B. TECH. PROJECT REPORT

On

Design and Development of Extended Recurrent Convolutional Neural Network (ERCNN) for Multi-label Text Classification

BY

Digvijay Singh

130001011

Jaspreet Singh Saluja

130001018

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2016

Design and Development of Extended Recurrent Convolutional Neural Network (ERCNN) for Multi-label Text Classification

A PROJECT REPORT

Submitted in partial fulfillment of the

requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Digvijay Singh **Jaspreet Singh Saluja**

Guided by:

Dr. Aruna Tiwari

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2016

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Design and development of Extended Recurrent Convolutional Neural Network (ERCNN) for Multi-label Text Classification"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science Engineering' completed under the supervision of **Dr. Aruna Tiwari, Assistant Professor,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Signature and name of the student(s) with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

Signature of BTP Guide(s) with dates and their designation

Preface

This report on "Design and development of Extended Recurrent Convolutional Neural Network for Multi-label Text Classification" is prepared under the guidance of **Dr. Aruna Tiwari**.

Through this report we have tried to implement a new approach for multi-label text classification. We have researched the existing approaches analysed their shortcomings. The algorithm which we have proposed is a novel work in the text classification research area and can find many real life applications.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added figures to make it more illustrative.

Digvijay Singh

Jaspreet Singh Saluja

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgements

We wish to thank Dr. Aruna Tiwari for her kind support and valuable guidance throughout the duration of the project.

It is her help and support, due to which we became able to complete the design and technical report.

Without her support this report would not have been possible.

Digvijay Singh

Jaspreet Singh Saluja

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Abstract

While deep Recurrent Convolutional Neural Networks (RCNN) have shown a great success in single-label/binary text classification, it is important to note that real world text generally contain multiple labels, which could correspond to different objects, scenes, actions and attributes in an image. Traditional approaches to multi-label text classification learn independent classifiers for each category and employ ranking or thresholding on the classification results. These techniques performance results are depriving. Moreover, they fail to explicitly exploit the label dependencies in a text.

The approaches developed over the years can be divided into two broad categories, i) Bag of Words, and ii) Word Embedding. The Bag of words approach doesn't take into account the sentiments of the words due to which it cannot give accurate results in some situations. SVM, back-propagation neural network and random forest are some of the approaches using the bag of words. We have implemented these for comparison with our approach. Word embedding maps words of the vocabulary to vectors of real numbers. It is better than the bag of words approach as it helps capture the semantic of the words. Convolutional neural network and recurrent neural network use word embeddings of the words as input in the input layer. Both of them have been implemented [16][17] and have shown better results than BoW approaches.

In this paper, we have proposed Extended Recurrent Convolutional Neural Networks(ERCNN) for multi-label text classification. RCNN is a combination of RNN and CNN which uses a bidirectional recurrent structure to avoid giving preference to the words which appear later in the text. The max pooling layer of convolutional neural network automatically judges features which play a key role in text classification. It is evident from the results that RCNN produces better results than the Bag of Words(BoW) approaches as well as RNN and CNN which use word embedding. We have extended the RCNN for multi-label problem and we will call it Extended RCNN in this paper. The complexity of ERCNN is O(n) which is same as RNN and CNN as they are applied sequentially.

Contents

List of Figures

1

Introduction

Natural Language Processing is an emerging field and with so much data produced each day on the internet, it is very important to design novel algorithms which are faster and more accurate. In this paper, we are proposing an algorithm for multi-label text classification. This section will brief you on the topics of text classification and multi-label classification.

1.1 Text Classification

Text Classification is a semi-supervised machine learning task that automatically assigns a given document to a set of predefined categories/labels based on its textual content and extracted features. Automatic Text Classification has important applications in content management, contextual search, opinion mining, product review analysis, spam filtering and text sentiment mining.

Documents may be classified according to their subjects or according to other attributes (such as document type, author, printing year etc.). In the rest of this article only subject classification is

considered. There are two main philosophies of subject classification of documents: the content-based approach and the request-based approach.

1.1.1 Applications

Text Classification has many applications now-a-days because of the huge amount of data we deal with everyday. Some of the major domains where text classification is used are listed below:

- Spam Filtering: Detecting if an email is legitimate or not.
- Language identification: Automatically determining the language of a text
- Genre classification: Automatically determining the genre of a text
- Readability assessment": automatically determining the degree of readability of a text.
- Sentiment analysis: determining the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

1.1.2 Process

- Documents Collection: This is first step of classification process in which we are collecting the different types of documents to train the model we will use for the classification process.
- Preprocessing: The data collected has to be pre-processed in the next steps to remove unwanted data or noise. Some of the steps taken are:
    - Tokenization: A document is treated as a string, and then partitioned into a list of tokens.
    - Removing stop words: Stop words such as "the", "a", "and", etc. are frequently occurring, so the insignificant words need to be removed.
    - Stemming word: Applying the stemming algorithm that converts different word form into similar canonical form.
- Indexing: The documents representation is one of the pre-processing technique that is used to reduce the complexity of the documents and make them easier to handle. One way to do it is to represent the words in the document in the form of vectors.

- Feature Selection: After pre-processing and indexing the important step of text classification, is feature selection to construct vector space, which improves the scalability, efficiency and accuracy of a text classifier. The main idea of Feature Selection (FS) is to select subset of features from the original documents. FS is performed by keeping the words with highest score according to predetermined measure of the importance of the word.
- Classification: The automatic classification of documents into predefined categories has observed as an active attention, the documents can be classified by three ways, unsupervised, supervised and semi-supervised methods. From last few years, the task of automatic text classification have been extensively studied and rapid progress seems in this area, including the machine learning approaches such as Bayesian classifier, Decision Tree, K-nearest neighbor(KNN), Support Vector Machines(SVMs), Neural Networks, Rocchio's.
- Performance Evaluations: This is Last stage of Text classification, in which the evaluations of text classifiers is typically conducted experimentally, rather than analytically. An important issue of Text categorization is how to measures the performance of the classifiers.

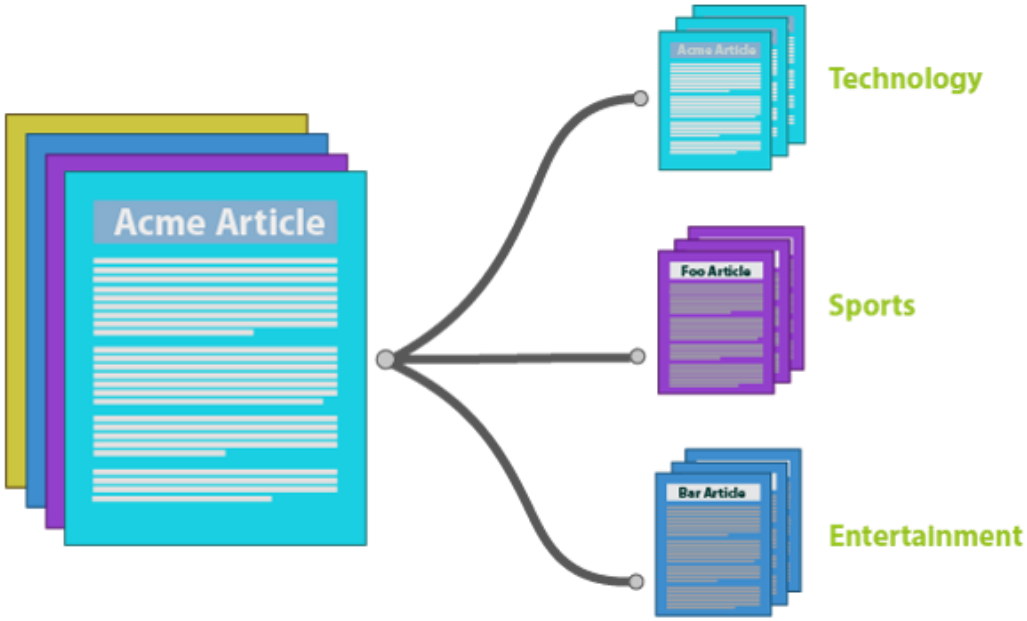


Figure 1. Text Classification based on labels

1.2 Multi-label Classification

1.2.1 Overview

Multi-label classification is a classification problem where multiple target labels must be assigned to each instance. Multi-label classification should not be confused with multi-class classification, which is the problem of categorizing instances into one of more than two classes. Formally, multi-label learning can be phrased as the problem of finding a model that maps inputs x to binary vectors y, rather than scalar outputs as in the ordinary classification problem.

