

1. INTRODUCTION

Language is a powerful tool to communicate and convey information. It is also a means to express emotion and sentiment. Sentiment analysis is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes and emotions towards entities such as products, services, organizations, individuals, issues, events, movies and topics. In simple words, it is used to track the mood of the public. It uses natural language processing and data mining techniques to the problem of extracting opinions from text. In data mining research field, machine learning techniques have been applied to automatically identify the information content in text. In recent years, the exponential increase in the Internet usage and exchange of public opinion is the driving force behind sentiment analysis. The web is a huge repository of structured and unstructured data. The analysis of this data to extract hidden public opinion and sentiment is a challenging task.

Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to the overall document. The most useful application of sentiment analysis is the sentiment classification of product reviews. This sentiment classification can be categorized into positive and negative. Positive rating returns are good, negative ratings are bad review. Users express opinions through review sites such as Amazon, Internet Movie Database (IMDB), as well as through blogs, discussion forums, peer-to-peer networks, user feedback, comments and various types of social network sites. These kinds of online media have resulted in large quantities of textual data containing opinion and facts. Over the years, there has been extensive research aimed at analyzing and classifying text and data, where the objective is to assign predefined category labels to documents based upon learned models. This has led to the development of sentiment analysis and classification systems.

Movie reviews are a good source for this kind of work because authors clearly express an opinion and authors are accompanied by rating that makes it easier to train learning algorithms on this data. Most text classification methods that classify a given document into one of the predefined classes are based on bag of words where each document is represented by set of words. The rapid growth of the World Wide Web has facilitated increased online communication

and opened up newer avenues for the general public to post their opinions online. This has led to generation of large amounts of online content rich in user opinions, sentiments, emotions, and evaluations. Recently, many web sites have offered reviews of items like books, cars, hotels, vacation destinations, movies etc. Informers describe the items in some detail and evaluate them as good/bad, preferred/not preferred and positive/negative. That is, whether people recommend or do not recommend a particular item. For example, people express their views for movies as, “I like movie and the story is fantastic”.

In sentimental classification problem, movie review mining is a challenge. The inspiration for this work has come from studies in classification. Research area of sentiment analysis is motivated for doing sentiment classification using new combinations of classifiers for improving accuracy. A modern approach towards sentiment classification is to use machine learning techniques which inductively build a classification model of a given set of categories by training several sets of labeled document.

The existing techniques of sentiment analysis use a predefined dataset for detection of emotions. These techniques are based on traditional classification models such as Naïve Bayes (NB), Naïve Bayes with bag of bigram features (BiNB) and support vector machines (SVM). However, these models could not reach the desired levels of accuracy due to their rule-based or corpus-based nature which restricts the accuracy of results to the size of the training data and the parameters of the model.

To overcome these challenges and inspired by the success of deep learning-based approaches, in our project we have applied a deep learning model - Convolutional Neural Network – to handle the task of sentiment analysis. Deep learning has an edge over the traditional machine learning algorithms, like SVM and Naïve Bayes, for sentiment analysis because of its potential to overcome the challenges faced by sentiment analysis and handle the diversities involved, without the expensive demand for manual feature engineering. Deep learning models promise one thing - given sufficient amount of data and sufficient amount of training time, they can perform the task of sentiment classification on any text genre with minimal restrictions and no task-specific or data-specific manual feature engineering.

In our project first, we used word2vec proposed by Google to compute vector representations of words, which is then given as input to the CNN. The purpose of using word2vec is to gain the vector representation of a word and reflect the distance of words. This will lead to a better initialization of CNN parameters and thus will efficiently improve the performance of the task at hand. This is followed by a max-pooling layer, which combines the outputs of the neuron clusters and identifies those words which play the major role in the prediction of sentiment.

1.1. Existing System

Extensive research has been done over the past years to exploit popular machine learning algorithms for the task of sentiment classification. Depending on the problem statement in opinion mining, these classifiers have shown good performance accuracy, provided proper feature engineering and pre-processing steps are carried out prior to the classification process.

Naive Bayes classifier is the simplest and the most widely used probabilistic classification algorithm. It is based on Bayes' Theorem. It basically calculates the posterior probabilities of events and assigns the label with the maximum posterior probability to the event.

Support Vector Machines (SVMs) have proved to be highly effective for the categorization of documents based on similar topics. This method aims to find large margin between the different classes. It is a supervised learning model which analyzes data and learns patterns which can be used to classify the data. Support vector machines attempt to find a hyperplane (in case of 2-class classification problem) which not only separates data points based on the category they belong to, but also tries to maximize this separation gap between the two classes, i.e., this is a constrained optimization problem. One major advantage of this classifier is that it makes no assumption on the documents to be classified and it endeavors to find the best classification margin for the data at hand instead of relying on probability values. It is one of the widely used machine learning algorithms, which yields very good results for the task of sentiment analysis.

1.1.1 Drawbacks of Existing System

- A major assumption made by the Naive Bayes classifier is that the features are conditionally independent, given the sentiment class of the document, which is not true in real-life situations.
- Furthermore, another problem with this technique is that, if some feature value, which was not encountered in the training data, is seen in the input data, its corresponding probability will be set to 0. Bayes classifier fails in this case.
- Most of the classical machines learning algorithms for text classification are either rule-based or corpus-based. Their efficiency depends on the quality of the annotated corpora as well as the feature engineering task involved prior to the classification. The features need to be manually handcrafted as well as they differ from domain to domain and document to document, which makes it less generic and more text-specific.
- The accuracy of these models depends on how the features were chosen, which makes the system liable.
- It is very difficult, and many a times not feasible, to adapt a system designed for a particular problem to new problems or in different language for the same problem. And for texts like reviews, which do not follow any rules or grammar as such, these approaches tend to perform very badly. Hence, extensive pre-processing and feature engineering needs to be done specific to the text genre, language and the problem statement using other NLP tools. Since these tools are not 100% accurate, the loss in accuracy in the pre-processing steps will in turn affect the overall accuracy of the sentiment analysis task.

1.2 Advantages of Proposed System

Although the traditional machine learning algorithms like SVM have shown good performance in various NLP tasks for the past few decades, they have some shortcomings

and deep learning models have the potential to overcome these limitations to a large extent. Some of the advantages of deep neural networks are:

- The strength of the deep learning models is no demand for carefully optimized hand-crafted features. Instead of the features, they take word embeddings as input which contain context information, and the intermediate layers of the neural network learn the features during the training phase itself. This means that a necessity, which is at the base of the traditional classification models, is no longer required for the functioning of deep learning models.
- Deep learning allows good representation learning. While feature representations of input text can be learned automatically from the training data for a particular task, the representations of the words, containing context information, can be learned from raw corpus in an unsupervised manner. This disregards any need for manual construction of appropriate features or word information.
- Sentiment analysis consists of varied problem statements. The ability to adapt to the task variations with very small changes in the system itself adds a feather in the cap of the deep learning paradigm.
- CNNs are very fast. With a large vocabulary, computing anything more than 3-grams can quickly become expensive. Even Google doesn't provide anything beyond 5-grams. Convolutional Filters learn good representations automatically, without needing to represent the whole vocabulary. It's completely reasonable to have filters of size larger than 5 as many of the learned filters in the first layer are capturing features quite similar to n-grams, but represent them in a more compact way.

Once I became successful in achieving reasonable accuracy in text classification using Convolutional Neural Networks, I designed a '*Movie Reviewer*' website to test the model on real-time data. The website aims to collect the reviews given by its users for different movies, analyze its sentiment using CNN model and then generate bar graphs to make the reviews readable and easy to analyze.

2. LITERATURE REVIEW

Sentiment analysis of online user generated content is important for many social media analytics tasks. In the context of social media, there are several challenges. First, there are huge amounts of data available. Second, messages on social networks are by nature informal and short. Therefore, sentiment analysis is a very challenging task [1][2]. Many different approaches to solve sentiment analysis have been developed by researchers from natural language processing and information retrieval, most of which in this field use bag of words representations [3]. With the development of Internet, we can gain huge amounts of data in different web sites such as social media or online-shopping sites. Therefore, many researchers begin to use this available data to analyze sentiment in the social media data. Snyder and Barzilay [4] analyzed the sentiment of multiple aspects of restaurants like food or atmosphere to deal with larger reviews in more detail. The deep learning framework is very powerful in both supervised and unsupervised learning fields. Because of the recent achievement of deep learning, more and more researchers are trying to solve the challenging sentiment analysis task using deep learning algorithms. A famous deep learning framework provided by Socher [5] is the Recursive Neural Network (RNN). Socher used fully labeled parse trees to represent the movie reviews from the rottentomatoes.com website. Recursive neural models compute parent node vectors by using the two children's node vectors to compute the parent vectors. Finally, the root node of the parse tree can represent the sentence. This is the simplest member of this family of the Recursive Neural Network proposed by Socher. In 2013 [6], Socher proposed an improved Recursive Neural Network called Recursive Neural Tensor Network (RNTN). The main idea is to use the same, tensor-based composition function for all nodes. In other words, they take into account the distance of the word in a sentence by adding a parameter to the models. However, there is a disadvantage in both RNN and RNTN. Both of them need to a fully labeled parse trees to train the neural network, so it will be a lot of heavy work to label each word and group of words. On the contrary, Convolutional Neural Network only needs the labels of the whole sentences instead of the fully labeled parse tree. At the same time, CNN have much fewer connections and parameters and they are easier to train.

Convolutional Neural Network [7] has been proved very effective in solving the tasks related to computer vision. CNN contains the convolution layers and pooling layers. In a general situation,

the convolutional layer is followed by the pooling layers which combine the outputs of neuron clusters [8][9]. However, when given more complex problems, the breadth and depth of CNN will continue to increase which would become limited by computing resources. Recently, thanks to the increasing efficient GPU computing, training a large deep convolutional neural network became possible. In 2012, Dan Ciresan et al. significantly improved upon the best performance in the literature for multiple image databases, including the MNIST database, the NORB database, the HWDB1.0 dataset (Chinese characters), the CIFAR10 dataset (dataset of 60000 32x32 labeled RGB images) [10], and the ImageNet dataset [11]. CNN models have subsequently been shown to be useful for the sentiment analysis problem. N. Kalchbrenner described a Dynamic Convolutional Neural Network (DCNN) [12] for the semantic modeling of sentences. The network used Dynamic k-Max Pooling, a global pooling operation over linear sequences.

In our work, we pre-train a CNN model using the word2vec output vectors in order to represent the distance of each word. The output vectors are then fed as input to the CNN model. This is followed by a max-pooling layer which identifies the words which hold more weight in determining the sentiment of the entire text. We tested our proposed system using user opinions about different reviews about movies. The test results show that our proposed system will be helpful to analyze and visualize the opinion of users.

