SUGGESTION OF GRADUAL RELAXATION TECHNIQUES USING EMOTION RECOGNITION

A PROJECT REPORT

submitted by

ABHISHEK S (CHN15CS000)
GOKUL V NAMBOOTHIRI (CHN15CS047)
NOBIN JACOB (CHN15CS000)
SEBON GEORGE VARGHESE (CHN15CS000)

to

The APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

In

Computer Science and Engineering

Department of Computer Science and Engineering
College of Engineering, Chengannur, Kerala -689121
May 2019

DECLARATION

We undersigned hereby declare that the project report ”Suggestion of gradual relaxtion techniques using emotion recognition”, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Mrs.C Jyothirmayi Devi, Assistant Professor. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: CHENGANNUR
Date:2019/06/07

ABHISHEK S
GOKUL V NAMBOOTHIRI
NOBIN JACOB
SEBIN GEORGE VARGHESE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING,CHENGANNUR

CERTIFICATE

This is to certify that the report entitled **”Suggestion of gradual relaxtion techniques using emotion recognition”** submitted by **ABHISHEK S** , **GOKUL V NAMBOOTHIRI**, **NOBIN JACOB** , **SEBIN GEORGE VARGHESE** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Department of Computer Science and Engineering, College of Engineering, Chengannur, Kerala -689121 is a bonafide record of the project work carried out by them under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Ms.C Jyothirmayi Devi
Assitant Professor
Project Supervisor

Ms. Manjusha Nair S
Associate Professor
Project Co-ordinator

Dr. Smitha Dharan
Professor and Head

ACKNOWLEDGEMENT

First and foremost we wish to express our wholehearted indebtedness to God Almighty for his gracious constant care and blessings showered over us for the successful completion of this project.

We are deeply indebted to **Dr. Jacob Thomas V**, Principal, College of Engineering, Chengannur and Associate Professor **Dr. Smitha Dharan**, Head of the Department of Computer Science and Engineering, College of Engineering, Chengannur, for providing and availing all the required facilities for undertaking the project in a systematic way.

We would like to express our deep gratitude to our staff-in-charge **Mrs. Manjusha Nair S**, Associate Professor, for the incite and encouragement given by her to improve the project. We are thankful to our guide **Ms.C Jyothirmayi Devi**, Assistant Professor for providing good suggestions and valuable advices to improve the project.

Gratitude is extended to all teaching and non teaching staffs of Department of Computer Science and Engineering, College of Engineering, Chengannur, for the sincere directions imparted and the cooperation in connection with the project.

We are also thankful to our parents for the support given in connection with the project. Gratitude may be extended to all well-wishers and friends who supported us.

ABHISHEK S
GOKUL V NAMBOOTHIRI
NOBIN JACOB
SEBIN GEORGE VARGHESE

ABSTRACT

A Facial expression is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind. Facial expression recognition system is implemented using Convolution Neural Network (CNN). CNN model of the project is based on LeNet Architecture. Kaggle facial expression dataset with seven facial expression labels as happy, sad, surprise, fear, anger, disgust, and neutral is used in this project. The system achieved 56.77 % accuracy and 0.57 precision on testing dataset.

Keywords: Facial Expression,CNN,Cognitive Activity

Contents

List of Figures

ABBREVIATIONS

CNN	Convolution Neural Network
IE	Information Extraction
DL	Deep Learning
ANN	Artificial Neural Networks

CHAPTER 1

INTRODUCTION

1.1 General Introduction

A Facial expression is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind.

