain the strain de...	propose theory explain strain dependence criti...
299998	Topological Data Analysis in Text Classificati...	stat cs math	While the strength of Topological Data Analy...	strength topological data analysis explored ma...
299999	Recurrences for Eulerian polynomials of type B...	math	We introduce new recurrences for the type B ...	introduce new recurrence type b type eulerian ...

300000 rows × 4 columns

**Figure 4.3: Sampled data**

abstract	clean_abstract
A systematic presentation of spinors in vari...	systematic presentation spinors various dimens...
We systematically calculate the relativistic...	systematically calculate relativistic correcti...
We analyze the solar and the atmospheric neu...	analyze solar atmospheric neutrino problem con...
We study exponentially suppressed contributi...	study exponentially suppressed contribution de...
In order to construct examples for interacti...	order construct example interacting quantum fi...
...	...
In this paper, two cooperative guidance laws...	paper two cooperative guidance law based twopo...
We describe a novel method for removing nois...	describe novel method removing noise wavelet d...
The problem of direction of arrival (DOA) es...	problem direction arrival doa estimation studi...
This paper addresses the problem of online i...	paper address problem online inverse reinforce...
Speech coding forms a crucial element in spe...	speech coding form crucial element speech comm...

**Figure 4.4: Clean Abstract**  
Abstract after preprocessing.

## 4.5 Tokenization and Padding

Tensorflow's tokenizer is used to tokenize the abstract after preprocessing. Once the text is split into individual tokens each word is mapped to a unique number. Thus, each sentence is converted into an array of numbers and is padded with zeroes in the beginning. This is done so that all array of numbers

have the same length.

```
[ [ 0 0 0 ... 128 27 140]
  [3649 6877 3649 ... 2719 3649 449]
  [ 0 0 0 ... 739 1878 7672]
  ...
  [ 0 0 0 ... 195 2948 181]
  [ 307 9 47 ... 501 236 1597]
  [ 0 0 0 ... 191 5129 302]]
(293992, 150)
```

**Figure 4.5: Tokenized data**

## 4.6 Word Embeddings

Each word needs to be converted to its word embedding form for training. Glove[6] embeddings of dimensions 50,100,200 and 300 are obtained from the standford nlp website. All the text files follow the same format. Every line consists of a word followed by it's embedding separated by space. Fasttext[7] library for python is used to obtain word embeddings of the desired dimension. Both glove and fasstext embeddings are converted to the form of a matrix where the row number represents the unique id generated by the tokenizer for a word. Each row contains the word embedding for the particular word. Since glove does not handle out of vocabulary words, the matrix generated using fasttext word embeddings is used for further study.

The number of rows in the matrix refers to the total number of unique words in the dataset while each row is of dimension (1,300) where 300 is the dimension of the word embedding.

## 4.7 Multi label text classification using deep learning

```

Out of vocabulary words: 201249
array([[ 0.,          0.,          0.,          ...,  0.,          ,
         0.,          0.,          ],
       [-0.27663001,  0.55093998,  0.13618 ,          ..., -0.63821 ,
         1.02839994,  0.053952 ],
       [-0.087511 ,  0.14542 ,  0.29126999,          ..., -0.095098 ,
         0.36195999, -0.53465003],
       ...,
       [ 0.,          0.,          0.,          ...,  0.,          ,
         0.,          0.,          ],
       [ 0.34474 , -0.11959 ,  0.41108 ,          ...,  0.25872999,
        -0.1656 ,  0.2414 ],
       [ 0.,          0.,          0.,          ...,  0.,          ,
         0.,          0.,          ]])

```

**Figure 4.6: Matrix generated using Glove Embeddings**  
Glove does not handle out of vocabulary words.

The length of the sentence is limited to 150 words to avoid memory overflow issues after observing the distribution of sentence lengths.

In all the LSTM based models, a LSTM cell of size 128 is used. The final LSTM cell is connected to a fully connected layer of 256 neurons which is connected to 25 neurons at the end. Inverted Dropout regularization is applied with a rate of 0.2. Adam optimizer is used to optimize the cost.

#### 4.7.1 LSTM

Then input layer is connected to the embedding layer which takes the number of dimensions, vocabulary size and the embedding matrix as parameters. The 2 dimensional input of (batch size, 150) is converted into a three dimensional matrix of size (batch size, 150, embedding dimension). LSTM cells take word embedding as an input at every time step and hidden states from previous steps are passed along. The final LSTM cell is connected to a dense layer of neurons which is connected to another dense layer of 25 neurons after applying dropout regularization.