3. OBJECTIVES

1. To implement Convolutional neural networks for sentiment analysis of movie reviews using the data set from *www.rottentomatoes.com* [13] and achieve high accuracy.
2. To implement this model in an interactive web interface to analyze real-time movie-review data.

4. REQUIREMENT ANALYSIS

4.1. Software Requirements

- *High Level Language:* Python3
- *Libraries:* TensorFlow, NumPy, re, Libchart
- *Web interface language:* PHP, HTML/CSS, JavaScript
- *Local Server:* WAMP Server 2.0
- *Tools:* JetBrains Pycharm Community Edition 2013, IDLE Python, Google word2vec
- *Operating System:* Windows XP or above.

4.2. Hardware Requirements

Table 1 Hardware Requirements

<i>H/W Component</i>	<i>Minimum</i>	<i>Recommendation</i>
Processor	133MHz	500MHz
RAM	32 MB	128 MB
Free Hard Disk Space	30 MB	50 MB
Monitor	VGA	SVGA

4.3. Functional Requirements

Functional requirements defining this project are as follows:

- Real-Time Processing: The software will take input, process data, and display output in real-time.
- Sentiment Analysis: Sentiment analysis will be performed on the keywords within the input text to determine the overall feedback (negative/positive) relative to the movie.
- Output: The software must output data in the form of simple charts and histograms. This output will be clear and easy to understand.

4.4. Non-Functional Requirements

Following are the parameters we will use, to evaluate our implemented model:

- Performance Measures: Python3 will provide prompt perform analysis of the data using the various software packages available in it. The output should display the latest results at all times, and if it lags behind, the user should be notified. The website should be capable of operating in the background should the user wish to utilize other applications.
- Reliability: The software will meet all of the functional requirements without any unexpected behavior. At no time should the output display incorrect or outdated information without alerting the user to potential errors. In this instance error message will be shown. As the training set becomes larger, the accuracy will tend to 100%.
- Availability: The software will be available at all times on the user's device (desktop or laptop), as long as the device is in proper working order. The functionality of the software will depend on any external services such as internet access that are required. If those services are unavailable, the user should be alerted.
- Security: The software should never disclose any personal information of users, and should collect no personal information from its users.

5. METHODOLOGY

In this section research methodology which will be adopted to carry-out the project work is discussed. Every step of process is briefly shown in the following figure:

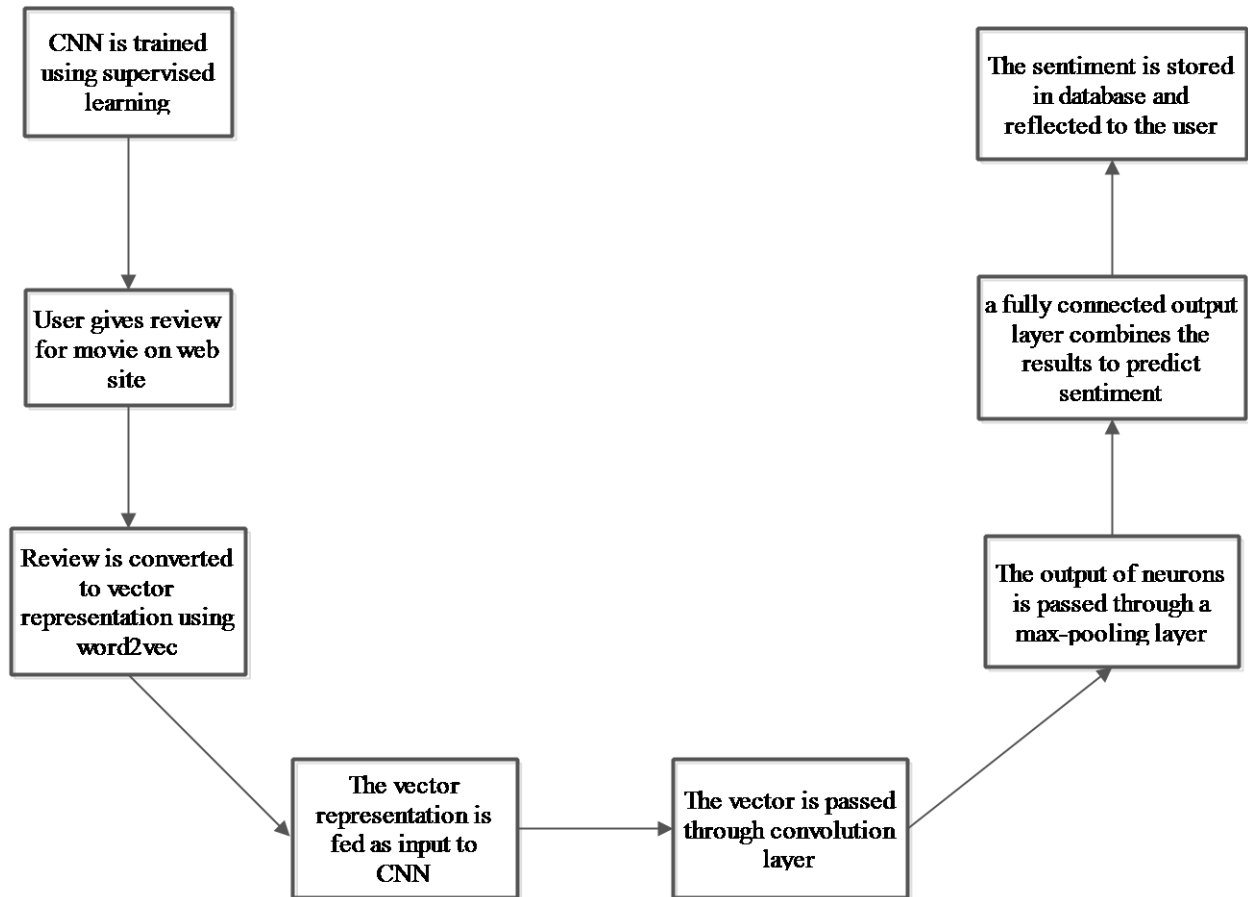


Figure 1 Project Flow

The steps of sentiment analysis are:

STEP 1: Training the CNN model using unsupervised learning. The CNN model is trained on a training data set consisting of 10662 reviews.

STEP 2: After training is complete, the model is tested and is ready to use to predict the sentiment of previously unseen data.

STEP 3: The user feedback for a movie is collected when the user enters his/her review on the website.

STEP 4: This feedback is converted into a vector representation by using word2vec provided by Google.

STEP 5: In the next step, this vector representation is fed as input to CNN.

STEP 6: The input feedback passes through a number of CNN and pooling layers, parameters are tuned and the different words in the feedback are analyzed for sentiment.

STEP 7: Finally, a fully connected output layer is constructed to combine the results of all layers and predict the final sentiment of the whole sentence.

STEP 8: This sentiment is then stored in the database corresponding to the movie and it is also reflected back to the user.

For the purpose of sentiment, we have used the Convolutional neural network model. We have used the word2vec to gain the vectors for the words as the input. Word2vec is a neural net that processes text before that text is handled by deep-learning algorithms. In this problem, what we wanted to classify the sentences with CNN, but CNN can't understand the sentences directly as a human. To deal with it, word2vec translates text into the vector that CNN can understand. At the same time, the vector produced by the word2vec can represent the distance of words. When two words' meaning is similar, the vectors' value of them is closed too. A string of sentences is expected by word2vec as its input. Each sentence, which is each array of words, is transformed to vectors and compared to other vectors' lists of words in an n-dimensional vector space. Related words or groups of words appear next to each other in that space. When we gain the vectors of words, it allows us to measure their similarities with some exactness and cluster them. CNN will have a better performance because of it.

After the sentence is converted into its vector representation, the vector is fed as input to the Convolutional neural network in the input layer. The CNN learns the values of its filters and uses an activation function, like ReLU or tanh, to combine the values of the filters with the sentence

matrix. After that, pooling layer is applied to form uni-variate feature vectors, which are then classified into suitable classes by a softmax function at the output layer.

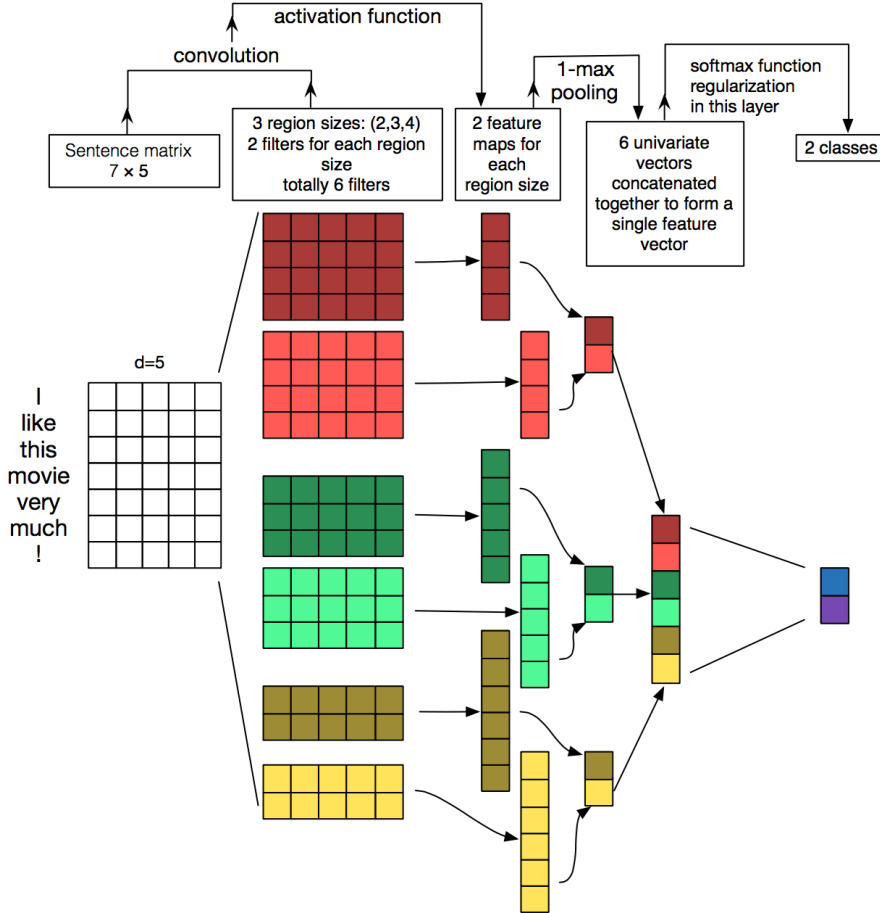


Figure 2 Architecture of CNN used in our work

In figure 2, three filter region sizes: 2, 3 and 4 are used, each of which has 2 filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a uni-variate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states.