Automatic recognition of facial expressions can be an important component of natural humanmachine interfaces; it may also be used in behavioral science and in clinical practice. It have been studied for a long period of time and obtaining the progress recent decades. Though much progress has been made, recognizing facial expression with a high accuracy remains to be difficult due to the complexity and varieties of facial expressions. On a day to day basics humans

commonly recognize emotions by characteristic features, displayed as a part of a facial expression. For instance happiness is undeniably associated with a smile or an upward movement of the corners of the lips. Similarly other emotions are characterized by other deformations typical to a particular expression. Research into automatic recognition of facial expressions addresses the problems surrounding the representation and categorization of static or dynamic characteristics of these deformations of face pigmentation . In machine learning, a convolutional neural

network (CNN, or ConvNet) is a type of feedforward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation. Convolutional networks were inspired by biological processes and are variations of multilayer perceptron designed to use minimal amounts of preprocessing. They have wide applications in image and video recognition, recommender systems and natural language processing. The convolutional neural network is also known as shift invariant or space invariant artificial neural network (SIANN), which is named based on its shared weights architecture and translation invariance characteristics.

1.2 Objective

The main aim of this project is to understand the principles of Facial Expression Recognition by developing a CNN classifier machine along with subsampling and pooling techniques. In this project,facial expression recognition system is implemented using convolution neural network to analyze the human state information to suggest relaxation techniques such as a movie that matches or overcomes his/her current state of mind. Facial images are classified into seven facial expression categories namely Anger, Disgust, Fear, Happy, Sad, Surprise and 'Neutral. Kaggle dataset is used to train and test the classifier. These seven categories of experssions are used to identify which genre of movie is perfect for a human. This system can be used as an added utility/feature for a service robot in domestic applications to enhance autonomous expression recogniton of a human and thus be able to provide relaxation suggestions instantly.

The main focus of this work is implementing a CNN system using supervised methods. We explore various statistical measure for identifying Facial Expressions, study the deep learning/Neural Network approaches for Named Face recogniton.

1.3 Scope

The scope of this system is to tackle with the problems that can arise in day to day life. Some of the scopes are:

1. The system can be used to detect and track a user's state of mind.
2. The system can be used in mini-marts, shopping center to view the feedback of the customers to enhance the business.

3. The system can be installed at busy places like airport, railway station or bus station for detecting human faces and facial expressions of each person. If there are any faces that appeared suspicious like angry or fearful, the system might set an internal alarm.
4. The system can also be used for educational purpose such as one can get feedback on how the student is reacting during the class.
5. This system can be used for lie detection amongst criminal suspects during interrogation
6. This system can help people in emotion related -research to improve the processing of emotion data.

CHAPTER 2

LITERATURE SURVEY

Two differnent approaches are used for face recognition, both of which include two different methodologies. Dividing the face into separate action units or keeping it as a whole for further processing appears to be the first and the primary distinction between the main approaches. In both of these approaches, two different methodologies, namely the 'Geometric based' and the 'Appearance-based' parameterizations, can be used. Making use of the whole frontal face image and processing it in order to end up with the classifications of 6 universal facial expression prototypes: disgust, fear, joy, surprise, sad ness and anger; outlines the first approach. Here, it is assumed that each of the above mentioned emotions have characteristic expressions on face and that's why recognition of them is necessary and sufficient. Instead of using the face images as a whole, dividing them into some sub-sections for further processing forms up the main idea of the second approach for facial expression analysis. As expression is more related with subtle changes of some discrete features such as eyes, eyebrows and lip corners; these fine-grained changes are used for analyzing automated recognition. There are two main methods that are used in both of the above explained approaches. Geometric Based Parameterization is an old way which consists of tracking and processing the motions of some spots on image sequences, firstly presented by Suwa et al to recognize facial expressions [7]. Cohn and Kanade later on tried geometrical modeling and tracking of facial features by claiming that each AU is presented with a specific set of facial muscles [8]. The disadvantages of this method are the contours of these features and components have to be adjusted manually in this frame, the problems of robustness and difficulties come out in cases of pose and illumination changes while the tracking is applied on images, as actions & expressions tend to change both in morphological and in dynamical senses, it becomes hard to estimate general parameters for movement and displacement. Therefore, ending up with robust decisions for facial actions under these varying conditions becomes to be difficult. Rather than tracking spatial points and using positioning and movement parameters that vary within time, color (pixel) information of related regions of face are processed in Appearance Based Parameterizations; in order to obtain the parameters that are going to form the feature vectors. Different features such as Gabor, Haar wavelet coefficients, together with feature extraction and selection methods such as PCA, LDA, and Adaboost are used within this framework. For classification problem, algorithms like Machine learning, Neural Network, Support Vector Machine, Deep learning, Naive Bayes are used. Raghuvanshi A. et al have built a Facial expression recognition system upon recent research to classify images of human faces into discrete emotion categories using convolutional neural networks [9]. Alizadeh, Shima, and Azar Fazel have developed Facial Expression Recognition system using Convolutional Neural Networks based on Torch model.