---

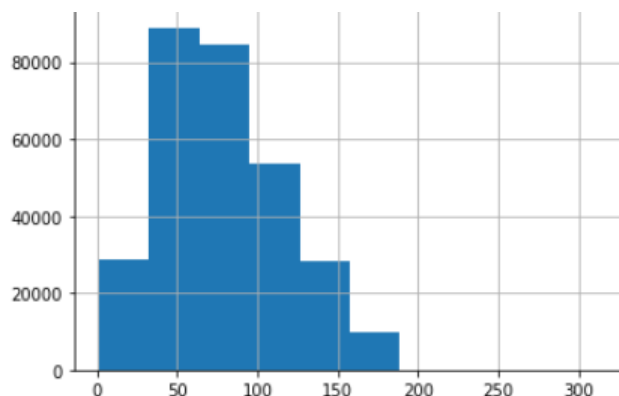
```

[[ 0.          0.          0.          ...  0.          0.
  0.          ]
 [-0.0273156  -0.01344623  0.01222771 ...  0.07275879 -0.00511334
 -0.02643684]
 [-0.06271255  0.01664297  0.04767837 ...  0.05345504 -0.0081516
  0.06989843]
 ...
 [-0.00320528  0.02285784  0.0044757 ...  0.02222969 -0.01331795
 -0.00324045]
 [ 0.02548821 -0.05357264 -0.00069272 ... -0.00375715 -0.0320634
  0.0346636 ]
 [-0.03807143  0.03105393  0.00046104 ...  0.00831371 -0.00895843
  0.00434265]]
(416152, 300)

```

---

**Figure 4.7: Matrix generated using Fasttext Embeddings**  
Fasttext handles out of vocabulary words using character level n grams.



**Figure 4.8: Distribution of sentence lengths**  
A histogram showing distribution of sentence lengths.

#### 4.7.2 Bidirectional LSTM

The output from forward LSTM and backward LSTM is concatenated. This results in a matrix of dimension (batch size, 150, lstm size) since an output vector of dimension (lstm size) is produced at each time step. Global Average Pooling is applied to reduce the dimensions of the matrix to 2. This calculates the average of all the vectors along axis=1.

Layer (type)	Output Shape	Param #
embedding_44 (Embedding)	(None, None, 300)	124845600
lstm_44 (LSTM)	(None, 128)	219648
dense_88 (Dense)	(None, 256)	33024
dropout_44 (Dropout)	(None, 256)	0
dense_89 (Dense)	(None, 25)	6425
Total params: 125,104,697		
Trainable params: 125,104,697		
Non-trainable params: 0		

**Figure 4.9: Summary of the LSTM model**

The dense layer connected to the final LSTM cell contains 256 neurons. A inverted dropout regularization with rate = 0.2 is applied on it.

Layer (type)	Output Shape	Param #
input_18 (InputLayer)	[(None, 150)]	0
embedding_63 (Embedding)	(None, 150, 300)	124845600
bidirectional_11 (Bidirectio	(None, 150, 512)	1140736
global_average_pooling1d_4 (	(None, 512)	0
dense_117 (Dense)	(None, 256)	131328
dropout_82 (Dropout)	(None, 256)	0
dense_118 (Dense)	(None, 25)	6425
Total params: 126,124,089		
Trainable params: 126,124,089		
Non-trainable params: 0		

**Figure 4.10: Summary of the Bi-LSTM model**

Each LSTM cell is of size 512 which specifies the number of neurons inside the LSTM cell.

### 4.7.3 Bidirectional LSTM with self attention

The output from the BiLSTM layer is given as input to the attention layer which calculates the context vector. The context vector is connected to two dense layers subsequently.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 150)]	0	
embedding_45 (Embedding)	(None, 150, 300)	124845000	input_1[0][0]
bidirectional (Bidirectional)	(None, 150, 512)	1140736	embedding_45[0][0]
attention (Attention)	(None, 150, 512)	0	bidirectional[0][0] bidirectional[0][0]
global_average_pooling1d (GlobalAveragePooling1D)	(None, 512)	0	attention[0][0]
dense_90 (Dense)	(None, 256)	131328	global_average_pooling1d[0][0]
dropout_45 (Dropout)	(None, 256)	0	dense_90[0][0]
dense_91 (Dense)	(None, 25)	6425	dropout_45[0][0]
Total params: 126,124,089			
Trainable params: 126,124,089			

**Figure 4.11: Summary of the Bi-LSTM Attention model**  
Global average pooling is applied on the attention scores to reduce the dimension.