5.1. CNN Hyper-parameters

5.1.1. Narrow vs. Wide convolution

This parameter defines how the convolution is applied to different parts of the input sentence. The input text is given as a matrix to the Convolutional Neural Network. Applying a 3×3 filter (or $n \times n$, in general) at the center of the matrix works properly, but not at the edges. The problem which arises is that we can not apply the filter to the first element of a matrix that doesn't have any neighboring elements to the top and left. This problem is eliminated by using *zero-padding*. Using this technique, all elements that would fall outside of the matrix are taken to be zero. By doing this, the filter can be applied to every element of the input matrix, and we can get a larger or equally sized output. Adding zero-padding is also called *wide convolution*, and not using zero-padding would be a *narrow convolution*.

Wide convolution is useful, or even necessary, when the filter size is large relative to the input size. the formula for the output size is:

$$n_{out} = (n_{in} + 2 * n_{padding} - n_{filter}) + 1.$$

where n_{out} is the output size, n_{in} is the input size, $n_{padding}$ is the padding size, and n_{filter} is the filter size.

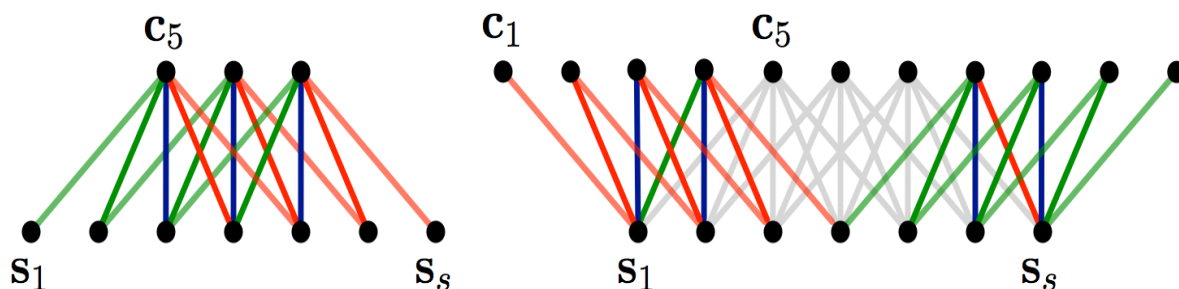


Figure 3 Narrow convolution (left) vs. wide convolution (right)

In the above example, the narrow convolution yields an output of size, $(7 - 5) + 1 = 3$; and a wide convolution yields an output of size $(7 + 2*4 - 5) + 1 = 11$.

5.1.2. Stride Size

Another hyper parameter for convolution neural network is the *stride size*, which defines by how much the filter will be shifted at each step. When the stride size is taken to be 1, the consecutive applications of the filter overlap. A larger stride size leads to fewer applications of the filter and a smaller output size. The following figure shows stride sizes of 1 and 2 applied to a one-dimensional input:

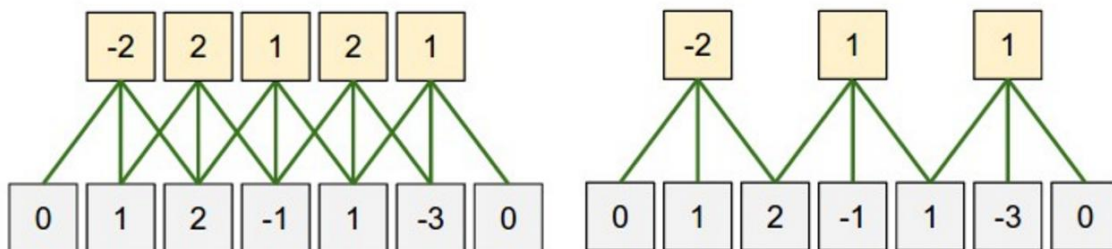


Figure 4 Stride size of 1 (left) and Stride size of 2 (right)

5.1.3. Pooling Layers

Pooling layers are typically applied after the convolutional layers in CNN. Pooling layers subsample their input. The most common way to do pooling is to apply a *max* operation to the result of each filter. It is not necessary to pool over the complete matrix, we can also pool over a window. For example, the following shows max pooling for a 2×2 window (in NLP we typically apply pooling over the complete output, yielding just a single number for each filter):

Pooling is necessary and advantageous for many reasons. Some of these are:

- Pooling provides a fixed size output matrix, which typically is required for classification. For example, if we have 1,000 filters and we apply max pooling to each, we will get a 1000-dimensional output, regardless of the size of the filters,

or the size of the input. This allows us to use variable size sentences, and variable size filters, but always get the same output dimensions to feed into a classifier.

- Pooling also reduces the output dimensionality but keeps the most salient information. Each filter detects a specific feature, such as detecting if the sentence contains a negation like “not amazing” for example. If this phrase occurs somewhere in the sentence, the result of applying the filter to that region will yield a large value, but a small value in other regions. By performing the max operation using pooling, we keep information about whether or not the feature appeared in the sentence, but we lose information about where exactly it appeared. In this way, we lose global information about locality (where in a sentence something happens), but we keep local information captured by the filters, like “not amazing” being very different from “amazing not”.
- In image recognition, pooling also provides basic invariance to translating (shifting) and rotation. When pooling over a region, the output will stay approximately the same even if the image is shifted/rotated by a few pixels, because the max operations will pick out the same value regardless.

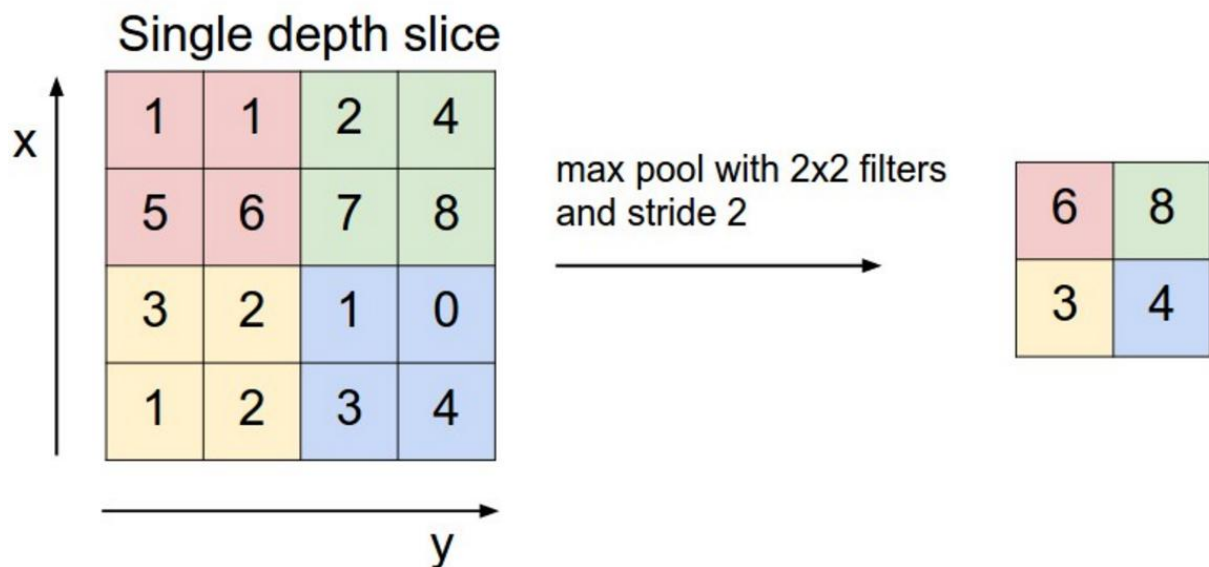


Figure 5 *max* pooling in CNN

6. SYSTEM DESIGN

In our work, we have used the word2vec to gain the vectors for the words as the input. Word2vec is a neural net that processes text before that text is handled by deep-learning algorithms. In this problem, what we wanted to classify the sentences with CNN, but CNN can't understand the sentences directly as a human. To deal with it, word2vec translates text into the vector that CNN can understand. At the same time, the vector produced by the word2vec can represent the distance of words. When two words' meaning is similar, the vectors' value of them is closed too. A string of sentences is expected by word2vec as its input. Each sentence, which is each array of words, is transformed to vectors and compared to other vectors' lists of words in an n-dimensional vector space. Related words or groups of words appear next to each other in that space. When we gain the vectors of words, it allows us to measure their similarities with some exactitude and cluster them. CNN will have a better performance because of it.

After we get the pre-trained word vectors from word2vec, we will train a convolutional neural network. In the architecture of CNN, the most time of training the neural network is spent in the convolution. The CNN learns the values of its filters and uses an activation function, like ReLU or tanh, to combine the values of the filters with the sentence matrix. After that, pooling layer is applied to form uni-variate feature vectors, which are then classified into suitable classes by a softmax function at the output layer. The different functions used in the system are:

- A word2vec function to convert a sentence into corresponding sentence matrix.
- A convolution function to convolve the filters over the input matrix.
- An activation function to combine the filters with the sentence matrix.
- A pooling function to concatenate a feature map into a uni-variate feature vector.
- A dropout function to prevent the model from over fitting the data.
- A softmax function to combine the values of output layer neurons to give a value in the range [0,1].
- A classification function to predict the class label based on the output of softmax function.
- An exec function to integrate Python with php in a web interface.