2.1 Related work(1)

1. Face detection and facial expression recognition system was developed and presented in ieee 2014
2. It used Artificial Neuro -Fuzzy interference system to train and test images captured.
3. Classifier is based on Takagi Sugeno fuzzy system.
4. Even though it played a communicative role in interpersonal relations,it demanded considerably high amount of computational requirement
5. Images used in training and testing were low quality images and cases of poor preprocessing.

2.2 Related work(2)

1. Real-time Convolutional Neural Networks for Emotion recognition in ieee 2018
2. This paper implement general Convolutional network framework for real time applications.
3. Accuracy levels of 66% were reported in the FER-2013 emotiondataset..

4. Use of guided back-propagation focused on dynamics of weights and to evaluate the learned features

CHAPTER 3

METHODOLOGY

This chapter describes the sytematical approach for implementation of an face expression recognition system.Development of an FER system basically has many approaches of which we have implemented a neural network based CNN .
The block diagram of the system is shown in following figures:

3.1 Training phase

During training, the system received a training data comprising grayscale images of faces with their respective expression label and learns a set of weights for the network. The training step took as input an image with a face. Thereafter, an intensity normalization is applied to the image. The normalized images are used to train the Convolutional Network. To ensure that the training performance is not affected by the order of presentation of the examples, validation dataset is used to choose the final best set of weights out of a set of trainings performed with samples presented in different orders. The output of the training step is a set of weights that achieve the best result with the training data. During test, the system received a grayscale image of a face from test dataset, and output the predicted expression by using the final network weights learned during training. Its output is a single number that represents one of the seven basic expressions

3.2 Datasets

The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) and selected data from Extended Cohn-canade dataset are used for training and FER2013 is alone used for testing.

3.2.1 FER2013

It comprises pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. Dataset has training set of 35887 facial images with facial expression labels.. The dataset has class imbalance issue, since some classes have large number of examples while some has few. The dataset is balanced using oversampling, by increasing numbers in minority classes. The balanced dataset contains 40263 images, from which 29263 images are used for training, 6000 images are used for testing, and 5000 images are used for validation.

3.2.2 Extended Cohn-canade AU coded(CK+)

The Cohn-Kanade AU-Coded Facial Expression Database is for research in automatic facial image analysis and synthesis and for perceptual studies. Cohn-Kanade is available in two versions and a third is in preparation. We used some selective data from the dataset which includes both posed and non-posed (spontaneous) expressions and additional types of metadata.In addition validated emotion labels have been added to the metadata. Additionally, CK+ provides

protocols and baseline results for facial feature tracking and action unit and emotion recognition. Tracking results for shape and appearance are via the approach of Matthews & Baker (2004).In the original dsitribution, CK included 486 FACS coded sequences from 97 subjects , but for CK+, there are 593 sequences from 123 subjects.

3.3 Architecture of CNN

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. CNN is designed with some modification on LeNet Architecture [10]. It has 6 layers without considering input and output. The architecture of the Convolution Neural Network used in the project is shown in the following figure: The structural and functional behaviours of each layer is described as follows.