#### 4.7.4 TextCNN

In a TextCNN model, convolutional filters of different sizes are applied in a parallel manner on top of word embedding layers. The filters are one dimensional and of dimension (filter size, embedding dimension). Maxpooling operation is performed after applying convolutional filters and the results are concatenated together. The concatenated result is flattened to reduce the dimension and is connected to dense layers at the end. Best results were obtained with 32 filters of sizes 2,4,6 and 8 each.

#### 4.7.5 CNN-LSTM

Convolutional filters of increasing width are applied sequentially over the embedding layer to extract features. Maxpooling is applied after each convolution is performed. The resultant embedding vector is given as input to a Bidirectional LSTM in multiple time steps which is connected to dense layers at the end. The output from the BiLSTM is flattened before being connected to the dense layers. Best results were obtained with filters of sizes 2,4 and 6 and number of filters 32,64 and 128.



Layer (type)	Output Shape	Param #	Connected to
input_6 (InputLayer)	[(None, 150)]	0	
embedding_8 (Embedding)	(None, 150, 300)	124845600	input_6[0][0]
reshape_2 (Reshape)	(None, 150, 300, 1)	0	embedding_8[0][0]
conv2d_8 (Conv2D)	(None, 149, 1, 32)	19232	reshape_2[0][0]
conv2d_9 (Conv2D)	(None, 147, 1, 32)	38432	reshape_2[0][0]
conv2d_10 (Conv2D)	(None, 145, 1, 32)	57632	reshape_2[0][0]
conv2d_11 (Conv2D)	(None, 143, 1, 32)	76832	reshape_2[0][0]
max_pooling2d_8 (MaxPooling2D)	(None, 1, 1, 32)	0	conv2d_8[0][0]
max_pooling2d_9 (MaxPooling2D)	(None, 1, 1, 32)	0	conv2d_9[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 1, 1, 32)	0	conv2d_10[0][0]
max_pooling2d_11 (MaxPooling2D)	(None, 1, 1, 32)	0	conv2d_11[0][0]
concatenate_5 (Concatenate)	(None, 4, 1, 32)	0	max_pooling2d_8[0][0] max_pooling2d_9[0][0] max_pooling2d_10[0][0] max_pooling2d_11[0][0]
flatten_4 (Flatten)	(None, 128)	0	concatenate_5[0][0]
dropout_15 (Dropout)	(None, 128)	0	flatten_4[0][0]
dense_9 (Dense)	(None, 25)	3225	dropout_15[0][0]
Total params: 125,040,953			
Trainable params: 125,040,953			
Non-trainable params: 0			

**Figure 4.12: Summary of the TextCNN model**

Global average pooling is applied on the attention scores to reduce the dimension.

#### 4.7.6 LSTM-CNN

Convolutional filters of increasing width are applied in a parallel manner to the output of the BiLSTM layer in contrast to CNN-LSTM where filters were applied sequentially on the embedding layer. Average pooling and maxpooling are also applied parallelly after convolution. The vectors are concatenated and global average pooling is applied to reduce the dimensions. Batch normalization is applied to normalize the result and is connected to dense layers in the end. Since the filters are applied parallelly on a 3 dimensional vector. The number of filters have to be of the same size across convolutions to avoid dimensionality related conflicts during concatenation. Best results were obtained with filters of sizes 2,4 and 6 and number of filters 32 each.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 150, 300)	124845600
conv1d_3 (Conv1D)	(None, 150, 32)	19232
max_pooling1d_3 (MaxPooling1D)	(None, 50, 32)	0
dropout_5 (Dropout)	(None, 50, 32)	0
conv1d_4 (Conv1D)	(None, 50, 64)	8256
max_pooling1d_4 (MaxPooling1D)	(None, 16, 64)	0
dropout_6 (Dropout)	(None, 16, 64)	0
conv1d_5 (Conv1D)	(None, 16, 128)	49280
max_pooling1d_5 (MaxPooling1D)	(None, 5, 128)	0
dropout_7 (Dropout)	(None, 5, 128)	0
bidirectional_1 (Bidirectional)	(None, 5, 1024)	2625536
dropout_8 (Dropout)	(None, 5, 1024)	0
flatten_1 (Flatten)	(None, 5120)	0
dense_2 (Dense)	(None, 256)	1310976
dropout_9 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 25)	6425
Total params: 128,865,305		
Trainable params: 128,865,305		
Non-trainable params: 0		