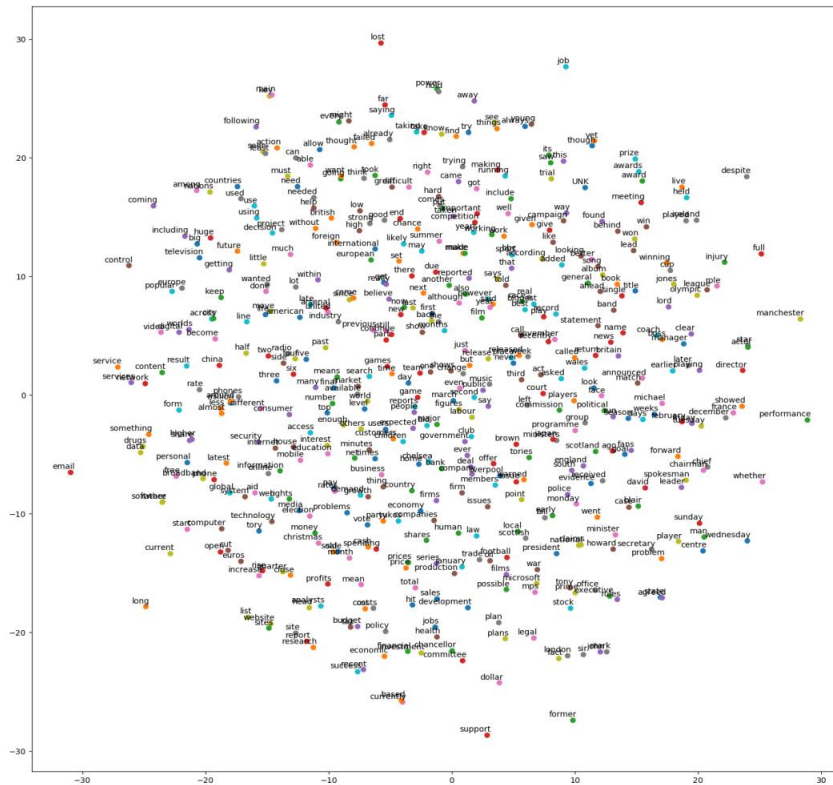


Figure 6 Word2Vec output

6.1. Data Flow Diagrams

The user-interface of our project allows users to post feedback in real-time and the ‘*Movie Reviewer*’ would classify the entered feedback to be of positive or negative sentiment. The complete process is depicted in the following Data Flow Diagrams:



Figure 7 Level 0 DFD

The user interface requires a new user to first sign up. In case the user has already signed up, he can directly log in to use the features of the website. The website allows the users to either review a movie or check out all the reviews given to a movie. For this, the user may select from movies which are provided in the portal. The entire process and data flow is as shown in Figure 8.

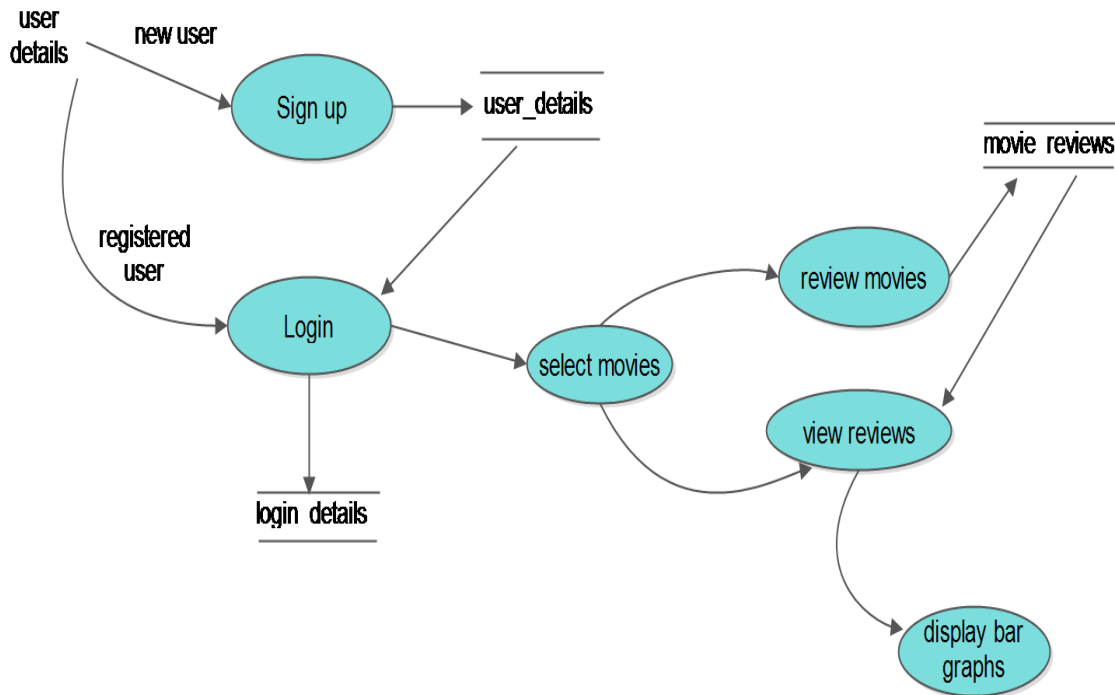


Figure 8 Level 1 DFD

6.2. UML Diagrams

6.2.1. Use Case Diagram

The users of the system are divided into three categories as: registered users, visitors and administrator. The role and functions of each user is depicted in the following use-case diagram:

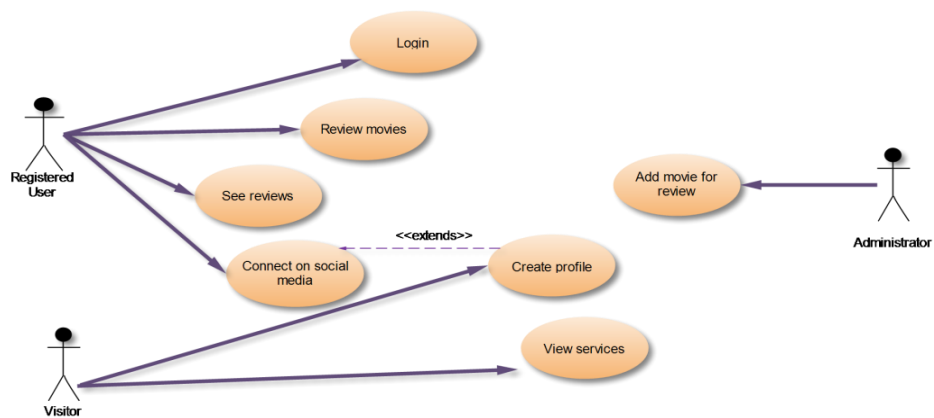


Figure 9 Use-Case Diagram

6.2. Sequence Diagram

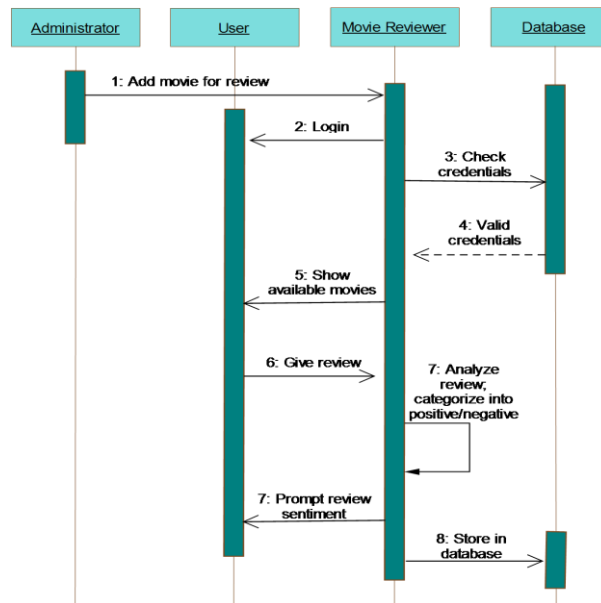


Figure 10 Sequence Diagram for giving reviews

Figure 10 shows the sequence of steps followed by the system when a registered user attempts to give feedback for a particular movie. The sequence of steps followed when a user attempts to view the movie reviews, is given by the following sequence diagram:

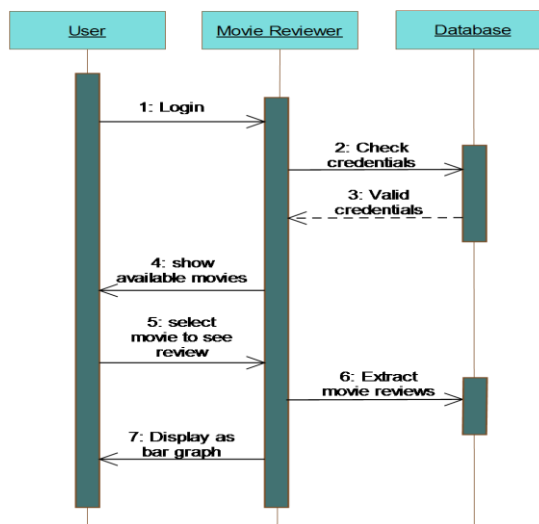


Figure 11 Sequence diagram for viewing reviews

6.3. Class Diagram

The following diagram depicts the different classes used in the project along with the functions that each class performs.

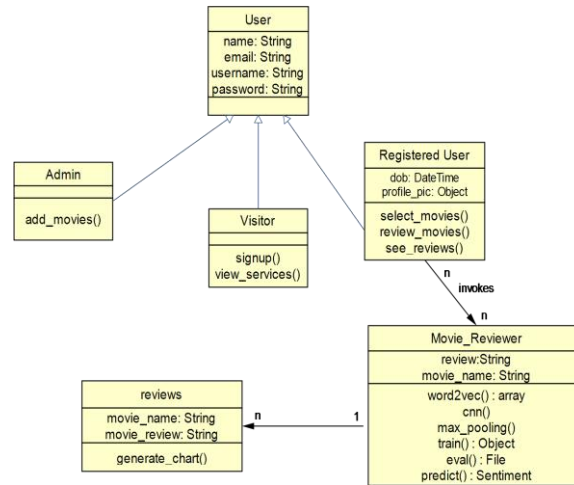


Figure 12 Class Diagram for movie reviewer

The users of the system are divided into three categories, viz., admin, visitor and registered user. The functions available to each user are as follows:

- **Admin:** The admin has the sole responsibility of adding movies for reviews into the website. However, the admin cannot give reviews from his admin portal to preserve the authenticity of reviews.
- **Visitor:** The visitor has the option to view the services provided by the website and to sign up in the website by giving his personal details.
- **Registered user:** The registered user is given the ability to review a movie and to see all reviews for different movies in the form of graphs. But for that, he has to first select a movie from a list of available movies. Additionally, the each user can review a particular movie only once. This has been done to preserve the authenticity of reviews.

The functions of the movie reviewer can only be invoked by the registered user. The movie reviewer runs the CNN model to predict the review sentiment and returns it back to the user.

7. IMPLEMENTATION

I worked on the movie review dataset available on the site www.rottentomatoes.com [13]. There are 10662 sentences of movie reviews, half of which were positive and the other half negative in the original dataset. The overall sentiment of the long movie review can be reflected by the label. First, we used word2vec to gain the vectors for the words as the input. In this problem, what I wanted to do was to classify the sentences with CNN, but CNN can't understand the sentences directly as a human. To deal with it, word2vec translates text into the vector that CNN can understand. At the same time, the vector produced by the word2vec can represent the distance of words. When two words' meaning is similar, the vectors' value of them is closed too. A string of sentences is expected by word2vec as its input. Each sentence, which is each array of words, is transformed to vectors and compared to other vectors' lists of words in an n-dimensional vector space. Related words or groups of words appear next to each other in that space. When we gain the vectors of words, it allows us to measure their similarities with some exactitude and cluster them. This output vector is then given as input to CNN model to train it.