1.2.2 Methods

There are two main methods for tackling the multi-label classification problem: problem transformation methods and algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification problems, which can then be

handled using single-class classifiers. Algorithm adaptation methods adapt the algorithms to directly perform multi-label classification. In other words, rather than trying to convert the problem to a simpler problem, they try to address the problem in its full form.

The multi-label context is used in many domains, such as text categorisation and labelling images but the main challenge encountered while implementing it is modelling dependencies between labels, which must be done efficiently to scale up to settings involving large datasets and data streams.

To model label dependency, most existing works are based on graphical models, among which a common approach is to model the co-occurrence dependencies with pairwise compatibility probabilities or co-occurrence probabilities and use Markov random fields to infer the final joint label probability. However, when dealing with a large set of labels, the parameters of these pairwise probabilities can be prohibitively large while lots of the parameters are redundant if the labels have highly overlapping meanings. Moreover, most of these methods either can not model higher-order correlations, or sacrifice computational complexity to model more complicated label relationships.

Objectives

- Our main goal is to develop a multi-label algorithm which is self-adaptive and able to cater large databases.
- Text classification system should take into account the semantic, sentiment and context of the text.
- Existing approaches has very low precision, so our aim is to develop an algorithm which is high on performance.

2

Literature Survey

In this chapter, we will focus on the work already done in text classification and multi-label algorithms. We will also analyse the difference between the algorithms and use data from previous implementations in other publications to offer a comparative study.

2.1 Related Work

Most of the progress in multi-label classification is to transform the multi-label classification problem into one or more single-label classification or regression problems.

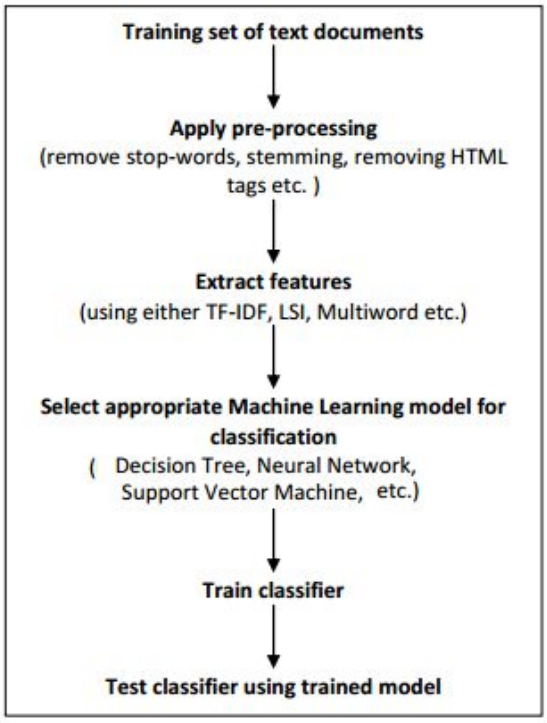


Figure 2. General steps for text classification model

There are two approaches studied in this field:

2.1.1 Bag of Words (BoW):

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

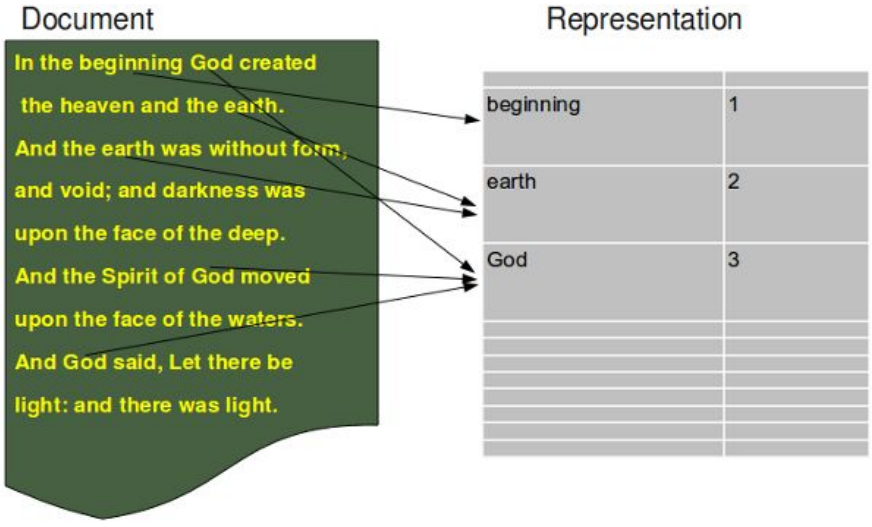


Figure 3. Bag of Words

Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

SVM: Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiate the two classes very well.
For this problem, while a movie can belong to multiple genres, whether tagging it with a particular genre is just a binary classification problem. Specifically, one can group all movies of a particular genre together as the positive samples and the rest as negative samples and train a binary classifier with these two disjoint sets. This approach is generally known as One-Vs-All.

Neural Network: The backpropagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted.

Limitations with existing models:

- A key problem in text classification is feature representation, which is commonly based on the bag-of-words (BoW) model, where unigrams, bigrams, n-grams or some exquisitely designed patterns are typically extracted as features.
- Traditional feature representation methods often ignore the contextual information or word order in texts and remain unsatisfactory for capturing the semantics of the words.

2.1.2 Word Embedding

The word embedding technique began development in 2000. Bengio et al.[20] provided in a series of papers the "Neural probabilistic language models" to reduce the high dimensionality of words representations in contexts by "learning a distributed representation for words"[20].

For word embedding representation, a feed-forward neural network that takes words from a vocabulary as input and embeds them as vectors into a lower dimensional space, which it then fine-tunes through back-propagation and yields word embeddings as the weights of the first layer, which is usually referred to as Embedding Layer.

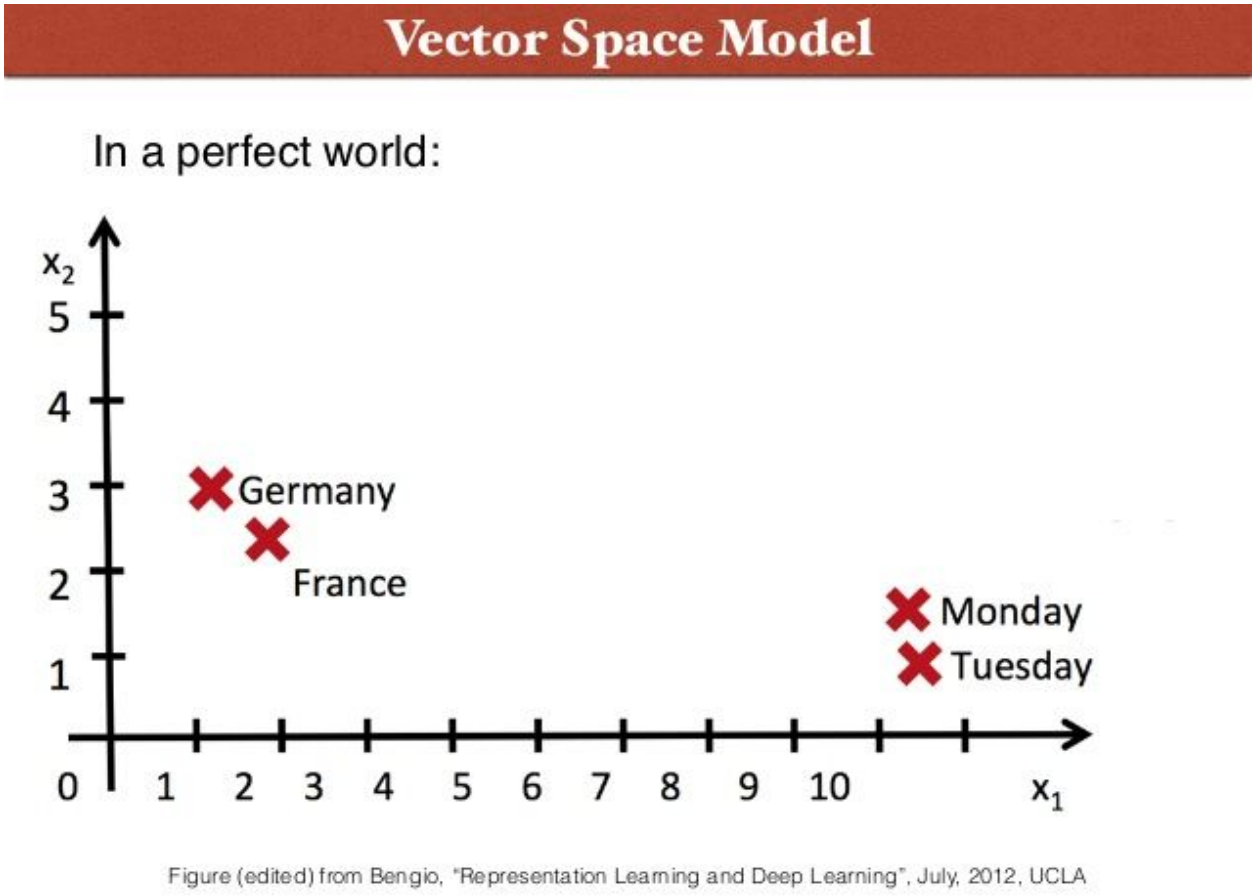


Figure 4. Word Embedding Example

The key problems with traditional models are that they fail to capture the contextual information. With the help of word embedding, some composition-based methods are proposed to capture the semantic representation of texts:

2.1.2.1 Architectures based on Word Embedding

Recursive NN: Socher et al. [10,11,12] proposed the Recursive Neural Network (Recursive NN) that has been proven to be efficient in terms of constructing sentence representations. However, the Recursive NN captures the semantics of a sentence via a tree structure. Its performance heavily depends on the performance of the textual tree construction. Moreover, constructing such a textual tree exhibits a time complexity of at least $O(n^2)$, where n is the length of the text. This would be too time-consuming when the model meets a long sentence or a document. Furthermore, the relationship between two sentences can hardly be represented by a tree structure. Therefore, Recursive NN is unsuitable for modeling long sentences or documents.

Recurrent NN: Another model, which only exhibits a time complexity O(n), is the Recurrent Neural Network (Recurrent NN). This model analyzes a text word by word and stores the semantics of all the previous text in a fixed-sized hidden layer [21]. The advantage of Recurrent NN is the ability to better capture the contextual information. This could be beneficial to capture semantics of long texts. However, the Recurrent NN is a biased model, where later words are more dominant than earlier words. Thus, it could reduce the effectiveness when it is used to capture the semantics of a whole document, because key components could appear anywhere in a document rather than at the end.

Convolutional Neural Network: To tackle the bias problem, the Convolutional Neural Network (CNN), an unbiased model is introduced to NLP tasks, which can fairly determine discriminative phrases in a text with a max-pooling layer. Thus, the CNN may better capture the semantic of texts compared to recursive or recurrent neural networks. The time complexity of the CNN is also O(n). However, previous studies on CNN's tends to use simple convolutional kernels such as a fixed window [22]. When using such kernels, it is difficult to determine the

window size: small window sizes may result in the loss of some critical information, whereas large windows result in an enormous parameter space (which could be difficult to train).

2.1.2.2 Activation Functions

Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions you may encounter in practice:

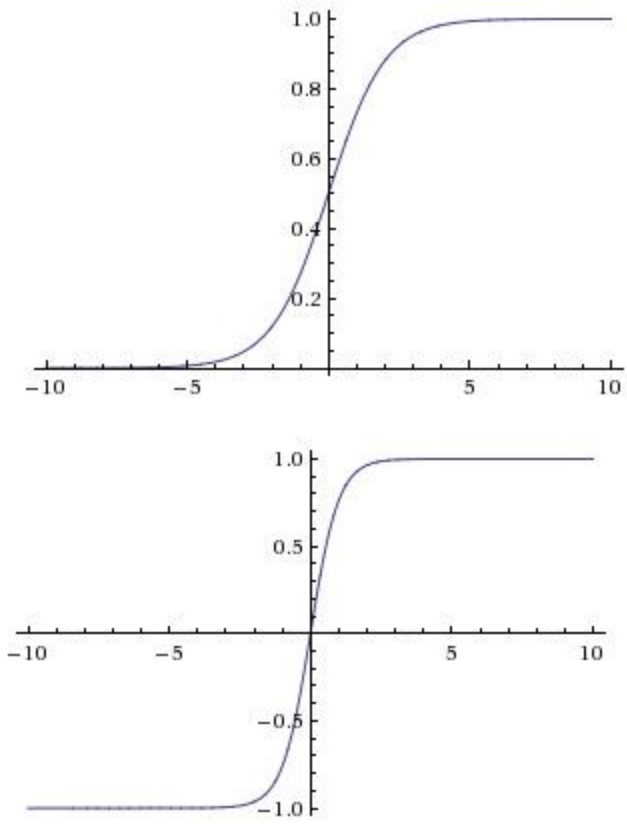


Figure 5. Tanh and Sigmoid activation functions

Top: Sigmoid non-linearity squashes real numbers to range between [0,1]

Bottom: The tanh non-linearity squashes real numbers to range between [-1,1].

Sigmoid. The sigmoid non-linearity has the mathematical form $\sigma(x)=1/(1+e^{-x})$ and is shown in the image above on the left. As alluded to in the previous section, it takes a real-valued number and "squashes" it into range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. The sigmoid function has seen frequent use historically since it has a nice interpretation as the firing rate of a neuron: from not firing at all (0) to fully-saturated firing at an assumed maximum frequency (1). In practice, the sigmoid non-linearity has recently fallen out of favor and it is rarely ever used. It has two major drawbacks:

- Sigmoids saturate and kill gradients. A very undesirable property of the sigmoid neuron is that when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. Recall that during backpropagation, this (local) gradient will be multiplied to the gradient of this gate's output for the whole objective. Therefore, if the local gradient is very small, it will effectively "kill" the gradient and almost no signal will flow through the neuron to its weights and recursively to its data. Additionally, one must pay extra caution when initializing the weights of sigmoid neurons to prevent saturation. For example, if the initial weights are too large then most neurons would become saturated and the network will barely learn.
- Sigmoid outputs are not zero-centered. This is undesirable since neurons in later layers of processing in a Neural Network (more on this soon) would be receiving data that is not zero-centered. This has implications on the dynamics during gradient descent, because if the data coming into a neuron is always positive (e.g. $\boldsymbol{x}>\mathbf{0}$ elementwise in $\boldsymbol{f}=\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{b}$)), then the gradient on the weights w will during backpropagation become either all be positive, or all negative (depending on the gradient of the whole expression $\boldsymbol{f}$). This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights. However, notice that once these gradients are added up across a batch of data the final update for the weights can have variable signs, somewhat

mitigating this issue. Therefore, this is an inconvenience but it has less severe consequences compared to the saturated activation problem above.

Tanh. The tanh non-linearity is shown on the image above on the right. It squashes a real-valued number to the range [-1, 1]. Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity. Also note that the tanh neuron is simply a scaled sigmoid neuron, in particular the following holds: $\boldsymbol{tanh(x)=2\sigma(2x)-1}$.

We will be using **Tanh** because of its above mentioned advantages.

3

Proposed Work

In this chapter, we will discuss the motivation for taking up this problem and then briefly talk about the approach that we are going to use for multi-label text-classification.

3.1 Motivation

There has been a lot of work done in text classification but classifying text in multiple-labels has not been explored much and is in its nascent stages. The motivation for taking up the multi-label text classification problem is to design a novel algorithm for the task which can efficiently scale up to settings involving large datasets and data streams.

3.2 Extended Recurrent Convolutional Neural Network

To address the limitation of the above models, we propose an Extended Recurrent Convolutional Neural Network (ERCNN) and apply it to the task of text classification. First, we apply a bi-directional recurrent structure, which may introduce considerably less noise compared to a

traditional window based neural network, to capture the contextual information to the greatest extent possible when learning word representations. Moreover, the model can reserve a larger range of the word ordering when learning representations of texts. Second, we employ a max-pooling layer that automatically judges which features play key roles in text classification, to capture the key component in the texts. By combining the recurrent structure and max-pooling layer, our model utilizes the advantage of both recurrent neural models and convolutional neural models. Furthermore, our model exhibits a time complexity of O(n), which is linearly correlated with the length of the text length.

For extending the above model for multi-label classification, We then add a fully connected output layer of size |L|, where L are the labels which we are using for classification, with a sigmoid activation function, which produces a probability for each of our potential labels. During training, these probabilities are used to compute the error, while during testing, we round each of these labels to 0 or 1 depending upon a set threshold.

4

Design and Implementation

In the previous chapter, we discussed the ERCNN approach that we will use for text classification task. This chapter contains the design of the architecture that we have implemented as well as other details of the implementation. The block diagram given below is the architecture that we have implemented.