**Figure 4.13: Summary of the CNN-LSTM model**  
Convolutional filters are applied sequentially on the embedding layer.

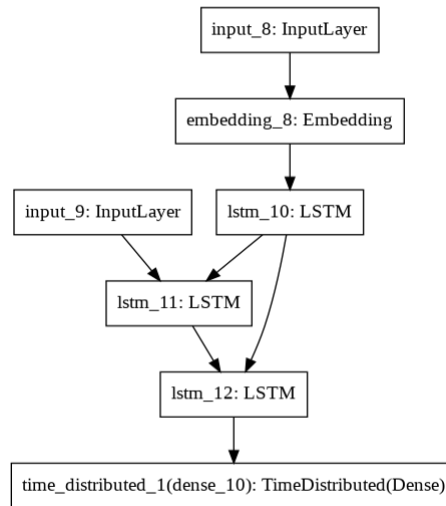
#### 4.7.7 Encoder Decoder with teacher forcing

The encoder resembles the usual LSTM model used for text classification with the exception of the dense layers. The hidden state of the last LSTM cell from the encoder is passed as the decoder's initial state. While the encoder takes word embeddings as input, the decoder takes the one hot encoded labels as the input. The sequence of labels is just a sequence of 1's and 0's. In the decoder, the sequence of labels are predicted one step at a time and the predicted labels at every time step are passed as the input to the next. To reduce training time and train the model more efficiently teacher forcing is used. In teacher forcing, during training time the original ground truth shifted right by 1 step is passed as input at each time step rather than the label predicted in the previous time step. This reduces training time and the weights are updated a lot faster.

Model: "model\_2"

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 150)]	0	
embedding_5 (Embedding)	(None, 150, 300)	124845600	input_3[0][0]
spatial_dropout1d_2 (SpatialDro	(None, 150, 300)	0	embedding_5[0][0]
bidirectional_4 (Bidirectional)	(None, 150, 1024)	3330048	spatial_dropout1d_2[0][0]
conv1d_12 (Conv1D)	(None, 149, 32)	65568	bidirectional_4[0][0]
conv1d_13 (Conv1D)	(None, 147, 32)	131104	bidirectional_4[0][0]
conv1d_14 (Conv1D)	(None, 145, 32)	196640	bidirectional_4[0][0]
max_pooling1d_12 (MaxPooling1D)	(None, 74, 32)	0	conv1d_12[0][0]
max_pooling1d_13 (MaxPooling1D)	(None, 73, 32)	0	conv1d_13[0][0]
max_pooling1d_14 (MaxPooling1D)	(None, 72, 32)	0	conv1d_14[0][0]
concatenate_2 (Concatenate)	(None, 219, 32)	0	max_pooling1d_12[0][0] max_pooling1d_13[0][0] max_pooling1d_14[0][0]
global_average_pooling1d_2 (Glo	(None, 32)	0	concatenate_2[0][0]
batch_normalization_2 (BatchNor	(None, 32)	128	global_average_pooling1d_2[0][0]
dropout_12 (Dropout)	(None, 32)	0	batch_normalization_2[0][0]
dense_6 (Dense)	(None, 25)	825	dropout_12[0][0]
Total params: 128,569,913			
Trainable params: 128,569,849			
Non-trainable params: 64			

**Figure 4.14: Summary of the LSTM-CNN model**  
Convolutional filters are applied parallelly on the output of BiLSTM layer.