The training program took about 6-7 hours for execution which forced us not to use a larger training set due to performance constraints of the available hardware. After training, the accuracy of the model was noted:

```
Parameters:
ALLOW_SOFT_PLACEMENT=True
BATCH_SIZE=64
CHECKPOINT_EVERY=100
DEV_SAMPLE_PERCENTAGE=0.1
DROPOUT_KEEP_PROB=0.5
EMBEDDING_DIM=128
EVALUATE_EVERY=100
FILTER_SIZES=3,4,5
L2_REG_LAMBDA=0.0
LOG_DEVICE_PLACEMENT=False
NEGATIVE_DATA_FILE=./data/rt-polaritydata/rt-polarity.neg
NUM_CHECKPOINTS=5
NUM_EPOCHS=200
NUM_FILTERS=128
POSITIVE_DATA_FILE=./data/rt-polaritydata/rt-polarity.pos

Loading data...
Vocabulary Size: 18758
Train/Dev split: 9596/1066
```

Figure 13 Training parameters

2017-12-05T20:24:30.411643: step 1, loss 3.77101, acc 0.515625	2017-12-06T02:30:25.652841: step 29981, loss 0, acc 1
2017-12-05T20:24:30.864793: step 2, loss 2.68104, acc 0.4375	2017-12-06T02:30:26.105987: step 29982, loss 0, acc 1
2017-12-05T20:24:31.317940: step 3, loss 2.13525, acc 0.546875	2017-12-06T02:30:26.543510: step 29983, loss 2.23517e-08, acc 1
2017-12-05T20:24:31.817963: step 4, loss 2.51492, acc 0.46875	2017-12-06T02:30:26.981030: step 29984, loss 3.35276e-08, acc 1
2017-12-05T20:24:32.302363: step 5, loss 2.0336, acc 0.578125	2017-12-06T02:30:27.418554: step 29985, loss 1.86265e-09, acc 1
2017-12-05T20:24:32.755510: step 6, loss 2.01413, acc 0.5625	2017-12-06T02:30:27.856076: step 29986, loss 1.18739e-05, acc 1
2017-12-05T20:24:33.208660: step 7, loss 2.32772, acc 0.59375	2017-12-06T02:30:28.324848: step 29987, loss 5.16252e-06, acc 1
2017-12-05T20:24:33.677432: step 8, loss 2.83083, acc 0.484375	2017-12-06T02:30:28.762372: step 29988, loss 6.23974e-07, acc 1
2017-12-05T20:24:34.130580: step 9, loss 2.59777, acc 0.5	2017-12-06T02:30:29.215519: step 29989, loss 0.00101908, acc 1
2017-12-05T20:24:34.599353: step 10, loss 1.70787, acc 0.515625	2017-12-06T02:30:29.668665: step 29990, loss 2.04891e-08, acc 1
2017-12-05T20:24:35.083755: step 11, loss 1.67392, acc 0.484375	2017-12-06T02:30:30.121813: step 29991, loss 1.15869e-05, acc 1
2017-12-05T20:24:35.552526: step 12, loss 2.02626, acc 0.5	2017-12-06T02:30:30.574962: step 29992, loss 3.32819e-06, acc 1
2017-12-05T20:24:36.036927: step 13, loss 1.71039, acc 0.515625	2017-12-06T02:30:31.012483: step 29993, loss 0, acc 1
2017-12-05T20:24:36.505699: step 14, loss 1.99172, acc 0.484375	2017-12-06T02:30:31.465633: step 29994, loss 0, acc 1
2017-12-05T20:24:37.005726: step 15, loss 2.21071, acc 0.5625	2017-12-06T02:30:31.918779: step 29995, loss 1.30385e-08, acc 1
2017-12-05T20:24:37.474497: step 16, loss 2.00714, acc 0.515625	2017-12-06T02:30:32.371927: step 29996, loss 0, acc 1
2017-12-05T20:24:37.943273: step 17, loss 1.96962, acc 0.484375	2017-12-06T02:30:32.809448: step 29997, loss 0.000464568, acc 1
2017-12-05T20:24:38.412044: step 18, loss 2.28015, acc 0.375	2017-12-06T02:30:33.262598: step 29998, loss 1.30385e-08, acc 1
2017-12-05T20:24:38.865194: step 19, loss 2.01488, acc 0.484375	2017-12-06T02:30:33.715744: step 29999, loss 2.42144e-08, acc 1
2017-12-05T20:24:39.365219: step 20, loss 1.85728, acc 0.46875	2017-12-06T02:30:34.137640: step 30000, loss 7.94728e-09, acc 1
2017-12-05T20:24:39.865245: step 21, loss 1.38274, acc 0.609375	
2017-12-05T20:24:40.365269: step 22, loss 2.03707, acc 0.453125	
2017-12-05T20:24:40.865294: step 23, loss 2.129, acc 0.484375	
2017-12-05T20:24:41.334067: step 24, loss 2.19748, acc 0.46875	
2017-12-05T20:24:41.849719: step 25, loss 1.79317, acc 0.53125	

Evaluation:	2017-12-06T02:30:35.528335: step 30000, loss 9.08451, acc 0.726079
-------------	--

Figure 14 Training process and accuracy

After the training phase is complete, testing of model is done to determine the accuracy of the model, i.e., the accuracy with which the model will be able to predict new and previously unseen data.

```

Parameters:
ALLOW_SOFT_PLACEMENT=True
BATCH_SIZE=64
CHECKPOINT_DIR=C:\Users\sony\Desktop\cnn-text-clas
EVAL_TRAIN=False
LOG_DEVICE_PLACEMENT=False
NEGATIVE_DATA_FILE=C:\Users\sony\Desktop\cnn-text-
POSITIVE_DATA_FILE=C:\Users\sony\Desktop\cnn-text-

C:\Users\sony\Desktop\cnn-text-classification-tf-m
[[ 1  2  3 ..., 0  0  0]
 [ 1 31 32 ..., 0  0  0]
 [ 57 58 59 ..., 0  0  0]
 ...,
 [ 75 84 1949 ..., 0  0  0]
 [ 1 2191 2690 ..., 0  0  0]
 [11512 3 147 ..., 0  0  0]]

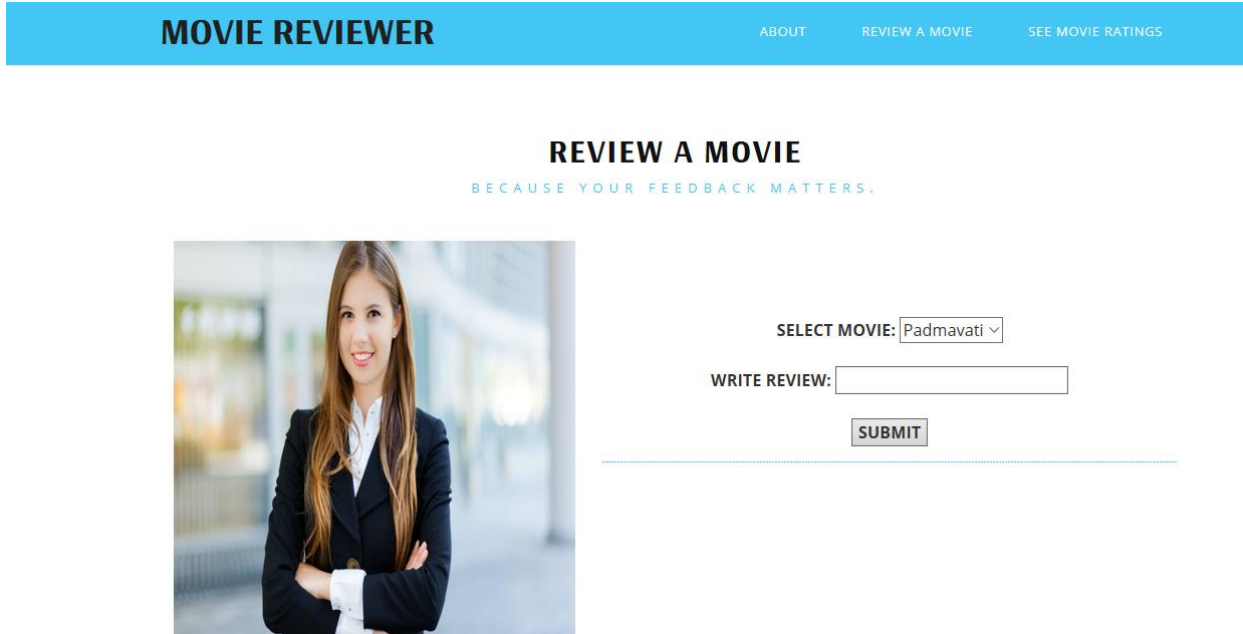
Evaluating...

[ 1.  1.  1. ..., 0.  0.  0.]
Total number of test examples: 10662
Accuracy: 0.972894

```

Figure 15 Testing parameters and accuracy

After testing the model, a user interface was designed to enable real-time classification of user data. The user-interface was designed in HTML using CSS, JavaScript and CSS and php was used to integrate the web page with Python3 and the Database.



MOVIE REVIEWER

ABOUT REVIEW A MOVIE SEE MOVIE RATINGS

REVIEW A MOVIE

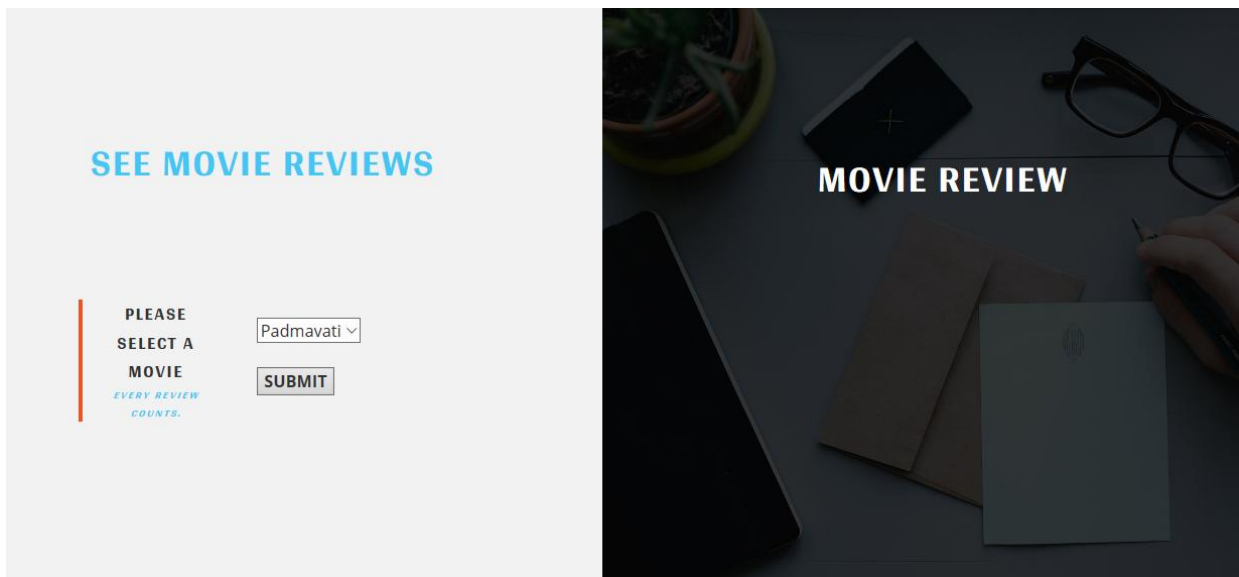
BECAUSE YOUR FEEDBACK MATTERS.