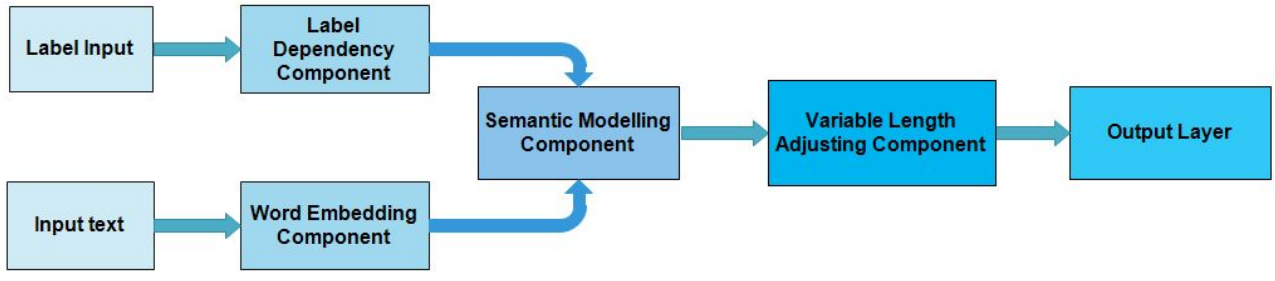


Figure 6. The architecture of the proposed ERCNN model for multilabel classification.

We will understand each component of the above architecture in detail now.

4.1 Modelling of ERCNN

We propose an Extended RCNN framework for multilabel classification problem. The architecture of the RCNN framework.

We propose a deep neural model to capture the semantics of the text. Figure 4 shows the network structure of our model. The input of the network is a document D, which is a sequence of words $w_1, w_2, ..., w_n$. The output of the net-work contains label elements. We use $p(k|D;Q)$ to denote the probability of the document being label k, where Q is the parameters in the network.

4.2 Word Embedding Component

As discussed in section 2.1.2, word embedding is a technique to represent words in the form of vectors. This component of our architecture uses Google's word2vec library to represent words in the text in the form of vectors.

- This component changes the word to its vector representation.
- Word embedding takes into account the semantic relations between the words.
- Example:
  king−man+woman≈queen

4.3 Label Dependency Component

- To model label correlations, we use Label powerset method with Ensembles of Pruned Sets (EPS) improvement

Label Powerset Method (LP)

- Make a single multi-class problem with 2^L possible class values and train with any

off-the-shelf multi-class classifier.

X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	1	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

X	$Y \in 2^L$
$x^{(1)}$	0110
$x^{(2)}$	1000
$x^{(3)}$	0110
$x^{(4)}$	1001
$x^{(5)}$	0001

Figure 7. Label Powerset Method

Label Powerset Method (LP) with Ensembles of Pruned Sets (EPS)

- 'Prune' out infrequent label sets, replace with sampled frequent sets.
- keep (most) label dependency information, reduce complexity and other LP issues.

X	$Y \in 2^L$
$x^{(1)}$	0110
$x^{(2)}$	1000
$x^{(3)}$	0110
$x^{(4)}$	1001
$x^{(5)}$	0001

X	$Y \in 2^L$
$x^{(1)}$	0110
$x^{(3)}$	0110
$x^{(4)}$	0001
$x^{(5)}$	0001

X	$Y \in 2^L$
$x^{(1)}$	0110
$x^{(2)}$	1000
$x^{(3)}$	0110
$x^{(4)}$	0001
$x^{(4)}$	1000
$x^{(5)}$	0001

Figure 8. Label Powerset Method with Ensembles of Pruned Sets Example

4.4 Semantic Modelling Component (Recurrent Structure)

We combine a word and its context to present a word. The contexts help us to obtain a more precise word meaning. In our model, we use a recurrent structure, which is a bidirectional recurrent neural network, to capture the contexts.

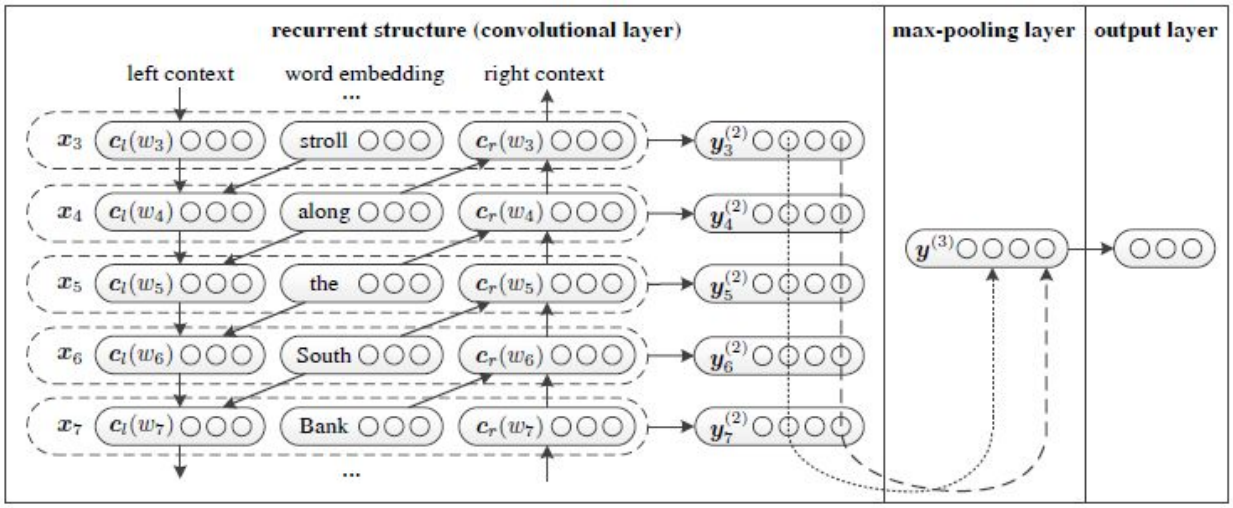


Figure 9. The structure of the recurrent convolutional neural network. This figure is a partial example of the sentence "A sunset stroll along the South Bank affords an array of stunning vantage points", and the subscript denotes the position of the corresponding word in the original sentence.

We define $c_l(w_i)$ as the left context of word w_i and $c_r(w_i)$ as the right context of word w_i. Both $c_l(w_i)$ and $c_r(w_i)$ are dense vectors with |c| real value elements. The left-side context $c_l(w_i)$ of word w_i is calculated using Equation (1), where e(w_{i-1}) is the word embedding of word w_{i-1}, which is a dense vector with |e| real value elements. $c_l(w_{i-1})$ is the left-side context of the previous word w_{i-1}. The left-side context for the first word in any document uses the same shared parameters $c_l(w_i)$. $W^{(l)}$ is a matrix that transforms the hidden layer (context) into the next hidden layer. $W^{(sl)}$ is a matrix that is used to combine the semantic of the current word with the next word's left context. f is a non-linear activation function. The right-side context $c_r(w_i)$ is calculated in a similar manner, as shown in Equation (2). The right-side contexts of the last word in a document share the parameters $c_r(w_n)$.

$$c_l(w_i) = f(W^{(l)} c_l(w_{i-1}) + W^{(sl)} e(w_{i-1})) \quad (1)$$

$$c_r(w_i) = f(W^{(r)} c_r(w_{i+1}) + W^{(sr)} e(w_{i+1})) \quad (2)$$

As shown in Equations (1) and (2), the context vector captures the semantics of all left- and right-side contexts.

For example, in Figure E4, $c_l(w_7)$ encodes the semantics of the left-side context "stroll along the South" along with all previous texts in the sentence, and $c_r(w_7)$ encodes the semantics of the right-side context "affords an . . . ". Then, we define the representation of word w_i in Equation (3), which is the concatenation of the left-side context vector $c_l(w_i)$, the word embedding $e(w_i)$ and the right-side context vector $c_r(w_i)$. In this manner, using this contextual information, our model may be better able to disambiguate the meaning of the word w_i compared to conventional neural models that only use a fixed window (i.e., they only use partial information about texts).

$$\boldsymbol{x_i = [\, c_l(w_i); e(w_i); c_r(w_i)\,]} \qquad \textbf{(3)}$$

The recurrent structure can obtain all c_l in a forward scan of the text and c_r in a backward scan of the text. The time complexity is O(n). After we obtain the representation x_i of the word w_i, we apply a linear transformation together with the tanh activation function to x_i and send the result to the next layer.

$$\boldsymbol{y_i^{(2)} = tanh\,(W^{(2)} x_i + b^{(2)})} \qquad \textbf{(4)}$$

$y_i^{(2)}$ is a latent semantic vector, in which each semantic factor will be analyzed to determine the most useful factor for representing the text.