**Figure 4.15: Flow diagram of the Encoder decoder model**  
Hidden state from the encoder is passed to the decoder.

### 4.7.8 BERT

The pretrained BERT model with 12 encoder stacks and 12 attention heads is obtained from tensorflow hub. AdamW which is a weighted version of the adam optimizer is used with a decaying learning rate. The preprocessing is done using BERT tokenizer which is a subword tokenizer. Subword tokenizers handle out of vocabulary words more efficiently than fasttext. Out of Vocabulary words are split into multiple subwords and considered as separate words before being converted to embeddings. BERT outputs a dictionary of three different items - Sentence embeddings for each sentence in the batch, Contextual word embeddings for every word of every sentence in the batch, encoder outputs of the transformer blocks. The encoder output of the transformer blocks is connected to a dense layer after applying dropout.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None,)]	0	
keras_layer_2 (KerasLayer)	{'input_type_ids': ( 0		input_1[0][0]
keras_layer_3 (KerasLayer)	{'encoder_outputs': 109482241		keras_layer_2[0][0] keras_layer_2[0][1] keras_layer_2[0][2]
dropout (Dropout)	(None, 768)	0	keras_layer_3[0][13]
dense (Dense)	(None, 25)	19225	dropout[0][0]
Total params: 109,501,466			
Trainable params: 109,501,465			
Non-trainable params: 1			

**Figure 4.16: Flow diagram of BERT**  
Transfer learning is done using pretrained BERT with weights.

## 4.8 Recommendation System

### 4.8.1 Dataset

The dataset that is used to build the recommendation system was scrapped from Semantic Scholar Website as it provides variety of data compared

to conventional datasets that provide only Title, Abstract and citations. The columns that are present in the dataset are as follows

**Table 4.1: Example 1**

<i>ColumnName</i>	<i>Information</i>
Title of the Article	Title of the research article is provided
Authors	The names of the authors is provided
Published Journal	The name of the journal under which the article is published is provided.
Year of Publication	Year in which the article is released is provided
Abstract of the article	This section contains a small summary about what is present in the article
Number of Citations	The number of times the given article is cited is provided
Highly Influenced papers	The count of research papers which uses the given research article as its main source of foundation
Cite Background	The count of research papers in which the background work of the given research article is cited
Cite Methods	The count of research papers in which the method employed by the given research article is cited
Cite Results	The count of research papers in which the final results of the given research article is cited
Twitter Mentions	The number of times the given article is tweeted on twitter
Citation Titles	Contains the name of articles that cites that particular paper
Citation Journals	Contains the journals in which the citation titles are published
Citation Years	The year in which the citation titles were published
Reference Titles	Contains the name of articles to which the selected paper refers to
Reference Journals	Contains the journals in which the reference titles are published
Reference Journals	Contains the journals in which the reference titles are published
Reference Years	The year in which the reference titles were published

The dataset contains 3000 rows of Research Papers that belong to the

four domains namely, Cloud Computing, Block Chain, Machine Learning and Internet of Things.