SELECT MOVIE: Padmavati

WRITE REVIEW:

SUBMIT

Figure 16 User screen for giving movie feedback



SEE MOVIE REVIEWS

PLEASE SELECT A MOVIE

Padmavati

SUBMIT

MOVIE REVIEW

Figure 17 User screen for viewing all movie feedbacks

8. SUBJECT SYSTEM

I have implemented the CNN model in Python 3 using JetBrains Pycharm as editor. The model was run in the editor and its accuracy and other parameters were noted in the editor window.

Then the model was integrated into a web interface with php on Wamp Server 2.0 and the real-time analysis for the sentiment of review data was done.

9. OBSERVATIONS AND FINDINGS

When the user will enter any sentence in the textbox, the sentiment of the sentence is analyzed and returned to the user in real-time. At the same time, the movie along with the sentiment score is stored in the database for further analysis.

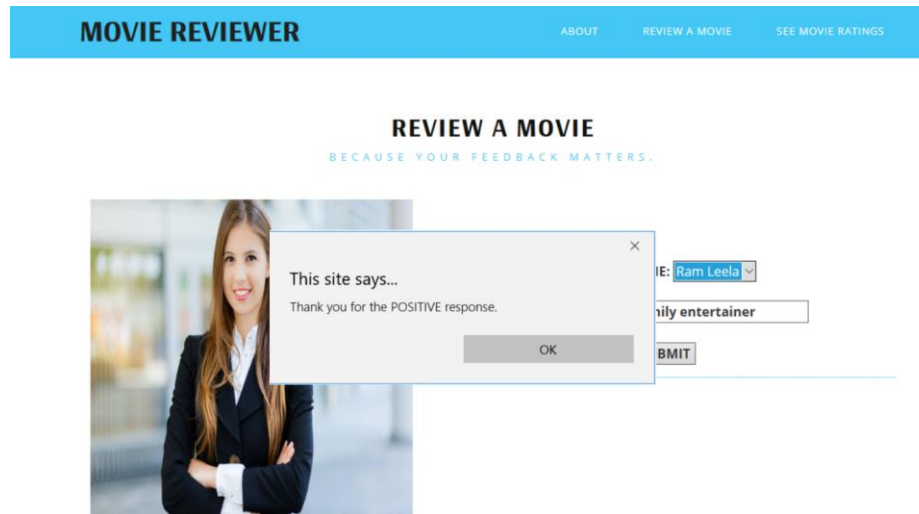


Figure 18 Response of feedback reflected to user

The user can also check out the reviews given by other users on the website in the form of bar graphs. For this, there is a separate functionality included in the application. The user has to go to the 'See Movie Ratings' tab to see movie reviews. Then the user selects the movie for which he wants to see reviews out of all available movies. After the user submits his choice, a bar graph depicting the overall ratings is displayed on the user screen.

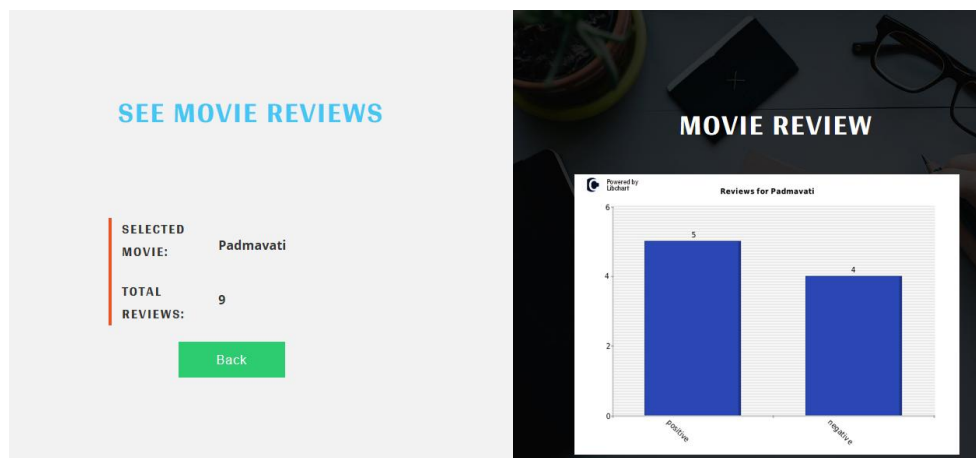


Figure 19 Bar graphs generated from movie reviews

I simulated the real-time analysis 100 times with random sentences and obtained the following results:

Table 2 Simulation Results

Classes	True positive	False Positive	Recall	Precision	F-measure
Positive	0.84	0.19	0.86	0.84	0.85
Negative	0.81	0.16	0.79	0.81	0.80

10. APPLICATIONS AND LIMITATIONS

10.1. Applications

- The CCN model which is implemented in the project can work on any data set without any modifications. In this way, the CNN model can predict the sentiment of any type of data and not only the movie review data, by changing the data set.
- The interactive user-interface is integrated with python code to make the real-time analysis of sentiment with CNN possible.
- The '*Movie Reviewer*' website allows users to give their feedback in text, which enables the users to express their emotions subjectively without any barriers.
- The '*Movie Reviewer*' allows the users to see the overall reviews of different movies to form their opinion and decide whether to watch it or not, based upon the reviews.

10.2. Limitations and Boundaries

- As the CNN model uses multiple convolution and pooling layers, the time taken by the system to analyze the system is a bit high.
- The CNN model is a heavy model in terms of CPU utilization and can slow down other processes while it is running, if the system has not enough GPU resources.
- The '*Movie reviewer*' website will give poor accuracy results if reviews, unrelated to movies are added because the model has been trained on the movie dataset only.

11. CONCLUSION AND FUTURE WORK

The task of sentiment analysis is still in the developing stage and far from complete. So we proposed a CNN model to improve the accuracy of this task with in performance limitations of the existing hardware.

CNNs are very fast. With a large vocabulary, computing anything more than 3-grams can quickly become expensive. Even Google doesn't provide anything beyond 5-grams. Convolutional Filters learn good representations automatically, without needing to represent the whole vocabulary. It's completely reasonable to have filters of size larger than 5 as many of the learned filters in the first layer are capturing features quite similar to n-grams, but represent them in a more compact way.

One more feature that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Although some studies have explored a similar feature and reported negative results, their results were based on reviews which are very different from tweets and they worked on an extremely simple model.

We were able to obtain a test accuracy of 0.876 from 1062 test cases. This accuracy can be further improved by using a larger data set. Applying this model to IOV scenario, the results obtained show that social relationships score has a strong impact on the data dissemination. The impact of social score was analyzed and results were obtained. The results depict that the data disseminated increases with an increase in social score.

Although, CNNs show considerable accuracy in predicting the sentiment of unseen data, the accuracy can still be improved further by using larger data sets provided there is availability of more powerful GPUs.

In the future, this project can be improved by using a larger training data set and also the emoticons can be analyzed and included in the determination of sentiment score.

REFERENCES

- [1] Krizhevsky A, Sutskever I, Hinton G E, “Imagenet classification with deep convolutional neural networks,” In Neural Information Processing Systems, pp. 1097-1105, 2012.
- [2] Mikolov T, Chen K, Corrado G, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.
- [3] Mikolov T, Sutskever I, Chen K, “Distributed representations of words and phrases and their compositionality,” in Neural Information Processing Systems, pp. 3111-3119, 2013.
- [4] Mikolov T, YihW, Zweig G, “Linguistic Regularities in Continuous Space Word Representations,” In HLT-NAACL, pp. 746-751, 2013.
- [5] Jia Y, Shelhamer E, Donahue J, “Caffe: Convolutional architecture for fast feature embedding,” in Proceedings of the ACM International Conference on Multimedia. ACM, pp. 675-678, 2014.
- [6] Liu B, Dai Y, Li X, “Building text classifiers using positive and unlabeled examples,” in Data Mining (ICDM), 2003. IEEE 3th International Conference on. IEEE, pp. 179-186, 2003.
- [7] Li G, Hoi S C H, Chang K, “Micro-blogging sentiment detection by collaborative online learning,” in Data Mining (ICDM), 2010. IEEE 10th International Conference.
- [8] Pang B, Lee L, “Opinion mining and sentiment analysis,” in Foundations and Trends in Information Retrieval, pp. 1-135, 2008.
- [9] Snyder B, Barzilay R, “Multiple Aspect Ranking Using the Good Grief Algorithm,” in HLT-NAACL, pp. 300-307, 2007.
- [10] Socher R, Lin C C, Manning C, “Parsing natural scenes and natural language with recursive neural networks,” in Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 129- 136, 2011.
- [11] Socher R, Perelygin A, Wu J Y, “Recursive deep models for semantic compositionality over a sentiment treebank,” In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1631-1642, 2013.
- [12] LeCun Y, Bottou L, Bengio Y, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
- [13] Polarity dataset v2.0 Introduced in Pang/Lee ACL 2004. Released June 2004.

APPENDIX-A DATA DICTIONARY

1. Parameter - A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data.

- They are required by the model when making predictions.
- They are estimated or learned from data.
- They are often not set manually by the practitioner.
- They are often saved as part of the learned model.

2. Hyper parameters - A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data.

- They are often used in processes to help estimate model parameters.
- They are often specified by the practitioner.