4.5 Variable Length Adjusting Component (Max Pooling Layer)

The convolutional neural network in our model is designed to represent the text. From the perspective of convolutional neural networks, the recurrent structure we previously mentioned is the convolutional layer.

When all of the representations of words are calculated, we apply a max-pooling layer.

$$\boldsymbol{y^{(3)} = \max y_i^{(2)}} \qquad \forall\ i \in (1, n) \quad \textbf{(5)}$$

The max function is an element-wise function. The k-th element of $y^{(3)}$ is the maximum in the k-th elements of $y_i^{(2)}$.

The pooling layer converts texts with various lengths into a fixed-length vector. With the pooling layer, we can capture the information throughout the entire text. There are other types of pooling layers, such as average pooling layers [22]. We do not use average pooling here because only a few words and their combination are useful for capturing the meaning of the document. The max-pooling layer attempts to find the most important latent semantic factors in the document. The pooling layer utilizes the output of the recurrent structure as the input. The time complexity of the pooling layer is O(n). The overall model is a cascade of the recurrent structure and a max-pooling layer, therefore, the time complexity of our model is still O(n).

The last part of our model is an output layer. Similar to traditional neural networks, it is defined as

$$\boldsymbol{y}^{(4)} = \boldsymbol{W}^{(4)} \boldsymbol{y}^{(3)} + \boldsymbol{b}^{(4)} \qquad \textbf{(6)}$$

Finally, the softmax function is applied to $y^{(4)}$.It can convert the output numbers into probabilities.

$$p_i = \frac{exp(y_i^{(4)})}{\sum_{k=1}^{n} exp(y_i^{(4)})} \qquad \textbf{(7)}$$

4.6 Training

Training Network parameters: We define all of the parameters to be trained as θ.

$$\theta = \{E, b^{(2)}, b^{(4)}, c_l(w_1), c_r(w_n), W^{(2)}, W^{(4)}, W^{(l)}, W^{(r)}, W^{(sl)}, W^{(sr)}\} \qquad (8)$$

Specifically, the parameters are word embeddings $E \in \mathrm{R}^{|e|\times|\mathrm{V}|}$, the bias vectors $b^{(2)} \in \mathrm{R}^{H}$, $b^{(4)} \in \mathrm{R}^{O}$, the initial contexts $c_l(w_1)$, $c_r(w_n) \in \mathrm{R}^{|c|}$ and the transformation matrixes $W^{(2)} \in \mathrm{R}^{H\times(|e|+2|c|)}$, $W^{(4)} \in \mathrm{R}^{O\times H}$, $W^{(l)}$, $W^{(r)} \in \mathrm{R}^{|c|\times|c|}$, $W^{(sl)}$, $W^{(sr)} \in \mathrm{R}^{|e|\times|c|}$, where |V | is the number of words in the vocabulary, H is the hidden layer size, and O is the number of document types.

The training target of the network is used to maximize the log-likelihood with respect to θ:

$$\theta \rightarrow \sum_{D\in D} \log p(label_D \,|\mathbf{D}, \theta) \qquad (9)$$

where D is the training document set and label D is the correct label of document D.

We use stochastic gradient descent [23]to optimize the training target. In each step, we randomly select an example (D, label D) and make a gradient step.

$$\theta \leftarrow \theta + \alpha \frac{\partial \log p(label\ D\,|D, \theta)}{\partial \theta} \qquad (10)$$

where α is the learning rate.

We use one trick that is widely used when training neural networks with stochastic gradient descent in the training phase. We initialize all of the parameters in the neural network from a uniform distribution. The magnitude of the maximum or minimum equals the square root of the "fan-in"[24]. The number is the network node of the previous layer in our model. The learning rate for that layer is divided by "fan-in".

Pre-training Word Embedding: Word embedding is a distributed representation of a word. Distributed representation is suitable for the input of neural networks. Traditional representations, such as one-hot representation, will lead to the curse of dimensionality [20]. Recent research [25] shows that neural networks can converge to a better local minima with a suitable unsupervised pre-training procedure. In this work, we use the Skip-gram model to pre-train the word embedding. this model is the state-of-the-art in many NLP tasks [26]. The Skip-gram model trains the embeddings of words w_1 , w_2. . . w_T by maximizing the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} log\, p(w_{t+j}\,|w_t) \qquad (11)$$

$$p(w_b\,|w_a) = \frac{exp\,(e'(w_b)^T e(w_a))}{\sum_{k=1}^{|V|} exp(e'(w_k)\, e(w_a))} \qquad (12)$$

where |V | is the vocabulary of the unlabeled text. $e'(w_i)$ is another embedding for w_i. We use the embedding e because some speed-up approaches (e.g., hierarchical softmax [27]) will be used here, and e' is not calculated in practice.

4.7 ERCNN Algorithm

The training network parameters would be:

$$\theta = \{E,\ b^{(2)},\ b^{(4)},\ c_l(w_1),\ c_r(w_n),\ W^{(2)}, W^{(4)},\ W^{(l)},\ W^{(r)},\ W^{(sl)},\ W^{(sr)}\}$$

(described in section 4)

The training target of the network is used to maximize the log-likelihood with respect to θ:

$$\theta \mapsto \sum_{D\in\mathbb{D}} \log p(class_D|D,\theta)$$

where D is the training document set and label $_D$ is the correct label of document D.

We are using stochastic gradient descent to optimize the training target. We initialize all of the parameters in the neural network from a uniform distribution.

We then add a fully connected output layer of size |L|, where L are the labels in which the text has to be classified, with a sigmoid activation function, which produces a probability for each of our potential labels. During training, these probabilities are used to compute the error, while during testing, we round each of these labels to 0 or 1 depending upon a set threshold.

5

Experimentation and Results

5.1 Dataset

We will use movie review database to classify movies based on their synopsis. Public movies' database such as IMDB provides genre information to assist searching. The tagging of movies' genres is still a manual process which involves the collection of users' suggestions sent to known email addresses of IMDB. Movies are often registered with inaccurate genres. Automatic genres classification of a movie based on its synopsis not only speeds up the classification process by providing a list of suggestion but the result may potentially be more accurate than an untrained human.

The Internet Movie Database (IMDB) has for synopsis of the synopsis and the genre tags for each movie.

There are 2 data files

plot.list.gz : Contains synopsis of 1,58,840 movies.

genre.list.gz: Contains 7,46,883 genre tags for different movies.

5.2 Data Pre-processing

Data Pre-processing involves extracting the required features from the data given in text format. There are multiple steps in data processing which are described as follows:

- Converting both the .list files to .txt and then to .csv using regular expression in python script. Reason for performing the conversion is, it becomes easy to apply ML algorithms on csv files.
- Choose the movie titles for which both synopsis and genre data is available. These turn out to be 16,000.
- 80% was randomly chosen as training data and rest 20% as test data.
- In total 40 genres, selected the 26 most popular genres.
- Process Synopsis to generate bag of words.
- Apply NLTK to remove all stopwords.
- Remove all numerical words.
- Python stemmer, so words with same index were mapped to single word.
- Created a dictionary from 16000 movies synopsis, so 63,840 words was generated.
- Out of the 63,840 words, only those which have occurred more than 15 times in all the training samples were used in BoW representation so only 10,656 (denoted as V) words were left.
- Term frequency inverse document frequency (tfidf) of the words were then computed.

$$tfidf(w_k, d_i) = w_{ki} * log \frac{m}{\sum_{d=1}^{m} 1\{w_{kd} > 0\}}$$

where w_{ki} is the frequency of the k-th word in the i-th movie's synopsis (d_i), and m is the number of training/test sets.