#### 4.8.2 Content Based Recommendation

Content based filtering relies upon TF-IDF scores and popularity measures that are available in the dataset. Two new columns are added to the dataset namely, Title and Abstract keywords respectively. This is achieved by removing all the stopwords from title and abstract columns of the dataset. The TF-IDF scores of the keywords are then calculated and stored in a matrix. The user input is cleaned of stopwords, and the TF-IDF score of the user keywords is calculated. The top 10 recommendable list is then obtained by comparing the TF-IDF matrix to which the user keyword is matched with. The top 10 scores are retrieved along with their indexes in the dataset and then rankings are calculated by the popularity measures. There are 3 popularity measures, Twitter Mentions, Highly Influenced Papers and Number of Citations.

```
In [113]: #sample execution
search = "tensorflow"
recommendForSearch(search)

{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 89.76770760218147, 6: 95.61746177771008, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0, 11: 0.0, 12: 0.0, 13: 0.0, 14: 0.0, 15: 0.0, 16: 0.0, 17: 0.0, 18: 0.0, 19: 0.0, 20: 0.0, 21: 0.0, 22: 0.0, 23: 0.0, 24: 0.0, 25: 0.0, 26: 0.0, 27: 0.0, 28: 0.0, 29: 0.0, 30: 0.0, 31: 0.0, 32: 0.0, 33: 0.0, 34: 0.0, 35: 0.0, 36: 0.0, 37: 0.0, 38: 0.0, 39: 0.0, 40: 0.0, 41: 0.0, 42: 0.0, 43: 0.0, 44: 0.0, 45: 0.0, 46: 0.0, 47: 0.0, 48: 0.0, 49: 0.0, 50: 0.0, 51: 0.0, 52: 0.0, 53: 0.0, 54: 0.0, 55: 0.0, 56: 0.0, 57: 0.0, 58: 0.0, 59: 0.0, 60: 0.0, 61: 0.0, 62: 0.0, 63: 0.0, 64: 0.0, 65: 0.0, 66: 0.0, 67: 0.0, 68: 0.0, 69: 0.0, 70: 0.0, 71: 0.0, 72: 0.0, 73: 0.0, 74: 0.0, 75: 0.0, 76: 0.0, 77: 0.0, 78: 0.0, 79: 0.0, 80: 0.0, 81: 0.0, 82: 0.0, 83: 0.0, 84: 0.0, 85: 0.0, 86: 0.0, 87: 0.0, 88: 0.0, 89: 0.0, 90: 0.0, 91: 0.0, 92: 0.0, 93: 0.0, 94: 0.0, 95: 0.0, 96: 0.0, 97: 0.0, 98: 0.0, 99: 0.0, 100: 0.0, 101: 0.0, 102: 0.0, 103: 0.0, 104: 0.0, 105: 0.0, 106: 0.0, 107: 0.0, 108: 0.0, 109: 0.0, 110: 0.0, 111: 0.0, 112: 0.0, 113: 0.0, 114: 0.0, 115: 0.0, 116: 0.0, 117: 0.0, 118: 0.0, 119: 0.0, 120: 0.0, 121: 0.0, 122: 0.0, 123: 0.0, 124: 0.0, 125: 0.0, 126: 0.0, 127: 0.0, 128: 0.0, 129: 0.0, 130: 0.0, 131: 0.0, 132: 0.0, 133: 0.0, 134: 0.0, 135: 0.0, 136: 0.0, 137: 0.0, 138: 0.0, 139: 0.0, 140: 0.0, 141: 0.0, 142: 0.0, 143: 0.0, 144: 0.0, 145: 0.0, 146: 0.0, 147: 0.0, 148: 0.0, 149: 0.0, 150: 0.0, 151: 0.0, 152: 0.0, 153: 0.0, 154: 0.0, 155: 0.0, 156: 0.0, 157: 0.0, 158: 0.0, 159: 0.0, 160: 0.0, 161: 0.0, 162: 0.0, 163: 0.0, 164: 0.0, 165: 0.0, 166: 0.0, 167: 0.0, 168: 0.0, 169: 0.0, 170: 0.0, 171: 0.0, 172: 0.0, 173: 0.0, 174: 0.0, 175: 0.0, 176: 0.0, 177: 0.0, 178: 0.0, 179: 0.0, 180: 0.0, 181: 0.0, 182: 0.0, 183: 0.0, 184: 0.0, 185: 0.0, 186: 0.0, 187: 0.0, 188: 0.0, 189: 0.0, 190: 0.0, 191: 0.0, 192: 0.0, 193: 0.0, 194: 0.0, 195: 0.0, 196: 0.0, 197: 0.0, 198: 0.0, 199: 0.0, 200: 0.0, 201: 0.0, 202: 0.0, 203: 0.0, 204: 0.0, 205: 0.0, 206: 0.0, 207: 0.0, 208: 0.0, 209: 0.0, 210: 0.0, 211: 0.0, 212: 0.0, 213: 0.0, 214: 0.0, 215: 0.0, 216: 0.0, 217: 0.0, 218: 0.0, 219: 0.0, 220: 0.0, 221: 0.0, 222: 0.0, 223: 0.0, 224: 0.0, 225: 0.0, 226: 0.0, 227: 0.0, 228: 0.0, 229: 0.0, 230: 0.0, 231: 0.0, 232: 0.0, 233: 0.0, 234: 0.0, 235: 0.0, 236: 0.0, 237: 0.0, 238: 0.0, 239: 0.0, 240: 0.0, 241: 0.0, 242: 0.0, 243: 0.0, 244: 0.0, 245: 0.0, 246: 0.0, 247: 0.0, 248: 0.0, 249: 0.0, 250: 0.0, 251: 0.0, 252: 0.0}
```

**Figure 4.17: TF-IDF Scores of TensorFlow**

```
In [116]: ▶ #sample execution
search = "tensorflow"
recommendForSearch(search)
```

	Final Score
6	15.304985
5	24.440584
0	5.174045
1	3.296236
2	3.552090
3	2.595154
4	2.940717
7	2.904844
8	3.855071
9	5.215119

**Figure 4.18: Popularity Scores of Tensorflow**

The top ten matches based on highest TF-IDF scores and their popularity scores.