3. Confusion matrix - a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.

Table 1 Confusion Matrix format

	Predicted Positive	Predicted Negative
Actual Positive	Tp	Fn
Actual Negative	Fp	tn

- $Precision (P) = tp / (tp + fp)$
- $Accuracy (A) = (tp + tn) / (tp + tn + fp + fn)$
- $Recall (R) = tp / (tp + fn)$
- $True Rate (T) = tp / (tp + fn)$
- $False Rate (F) = fp / (tp + fn)$
- $F = 2.P.R / (P + R)$

APPENDIX-B UML DIAGRAMS

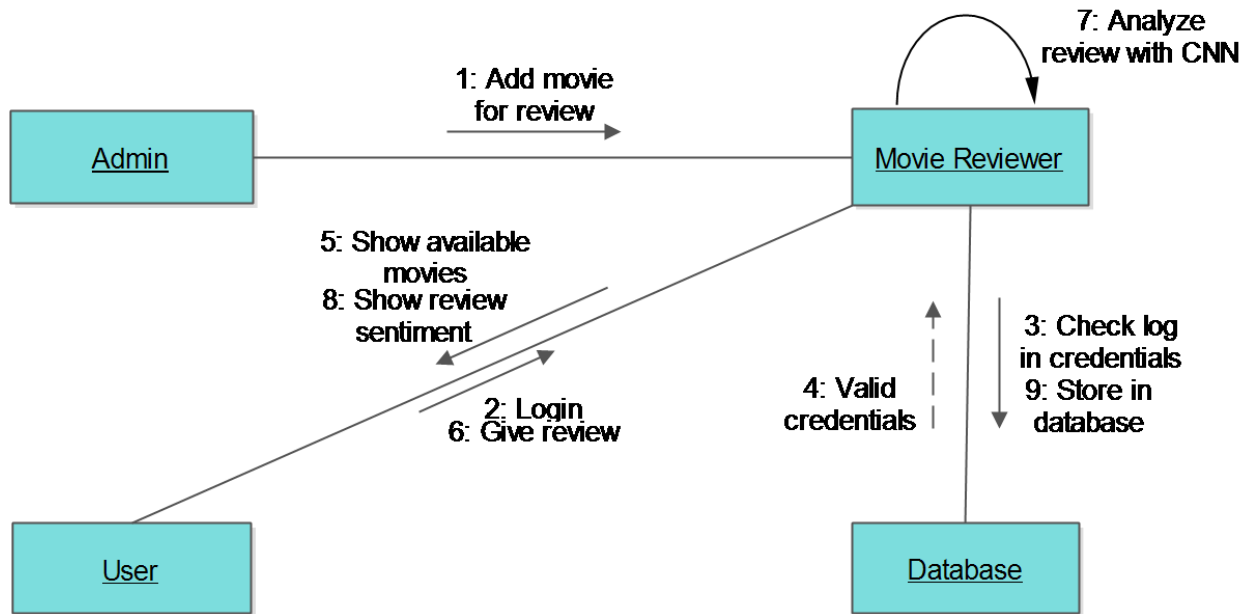


Figure 1 Collaboration Diagram for adding review

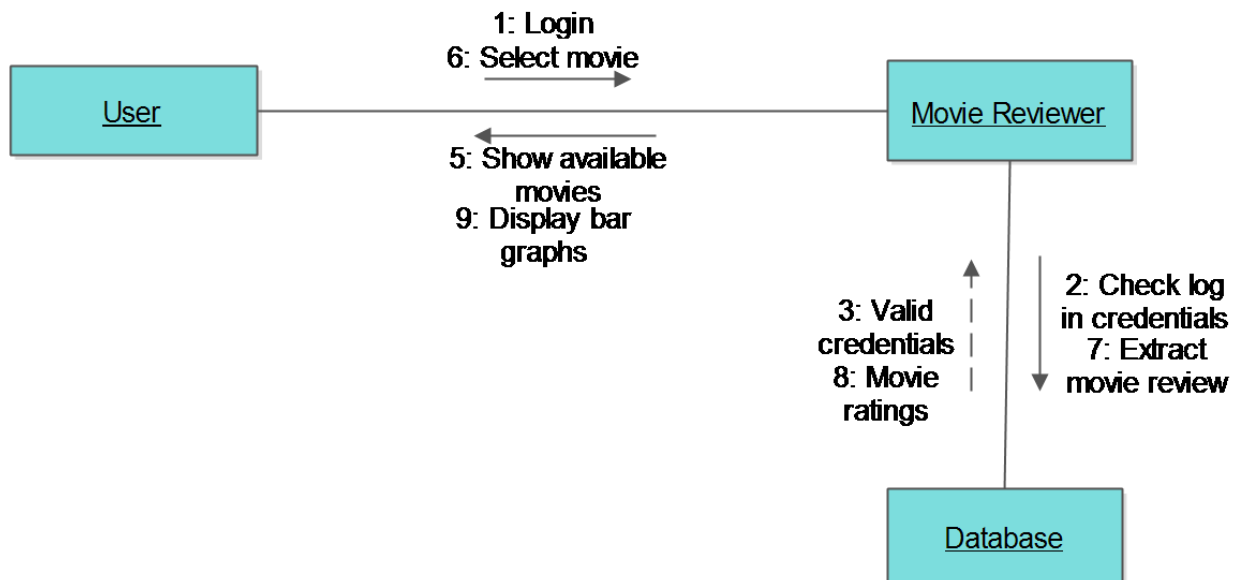


Figure 2 Collaboration Diagram for viewing reviews

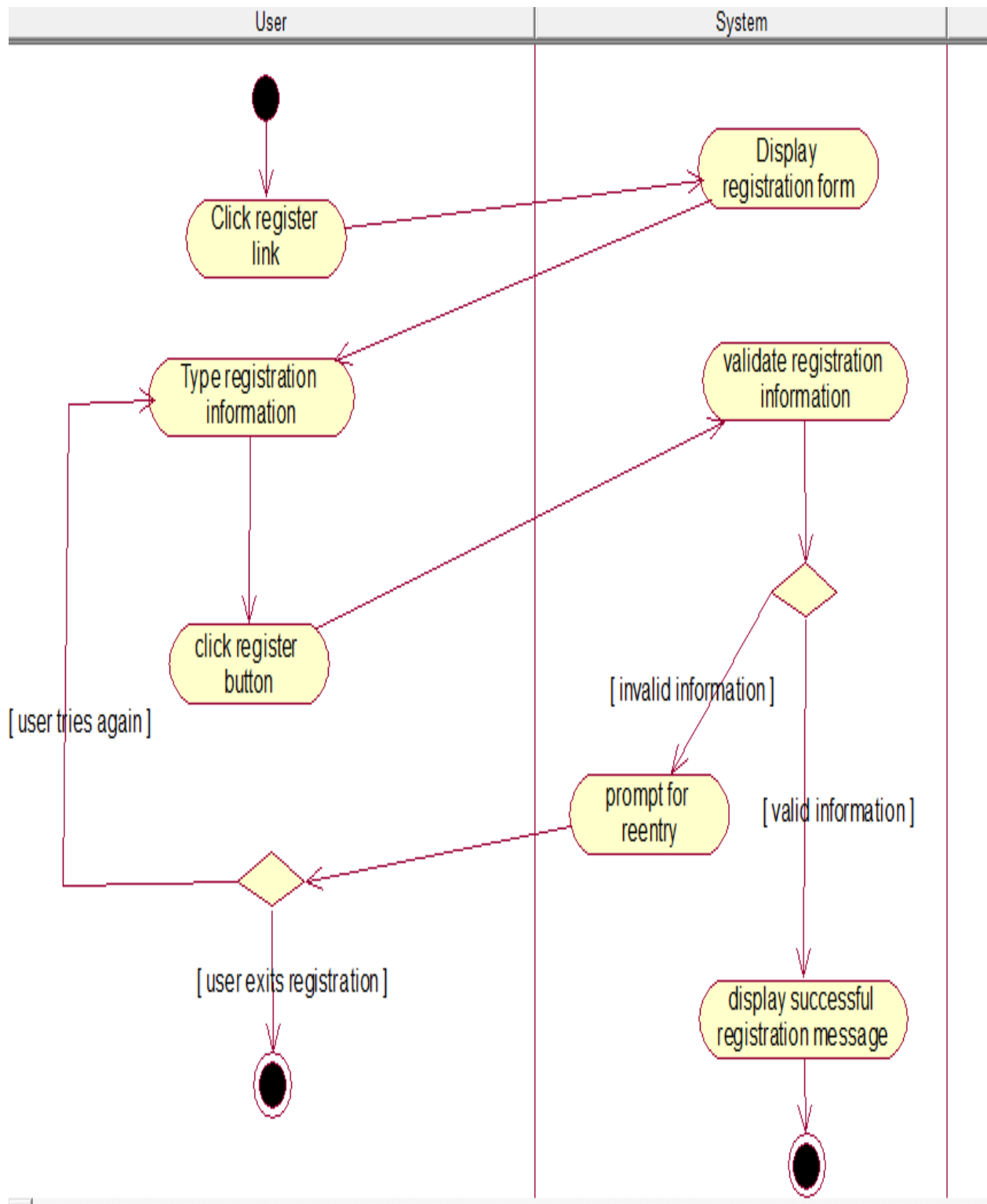


Figure 3 Activity Diagram for signup

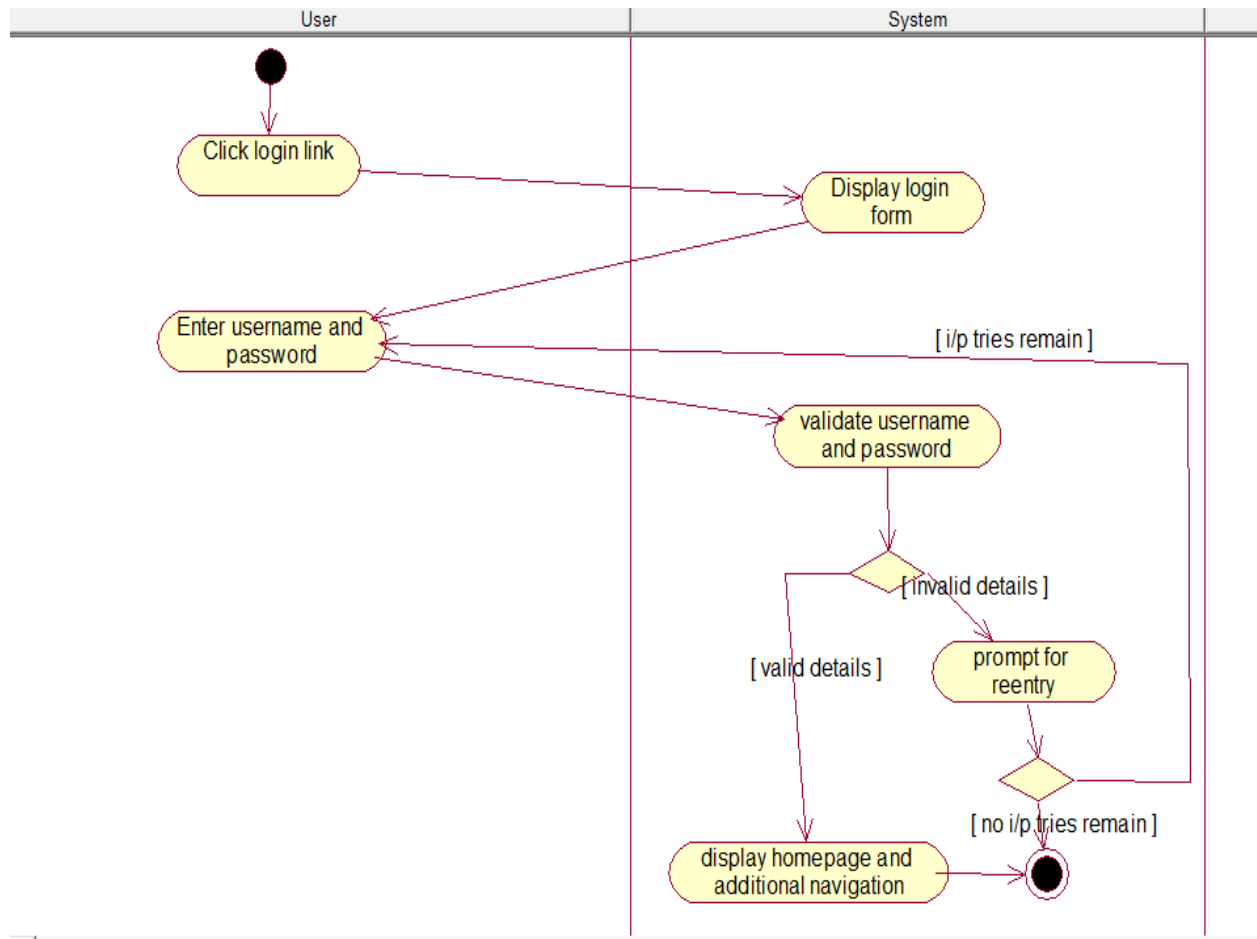


Figure 4 Activity Diagram for log in

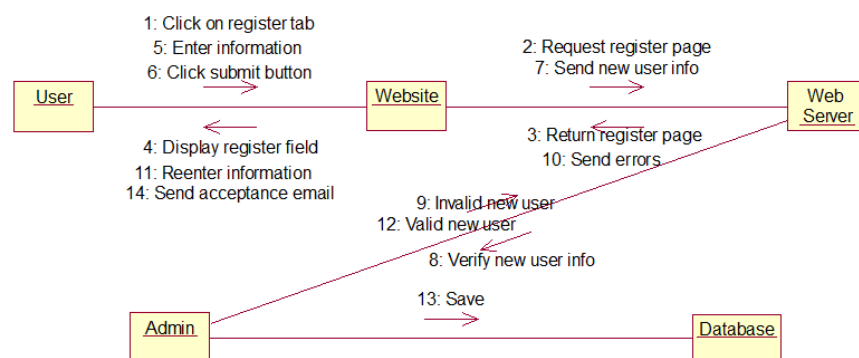


Figure 5 Collaboration diagram for registration

Peer Review of Capstone Project

“Deep-Learning Based Sentiment Analysis of Movie-Review Data”

By peer – Neha Jain

This project, developed by Amuleen Gulati, is basically the implementation of a Deep-Learning based model- Convolutional Neural Network. Deep learning is an emerging research field and works are being carried out to implement its applications. I have heard about CNN In image recognition field but she has implemented it in the field of NLP which I found new and interesting. Based on top of that, she has developed a web application named ‘Movie Reviewer’ which integrates the CNN model with the web. This allows the real-time analysis of a feedback sentiment. To use the application, I had to first create a profile by filling the sign up form and then I was directed to my personalized page on the application. On this page, I had the option of reviewing any of the multiple movies for which I had a choice. When I submitted my review, the application prompted an alert, telling me the sentiment of the review which I had written.

It is quite a useful application as many times we are not able to depict our exact sentiments objectively. The star ratings which we give to movies may be erroneous as we may not be able to justify our ratings with our sentiments. This application automatically analyzes our hidden sentiments from the use and nature of our feedback which is a plus point.

In addition, I was also given the option of viewing the user feedbacks of different movies. When I selected a movie to see its feedback, I was shown a bar graph depicting the number of positive and negative reviews for the movie that I had selected. I found this generation of feedback graph quite useful as it gave a clear picture of the liking and disliking of people for that movie. Given such a pictorial view, I can easily decide whether I should spend my money on the movie or not.

CRITIQUE:

Overall, I found the application very user-friendly and useful. As a user of the application, I would really enjoy the features of the application for my benefit. But the only problem that I encountered while using the application was that the response time of the application is a bit high. With the optimization of the algorithms, this problem may be removed in future.

REFLECTIVE DIARY

Aug20 – Aug27:

I have started searching for various topics of the project. Side by side I am also working on a research paper related to Content-Centric Networking. I had started reading papers on CCN in June and by now; I have started writing the paper. As CCN is a new and interesting field, I am thinking of developing an algorithm for data dissemination in CCN and then I will simulate the same algorithm and collect the results of the same for the capstone project.

Aug28 – Sep04:

I covered up the introduction part of the paper and also side by side, I am thinking about the system which I will propose in the paper and for the project. I made a rough design of the proposed system depicting how the system will work and how data will be disseminated in the network from one node to the next. I also searched on the internet for the network simulators which I can use and I studied a little bit about NS2 simulator.

Sep05 – Sep12:

In this week, I discussed about the system which I have proposed with my guide. He liked the way in which I had proposed the data flow in the network but my work was lacking that novelty which would make my work stand out. So now, my guide has told me to study a Deep Learning tutorial and specifically about Convolutional Neural Networks, which is an emerging field of research in various applications. He told me to study it carefully and then think about how I can integrate it with my work.

Sep13 – Sep20:

As I have started studying about the Convolutional neural networks, I have started developing interest in knowing how it works, what its parameters are and how it computes the output. I read multiple articles on the internet and saw many video tutorials to understand its working. Most of the work in this field has been done in image recognition and computer vision. I also read its various applications including object recognition, image classification etc. and I think it would be a good idea to implement it for the capstone project.

Sep21 – Sep28:

I have started working on the Convolutional Neural Network implementation. But on Tuesday, as I was searching the internet for some guidance, I came across the application of CNNs in Natural Language Processing as well. I was intrigued by the way in which people have the capability of adapting the same concept to different scenarios. Image detection had a little scope in the research work that I am doing, but as I saw the application of CNN in NLP, it quickly struck my mind what are the possibilities of doing this work and what advantages this would give to my project work. So now, I have started working on CNN in the field of NLP and the domain which I have chosen specifically is Sentiment Analysis.