Genre Distribution

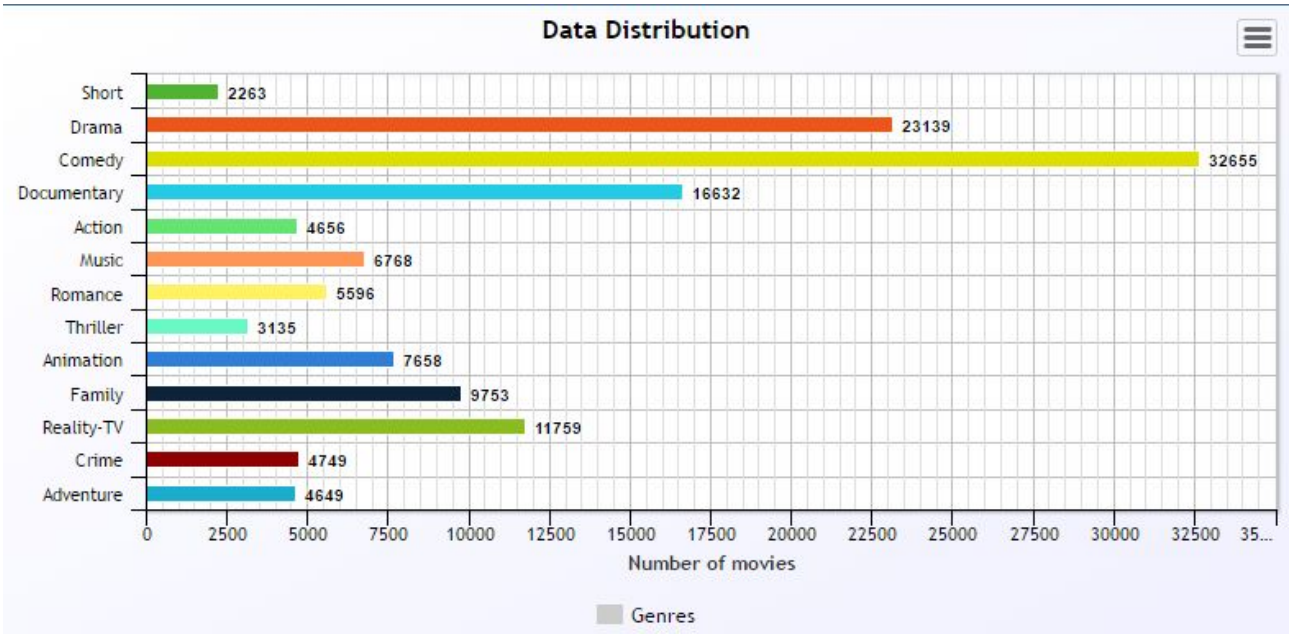


Figure 10. Data Distribution

Code snippets can be found in Appendix.

5.3 Experimentation Setup

Training Hyper-parameters

Learning rate of the stochastic gradient descent = 0.01,
Hidden layer size as H = 100,
Vector size of the word embedding as |e| = 50
Size of the context vector as |c| = 50.

We train word embeddings using the default parameter in word2vec with the Skip-gram algorithm. We used Wikipedia dumps in English to train the word embedding.

5.4 Performance parameters

The performance is based on the following factors:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F Measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where,
TP= number of true positive
FP= number of false positive and
FN= number of false negative

5.5 Results and Comparison

Approach (BoW)	Precision	Recall	F Measure
SVM	0.661	0.396	0.472
Random Forest	0.582	0.382	0.421
Backpropagation NN	0.67630	0.41513	0.49896

Approach(Word Embedding)	Precision	Recall	F Measure
Recurrent NN	0.7406	0.4954	0.5936
Convolutional NN	0.7061	0.5274	0.6038
ERCNN	0.7714	0.5863	0.6662

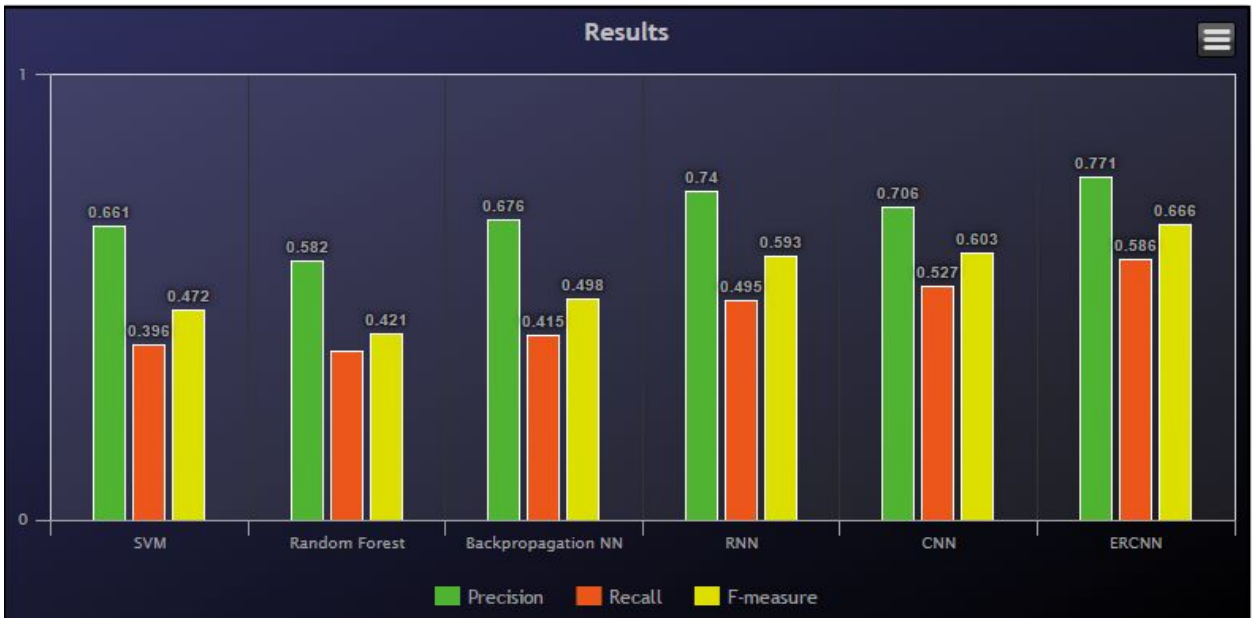


Figure 11. Results comparing ERCNN with other Text Classification approaches

The results show that all the Word Embedding approaches perform better than the Bag of Words (BoW) approaches. This can be explained based on the fact that word embedding captures the semantics of the word which the BoW doesn't take into account.

The Recurrent NN uses the word vectors gives more weight to the words which come later in the text. ERCNN overcomes this drawback by using a bi-directional recurrent structure which takes both left context and right context into consideration. ERCNN also uses the max pooling layer present in CNN to extract the most important features from the text. It produces better results than CNN because instead of using a fixed-size window like CNN, ERCNN uses the the left and right context which help to solve the problem of too much information due to large window size or loss of critical information due to a small window size.

6

Conclusions and Future Scope

Following are the conclusions based on the results:

- The approaches using Bag of Words(BoW), (SVM, Backpropagation NN, and Random Forest) performed well but as BoW fails to capture the semantic of words, they can only have accuracy up to a particular limit.
- As shown in the results, Extended Recurrent Convolutional Neural Network(ERCNN), outperforms the approaches using BoW. This is because it uses Word Embedding which takes into account the semantic of the words.
- ERCNN gives better results than RNN because the bi-directional recurrent structure overcomes the problem of words appearing later in the text getting more weight.

As future scope of the problem, we can implement an architecture which also takes the label dependencies into account. Real data mostly has a lot of labels in which the text has to be classified. For large number of labels, we have to consider the label dependencies to get the correct set of labels.

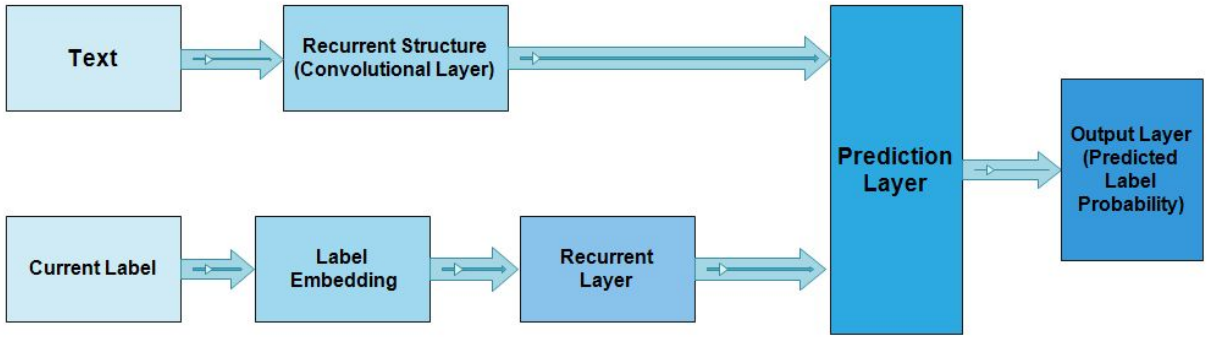


Figure 12. Proposed RCNN-RNN architecture for Text-Classification

The above architecture uses multi-label Recurrent neural network(RNN) to learn a joint text-label embedding to model the semantic relevance between text and labels. The text representation as vectors are generated using RCNN. In the Prediction Layer, the text and the labels are projected to the same low-dimensional space to model the image-text relationship as well as the label redundancy.