### 4.8.3 Collaborative Based Recommendation

This system is implemented using citation analysis unlike content based recommendation. The references of the user paper is matched with every documents references in the corpus. This set of potential recommendable papers is retrieved upon which the second condition is applied. We find papers that have cited both the user paper and any of the papers available in the potential recommendable list. In other words, more papers that have cited them together the better close they are.

## CHAPTER 5

### RESULTS

#### 5.1 Results for multi label text classification

The models are trained and validated with a validation split of 0.02 against each epoch. Two metrics are used to evaluate the model. Dropout regularization with rate-0.2 is used to avoid overfitting.

##### 5.1.1 Metrics for evaluating multi label classification

###### 5.1.1.1 Hamming Loss

Hamming loss is the fraction of labels that are incorrectly predicted for a single sample. To find the total hamming loss, individual hamming loss scores across each sample is added up and the average is taken. If  $\hat{Y}$  is the predicted value for the j-th label of a given sample and y is the corresponding true value, and n is the number of classes or labels, then the Hamming loss is defined as:

$$Hamming\ Loss = 1/n \sum_{j=0}^{n-1} 1(\hat{Y} \neq Y) \quad (5.1)$$

where  $1(x)$  is the indicator function.



### 5.1.1.2 Subset Accuracy

In multilabel classification, subset accuracy computes the percentage of the set of labels predicted for a sample exactly matching the corresponding set of labels.

model	embedding	dimension	batch_size	epochs	val_subset_accuracy	val_hamming_loss	test_subset_accuracy	test_hamming_loss
LSTM	fasttext	300	1024	5	69.02	0.017	69.9	0.017
Bi-LSTM	fasttext	300	1024	5	69.37	0.012	70.22	0.017
Bi-LSTM attention	fasttext	300	1024		69.8	0.017	70.12	0.016
TextCNN	fasttext	300	1024	5	68.54	0.017	59.3	0.05
CNN-BiLSTM	fasttext	300	1024	7	69.21	0.017	69.9	0.017
BiLSTM-CNN	fasttext	300	1024	7	68.59	0.018	68.7	0.018
Seq2Seq	fasttext	300	1024	5	68.5	0.018	69.7	0.018
<b>BERT uncased 12 encoder stack</b>	<b>sentence + positional + word</b>	<b>variable</b>	<b>64</b>	<b>4</b>	<b>71.5</b>	<b>0.018</b>	<b>72.8</b>	<b>0.015</b>

**Figure 5.1: Results**

Test and validation subset accuracy and hamming loss for each model.

### 5.1.1.3 Inference

The best performance was achieved with BERT which produced a hamming loss 0.015 and subset accuracy of 72.8 percentage on the test dataset.

Although the subset accuracy appears low the model is performing really well since it is able to predict 99 percentage of the labels across every row of the test dataset. In 72.8 percentage of the rows, there was an exact match for all the 25 labels. The reason for BERT's performance can be attributed to the multi head attention and the way of handling out of vocabulary words through subword tokenization. Since the dataset contains a lot of scientific words, they may not be present in any pretrained embedding vocabulary. BERT handles this effectively through subword tokenization.

## 5.2 Results for recommendation system

Two filtering methods were built for recommendation of research and the following results were obtained

```
filename = 'final.csv'
findMoreSimilar(13, filename)
```

Out[3]: ['TensorFlow: A System for Large-Scale Machine Learning',  
'MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems',  
'cuDNN: Efficient Primitives for Deep Learning',  
'Learning Deep Structured Models',  
'Learning Fine-Grained Image Similarity with Deep Ranking',  
'Deep Learning-Based Classification of Hyperspectral Data',  
'Neural Architecture Search with Reinforcement Learning',  
'Deep learning in bioinformatics',  
'Learning Transferable Features with Deep Adaptation Networks',  
'Building Machines That Learn and Think Like People',  
'Cross-Stitch Networks for Multi-task Learning',  
'Deep Residual Learning for Image Recognition',  
'Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering',  
'Federated Optimization: Distributed Machine Learning for On-Device Intelligence',  
'Project Adam: Building an Efficient and Scalable Deep Learning Training System',  
'Deep multiple instance learning for image classification and auto-annotation',  
'Learning visual similarity for product design with convolutional neural networks',  
'Learning a Predictable and Generative Vector Representation for Objects']