Sep29 – Oct06:

In this week, I learnt about the various CNN hyper parameters which affect the performance of CNN model including stride size, filter size, number of filters etc. Also, I read some papers on CNN and regarding the work that has been previously done in the field of sentiment analysis. I read about the traditional techniques which are being used in sentiment and text analysis and what its drawbacks are. I also read about the advantages of CNN over these techniques and how the implementation of CNN would be beneficial to the domain.

Oct07 – Oct14:

In this week, I have sketched a rough algorithm of CNN along with its different parameters. I read multiple papers to decide the values of the parameters which would give optimal results. Along with that, I read different functions that are used in a CNN including ReLU, tanh, sigmoid and softmax function and why these are used. I also read about the different mechanisms by which the error can be reduced like gradient descent and back propagation and also I read that CNN can over fit the data due to its powerful nature and that how dropout mechanism can reduce this possibility.

Oct15 – Oct21:

I started to implement CNN in Python in this week. I have learnt some Python in the previous semester so I decided to code in Python. However, I had been using Python 2.7 is an out dated version, but it didn't give me much problem till date. I have made several programs in it in the

previous semester. But the implementation of CNN requires TensorFlow library which is not available in Python 2.7, which has forced me to upgrade to Python 3. At the same time, I don't want to lose my previous work and make it incompatible and non-workable because there are some changes in the syntax of code in Python 3, which although is minor, but gives errors while running the code. So now, I am looking for alternatives to download and install two versions of Python on my laptop.

Oct22 – Oct29:

In this week, I have done some major work. First I have installed Anaconda which will allow me to run two different versions of Python on my single machine and secondly I versed myself with the new syntax of Python 3 with which I will have to work now. Also, I have started the coding part. I am taking help from a nice tutorial which I found on the internet. In that tutorial, the steps to implement a CNN are explained very precisely and it is helping me a lot.

Oct30 – Nov6:

I have completed the coding part of CNN and now I am proceeding towards the evaluation part. The training part has been completed but it was rather a difficult part. First, I had to search for a relevant data set which I could use in my work. I decided to use a movie review dataset and I got it from rottentomatoes.com. Then when I started training my model on the data set, it gave errors repeatedly telling me that my laptop does not have the required GPU resources to train the model with that dataset. Then I shifted the device from GPU to CPU and reduced the size of the training data set to run it on my existing laptop. I know this will affect the accuracy of the model but I don't have any other option.

Nov7 – Nov14:

In this week, I have finally completed the implementation part with the training, testing and evaluation complete. The training part took a lot of time, it took about 8-10 hours to train the model and I had to do it twice because the first time I forgot to take the screenshot and then I remembered I would need it for the report, so I had to run it once again. Then I carried out the testing and evaluation part and stored the result in a csv file for future use. I have also obtained the accuracy and confusion matrix of the model.

Nov15 – Nov22:

Since most of my work is done now, but I still have almost 3 weeks, I have decided that I will make a web interface for the model. I have trained and tested the data and also obtained some numerical figures as the result but I want some real-time application of the model, so that whoever uses the application is able to predict how accurate my model is. So, I have decided to make a movie reviewing application in which the user will be able to add reviews for a movie and the model will run in the background and will predict the sentiment of the user input.

Nov23 – Nov30:

In this week, I made the user interface part and integrated it with Python. The making of user interface was easy as I have done quite a handful of projects in php by now. But the integration part was a bit difficult. It took me several attempts and numerous hours to find the proper code for integration of python with php. Nonetheless, my work is complete now and now I have to make its documentation.

Dec01 – Dec07:

In this week, I made the report. I started by making all the UML diagrams of the project and then gathered the content that I will write in the report. Report making is a very tiresome job for me. Although the report has been completed by me, but every time I look at it, I find a new mistake and then I have to re-work to correct it. Also, I spent this week making a nice presentation for the project as it has to be shown for the capstone evaluation next week.

Dec08 – Dec14:

The work is almost complete. I just made a video and uploaded it on YouTube as per the requirements of evaluation and also made the poster. I also showed my work to my friend and she reviewed it nicely.