A similar architecture has been used in [18] for multi-label image classification where RNN is used for label representation and CNN is used to generate image embedding vectors. The results are significantly better than the others approaches used for comparison including KNN and RNN.

References

[1] Imdb alternative interfaces. URL http://www.imdb.com/interfaces.

[2] Natural language toolkit. URL http://www.nltk.org.

[3] Imdb genres database. URL : ftp://ftp.fu-berlin.de/pub/misc/movies/database/genres.list.gz.

[4] Imdb plots database. URL ftp://ftp.fu-berlin.de/pub/misc/movies/database/plot.list.gz.

[5] R. Boulton. Pystemmer 1.0.1. URL:http://pypi.python.org/pypi/PyStemmer/1.0.1.

[6] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST)2.3 (2011): 27.

[7] Zhang, Min-Ling, and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." Pattern recognition 40.7 (2007): 2038-2048.

[8] McCallum, Andrew. "Multi-label text classification with a mixture model trained by EM." AAAI'99 workshop on text learning. 1999.

[9] Ueda, Naonori, and Kazumi Saito. "Parametric mixture models for multi-labeled text." Advances in neural information processing systems. 2002.

[10]Socher, Richard, et al. "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection." Advances in Neural Information Processing Systems. 2011..

[11]Socher, Richard, et al. "Semi-supervised recursive autoencoders for predicting sentiment distributions." Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.

[12]Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[13]Oquab, Maxime, et al. "Learning and transferring mid-level image representations using convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[14]Ho, Ka-Wing. "Movies' Genres Classification by Synopsis."

[15]McAfee, Lawrence. "Document classification using deep belief nets."CS224n, Sprint 42 (2008).

[16]Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang. "Recurrent Neural Network for Text Classification with Multi-Task Learning." arXiv preprint arXiv:1605.05101 (2016).

[17]Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." Advances in Neural Information Processing Systems. 2015.

[18]Wang, Jiang, et al. "CNN-RNN: A Unified Framework for Multi-label Image Classification." arXiv preprint arXiv:1604.04573 (2016).

[19]Berger, Mark J. "Large Scale Multi-label Text Classification with Semantic Word Vectors."

[20]Bengio, Yoshua, et al. "A neural probabilistic language model." journal of machine learning research 3.Feb (2003): 1137-1155.

[21]Elman, Jeffrey L. "Finding structure in time." Cognitive science 14.2 (1990): 179-211.

[22]Collobert, Ronan, et al. "Natural language processing (almost) from scratch." Journal of Machine Learning Research 12.Aug (2011): 2493-2537.

[23]Bottou, Léon. "Stochastic gradient learning in neural networks." Proceedings of Neuro-Nımes 91.8 (1991).

[24]Plaut, David C., and Geoffrey E. Hinton. "Learning sets of filters using back-propagation." Computer Speech & Language 2.1 (1987): 35-61.

[25]Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

[26]Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors." ACL (1). 2014.

[27]Morin, Frederic, and Yoshua Bengio. "Hierarchical Probabilistic Neural Network Language Model." Aistats. Vol. 5. 2005.

Appendix

Data preprocessing:

The raw data was in a particular format in a text file.

To extract the data from text and obtaining fruitful information from it we converted this raw data into excel file using regular expression in python language.

```
#Regular expression that matches the Data format

matchObj = re.match( r'"(.*)"\s+\((.*)\)\s+(.*)', lines[i], re.M|re.I)
```

Then we merged the movie details and synopsis into a single excel file, which has movie name and year as unique id and synopsis and genre info.

```
#Merging the data into single excel file

inters=pd.merge(df,df2, on=['Movie','Year'], how='inner')
```

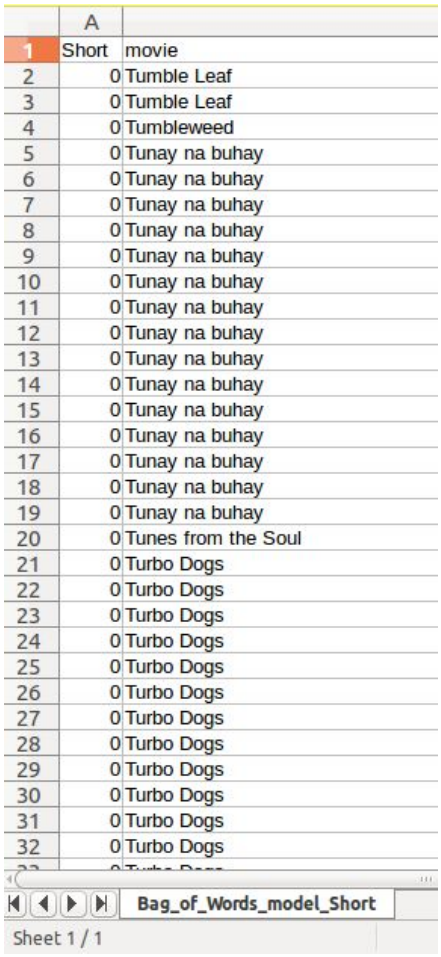
RCNN model :

This is the forward pass of the model:

```
def forward(self, x_list):
    cl_list = []
    for x in x_list:
        wvec = self.embed(x)
        cl_list.append(wvec)          #Capturing the left context
    cr_list = []
    for x in reversed(x_list):
        wvec = self.embed(x)
        cr_list.append(wvec)          #Capturing the right context
    xi_list = []
    for cl, cr in zip(cl_list, cr_list):
```

```
            xi_list.append(F.concat((cl, cr)))
        yi_list = []
        for xi in xi_list:
            yi_list.append(F.tanh(self.fc1(xi)))
        y3 = yi_list[0]
        for yi in yi_list[1:]:
            y3 = F.maximum(yi, y3)
        y4 = self.fc2(y3)
        return y4
```

Pre-processed Data:

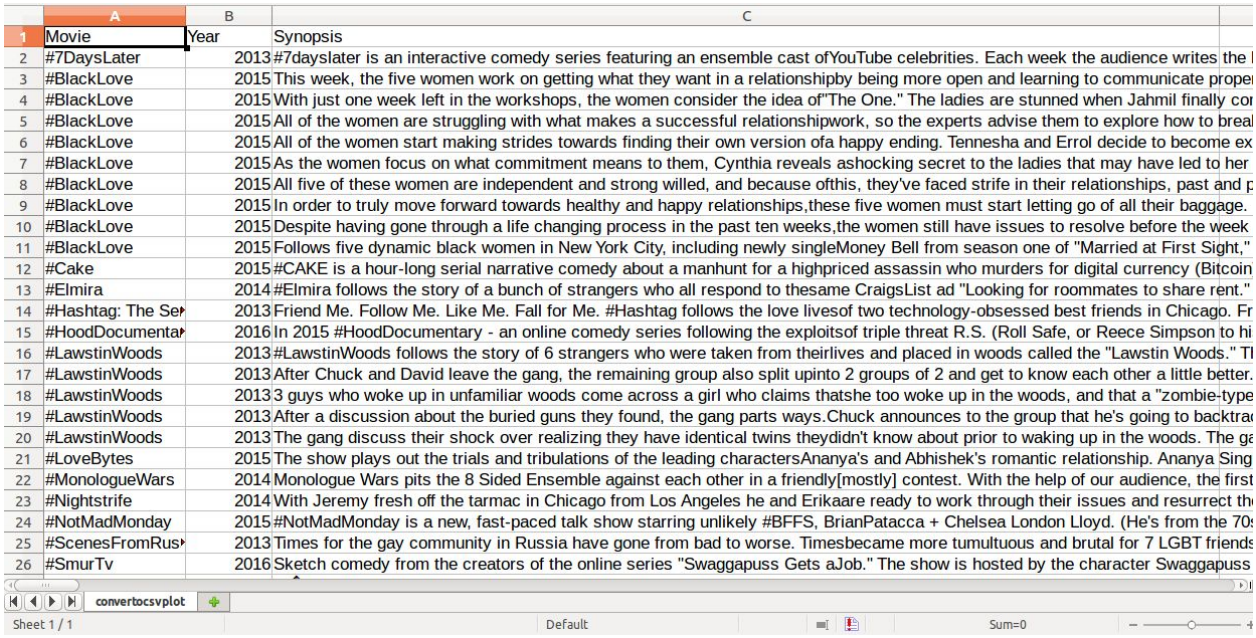


	A	
1	Short	movie
2	0	Tumble Leaf
3	0	Tumble Leaf
4	0	Tumbleweed
5	0	Tunay na buhay
6	0	Tunay na buhay
7	0	Tunay na buhay
8	0	Tunay na buhay
9	0	Tunay na buhay
10	0	Tunay na buhay
11	0	Tunay na buhay
12	0	Tunay na buhay
13	0	Tunay na buhay
14	0	Tunay na buhay
15	0	Tunay na buhay
16	0	Tunay na buhay
17	0	Tunay na buhay
18	0	Tunay na buhay
19	0	Tunay na buhay
20	0	Tunes from the Soul
21	0	Turbo Dogs
22	0	Turbo Dogs
23	0	Turbo Dogs
24	0	Turbo Dogs
25	0	Turbo Dogs
26	0	Turbo Dogs
27	0	Turbo Dogs
28	0	Turbo Dogs
29	0	Turbo Dogs
30	0	Turbo Dogs
31	0	Turbo Dogs
32	0	Turbo Dogs