3.3.1 Input Layer

There are various models that are used for the developing statistical NER. These models have their own mathematical approaches and techniques for training the corpus, determining the probabilistic values and have their own methodologies of working to get the desired result.

3.3.2 Convolution and Pooling Layers

A conditional random field (CRF) is a type of discriminative undirected probabilistic model. It is most often used for

3.3.3 Fully Connected Layer

3.3.4 Output Layer

3.4 Movie Suggestion phase

The CRF model can be extended into higher order by increasing the number of the sequence of label. In this case the labels are made dependent on fixed numbers of variables. If the number of variable becomes very large then training and inference becomes complex and hence alternative training and inference methods are applied. The semi-Markov conditional random field (semi-CRF), models variable-length segmentations of a label sequence. This provides much of the power of higherorder CRFs to model long-range dependencies of the sequence at a reasonable computational cost. Conditional random field were introduced by Lafferty et al.as a statistical modeling tool for pattern recognition and machine learning using structured prediction. McCallum and Li proposed a feature induction method for CRF in NE. Let $\langle o = o_1, o_2, ., ., ., o_T \rangle$ be some observed input data sequence, such as a sequence of words in a text inside the document. Let S be a set of FSM states,each of which is associated with a label. Let $\langle s = s_1; s_2; .; .; .; s_T \rangle$ be some sequence of states. By the Hammersley Clifford theorem, CRFs define the conditional probability of a state sequence given an input sequence to be

$$P(s|o) = \frac{1}{Z} exp \left(\sum_{t=1}^{T} \lambda_k f_k(s_{t-1}, s_t, o, t) \right)$$

where Z is the normalization factor obtained by marginalizing over all state sequences, $f_k(s_{t-1}, s_t, o, t)$ is an arbitrary feature function and λ_k is the learned weight for each feature function. By using dynamic programming, state transition between two CRF states can be efficiently calculated. The modified forward values, $\alpha_t(s_t)$, to be the "unnormalized probability" of arriving state s_t given the observations, $\langle o = o_1, o_2, ., ., ., o_T \rangle$. $\alpha_0(s)$ is set to probability of starting in each state s, and recursively calculated as :

$$\alpha_{t+1}(s) = \sum_{s'} \alpha_t(s') exp \left(\sum_{t=1}^{T} \lambda_k f_k(s', s, o, t) \right)$$

The backward procedure and Baum-Welch have been similarly modified. Z_0 is given by $\sum_s \alpha_T(s)$

3.4.0.1 SVM Based Models

Support Vector Machine was first introduced by Cortes and Vapnik (1995) based on the idea of learning a linear hyperplane that separate the positive examples from negative example by large margin. Large margin suggests that the distance between the hyperplane and the point from either instances is maximum.The points closest to hyperplane on either side are known as support vectors.

The linear classifier is based on two parameters, a weight vector W perpendicular to the hyperplane that separates the instances and a bias b which determines the offset of the hyperplane from the origin. A sample x is classified as positive instance if $f(x) = wx + b > 0$ and negative otherwise.If the data points are not linearly separable, then a slack is used to accept some error in classification. This prevents the classifier to overfit the data. When there are more than two classes, a group of classifiers are used to classify the instance.The below figure shows the geometric interpretation.

Hyperplanes and Support Vectors : Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. Deleting the support vectors will change the position of the hyperplane. These are the points that help to build the SVM.

Cost Function and Gradient Updates : The SVM algorithm tries to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.If the loss function is convex, then a locally optimal point is globally optimal (provided the optimization is over a convex set, which it

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not,

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{otherwise} \end{cases}$$

Hinge loss function

we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss.Take partial derivates with respect to the weigts to find the gradients.Using the gradients weights are updated.

$$\frac{\partial(1 - y_i \langle x_i, w \rangle)}{\partial w_k} = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{otherwise} \end{cases}$$

Gradients

When there is no misclassification, i.e when the model correctly predicts the class of the data point, update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient update- No misclassification