**Figure 5.2: Collaborative Based Recommendation System**

```

In [4]: search = "deep learning"
        recommendForSearch(search)

Out[4]: 12          Deep Learning
        175      Deep Reinforcement Learning with Double Q-lear...
        232      Deep Bayesian Active Learning with Image Data
        106          Deep learning in bioinformatics
        352      A Brief Survey of Deep Reinforcement Learning
        133          Generalization in Deep Learning
        344      Deep learning in agriculture: A survey
        831      Learning IoT in Edge: Deep Learning for the In...
        230      Deep learning for sentiment analysis: A survey
        164      Applications of Deep Learning and Reinforcemen...
        Name: Title of the Article, dtype: object

```

**Figure 5.3: Content Based Recommendation System**

## REFERENCES

- [1] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. pages 207–212, 2016.
- [2] Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 08 2014.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [4] R. Sharma, D. Gopalani, and Y. Meena. Collaborative filtering-based recommender system: Approaches and research challenges. pages 1–6, 2017.
- [5] M. V. Murali, T. G. Vishnu, and N. Victor. A collaborative filtering based recommender system for suggesting new trends in any domain of research. pages 550–553, 2019.
- [6] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014.
- [7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.
- [8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [9] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [10] J. Xie, B. Chen, X. Gu, F. Liang, and X. Xu. Self-attention-based bilstm model for short text fine-grained sentiment classification. *IEEE Access*, 7:180558–180570, 2019.
- [11] P. Szymański and T. Kajdanowicz. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*, February 2017.
- [12] Jinseok Nam, Eneldo Mencia, Hyunwoo Kim, and Johannes Furnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. 12 2017.

- [13] The pandas development team. pandas-dev/pandas: Pandas, February 2020.
- [14] Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016.
- [15] Edward Loper Bird, Steven and Ewan Klein. Natural Language Processing with Python. *O'Reilly Media Inc*, 2009.
- [16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [17] Colin B. Clement, Matthew Bierbaum, Kevin P. O’Keeffe, and Alexander Amir Alemi. On the use of arxiv as a dataset. *ArXiv*, abs/1905.00075, 2019.
- [18] Gil Joon-Min Kim, Sang-Woon. Research paper classification systems based on tf-idf and lda schemes. *Human-centric Computing and Information Sciences*, 9:30, 2019.
- [19] Iz Beltagy, Kyle Lo, and Arman Cohan. scibert: A pretrained language model for scientific text. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, 2019.
- [20] Pedram Amoli and Omid Sh. Scientific documents clustering based on text summarization. *International Journal of Electrical and Computer Engineering*, 5:782–787, 08 2015.
- [21] Grigorios Tsoumakas and I. Vlahavas. Random k -labelsets: An ensemble method for multilabel classification. 4701:406–417, 08 2007.
- [22] Newton Spolaôr, Everton Cherman, Maria-Carolina Monard, and Huei Lee. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151, 03 2013.

- [23] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85:254–269, 08 2009.
- [24] Zhi-Hua Zhou Min-Ling Zhang. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038 – 2048, 2007.
- [25] Philipp Probst, Quay Au, Giuseppe Casalicchio, Clemens Stachl, and Bernd Bischl. Multilabel classification with r package mlr. 2017.
- [26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [27] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, pages 217–226, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [28] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 254–269, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [29] Gang Kou, Pei Yang, Yi Peng, Feng Xiao, Yang Chen, and Fawaz E. Alsaadi. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, 86:105836, 2020.
- [30] Ricardo Sousa and João Gama. Multi-label classification from high-speed data streams with adaptive model rules and random rules. *Progress in Artificial Intelligence*, 7, 01 2018.
- [31] Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. Magnet: Multi-label text classification using attention-based graph neural network. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, 2020.
- [32] Passent Elkafrawy, Amr Mausad, and Heba Esmail. Experimental comparison of methods for multi-label classification in different application domains. *International Journal of Computer Applications*, 114:1–9, 03 2015.
- [33] Yoon Kim. Convolutional neural networks for sentence classification, 2014.