Bag_of_Words_model_Short

Sheet 1 / 1

Short movies

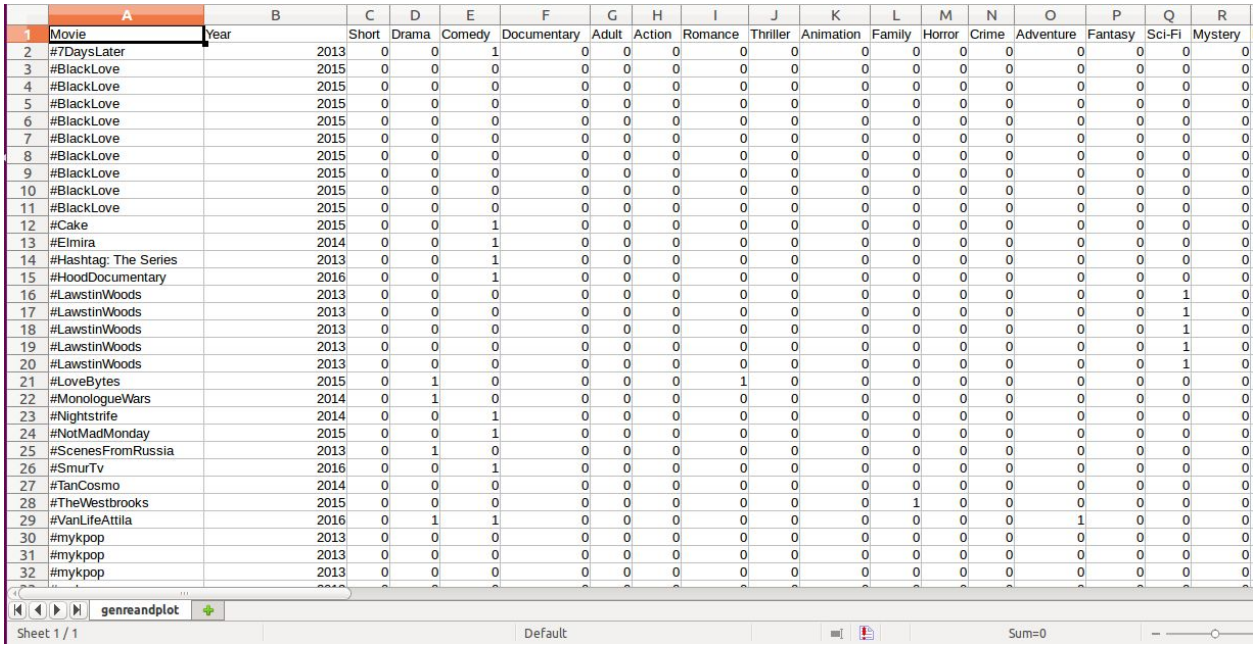


	Movie	Year	Synopsis
2	#7DaysLater	2013	#7dayslater is an interactive comedy series featuring an ensemble cast ofYouTube celebrities. Each week the audience writes the
3	#BlackLove	2015	This week, the five women work on getting what they want in a relationshipby being more open and learning to communicate prope
4	#BlackLove	2015	With just one week left in the workshops, the women consider the idea of"The One." The ladies are stunned when Jahmil finally co
5	#BlackLove	2015	All of the women are struggling with what makes a successful relationshipwork, so the experts advise them to explore how to breal
6	#BlackLove	2015	All of the women start making strides towards finding their own version ofa happy ending. Tennesha and Errol decide to become ex
7	#BlackLove	2015	As the women focus on what commitment means to them, Cynthia reveals ashocking secret to the ladies that may have led to her
8	#BlackLove	2015	All five of these women are independent and strong willed, and because ofthis, they've faced strife in their relationships, past and p
9	#BlackLove	2015	In order to truly move forward towards healthy and happy relationships,these five women must start letting go of all their baggage.
10	#BlackLove	2015	Despite having gone through a life changing process in the past ten weeks,the women still have issues to resolve before the week
11	#BlackLove	2015	Follows five dynamic black women in New York City, including newly singleMoney Bell from season one of "Married at First Sight,"
12	#Cake	2015	#CAKE is a hour-long serial narrative comedy about a manhunt for a highpriced assassin who murders for digital currency (Bitcoin
13	#Elmira	2014	#Elmira follows the story of a bunch of strangers who all respond to thesame CraigsList ad "Looking for roommates to share rent."
14	#Hashtag: The Se	2013	Friend Me. Follow Me. Like Me. Fall for Me. #Hashtag follows the love livesof two technology-obsessed best friends in Chicago. Fr
15	#HoodDocumentar	2016	In 2015 #HoodDocumentary - an online comedy series following the exploitsof triple threat R.S. (Roll Safe, or Reece Simpson to hi
16	#LawstinWoods	2013	#LawstinWoods follows the story of 6 strangers who were taken from theirlives and placed in woods called the "Lawstin Woods." TI
17	#LawstinWoods	2013	After Chuck and David leave the gang, the remaining group also split upinto 2 groups of 2 and get to know each other a little better.
18	#LawstinWoods	2013	3 guys who woke up in unfamiliar woods come across a girl who claims thatshe too woke up in the woods, and that a "zombie-type
19	#LawstinWoods	2013	After a discussion about the buried guns they found, the gang parts ways.Chuck announces to the group that he's going to backtra
20	#LawstinWoods	2013	The gang discuss their shock over realizing they have identical twins theydidn't know about prior to waking up in the woods. The ga
21	#LoveBytes	2015	The show plays out the trials and tribulations of the leading charactersAnanya's and Abhishek's romantic relationship. Ananya Sing
22	#MonologueWars	2014	Monologue Wars pits the 8 Sided Ensemble against each other in a friendly[mostly] contest. With the help of our audience, the first
23	#Nightstrife	2014	With Jeremy fresh off the tarmac in Chicago from Los Angeles he and Erikaare ready to work through their issues and resurrect th
24	#NotMadMonday	2015	#NotMadMonday is a new, fast-paced talk show starring unlikely #BFFS, BrianPatacca + Chelsea London Lloyd. (He's from the 70
25	#ScenesFromRus	2013	Times for the gay community in Russia have gone from bad to worse. Timesbecame more tumultuous and brutal for 7 LGBT friends
26	#SmurTv	2016	Sketch comedy from the creators of the online series "Swaggapuss Gets aJob." The show is hosted by the character Swaggapuss

convertocsvplot

Sheet 1 / 1 Default Sum=0

Data after extracting year and synopsis



	Movie	Year	Short	Drama	Comedy	Documentary	Adult	Action	Romance	Thriller	Animation	Family	Horror	Crime	Adventure	Fantasy	Sci-Fi	Mystery
2	#7DaysLater	2013	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	#BlackLove	2015	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	#Cake	2015	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	#Elmira	2014	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	#Hashtag: The Series	2013	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15	#HoodDocumentary	2016	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
16	#LawstinWoods	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
17	#LawstinWoods	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
18	#LawstinWoods	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
19	#LawstinWoods	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
20	#LawstinWoods	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
21	#LoveBytes	2015	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22	#MonologueWars	2014	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	#Nightstrife	2014	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
24	#NotMadMonday	2015	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
25	#ScenesFromRussia	2013	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	#SmurTv	2016	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
27	#TanCosmo	2014	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	#TheWestbrooks	2015	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
29	#VanLifeAttila	2016	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0
30	#mykpop	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	#mykpop	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	#mykpop	2013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

genreandplot

Sheet 1 / 1 Default Sum=0

Movies with corresponding labels as 1