When there is a misclassification, i.e when the model make a mistake on the prediction of the class of the data point, then include the loss along with the regularization parameter to perform gradient update.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Gradient update- No misclassification

3.4.0.2 Multinomial Naive Bayes Models

In machine learning, naive Bayes classifiers are a family of simple ”probabilistic classifiers” based on applying Bayes’ theorem with strong (naive) independence assumptions between the features.Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

The Naive Bayes classification algorithm is based off of Bayes' Theorem. Bayes' Theorem is a beautiful yet simple theorem developed primitively by English statistician Thomas Bayes in the early 1700s. At its most basic level, Bayes' Theorem uses conditional probabilities to "predict" the outcome of later probabilities. The Bayesian concept of a priori probability is important: it is used as a ground truth from which we can use to obtain the outcome of other probabilities.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Bayes Theorem

$P(c)$ indicates the a priori probability of the given class, while $P(x)$ represents the probability of a given predictor.

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.It estimates the conditional probability of a particular word given a class as the relative frequency of term t in documents belonging to class(c). The variation takes into account the number of occurrences of term t in training documents from class (c),including multiple occurrences.

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \ldots, p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document . The likelihood of observing a histogram x is given by

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i!)}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

In contrast to the other event models, the multinomial model captures word frequency information in documents.In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary. The model assumes that the lengths of documents are independent of class. It also make a similar naive Bayes assumption: that the probability of each word event in a document is independent of the word's context and position in the document. Thus, each document is drawn from a multinomial distribution of words with as many independent trials as the length of document . This yields the familiar "bag of words" representation for documents.

3.4.1 Feature space for NERC

Features are descriptors or characteristic attributes of words designed for algorithmic consumption. An example of a feature is a Boolean variable with the value true if a word is capitalized and false otherwise. Feature vector representation is an abstraction over text where typically each word is represented by one or many Boolean, numeric and nominal values. Usually, the NERC problem is resolved by applying a rule system over the features. For instance, a system might have two rules, a recognition rule: "capitalized words are candidate entities" and a classification rule: "the type of candidate entities of length greater than 3 words is organization". These rules work well for the exemplar sentence above. However, real systems tend to be much more complex and their rules are often created by automatic learning techniques. In this scenario, the sentence "The president of Apple eats an apple.", excluding the punctuation, would be represented by the following feature vectors: $< true, 3, \backslash the" >, < false, 9, \backslash president" >, < false, 2, \backslash of" >, < true, 5, \backslash apple" >$ $, < false, 4, \backslash eats" >, < false, 2, \backslash an" >, < false, 5, \backslash apple" >$

Ihe features most often used for the recognition and classification of named entities are organized them along three different axes:

1. Word-level features

2. List lookup features

3. Document and corpus features

3.4.1.1 Word-level features

Word-level features are related to the character makeup of words. They specifically describe word case, punctuation, numerical value and special characters.

3.4.1.2 List lookup features

Lists are the privileged features in NERC. The terms “gazetteer”, “lexicon” and “dictionary” are often used interchangeably with the term “list”. List inclusion is a way to express the relation “is a” (e.g., Paris is a city). It may appear obvious that if a word (Paris) is an element of a list of cities, then the probability of this word to be city, in a given text, is high. However, because of word polysemy, the probability is almost never 1 (e.g., the probability of “Fast” to represent a company is low because of the common adjective “fast” that is much more frequent).

Common nouns listed in a dictionary are useful, for instance, in the disambiguation of capitalized words in ambiguous positions.Most approaches implicitly require candidate words to exactly match at least one element of a pre-existing list. However, we may want to allow some flexibility in the match conditions. At least three alternate lookup strategies are used in the NERC field.First, words can be stemmed (stripping off both inflectional and derivational suffixes) or lemmatized (normalizing for inflections only) before they are matched

3.4.1.3 Document and corpus features

Document features are defined over both document content and document structure.Large collections of documents (corpora) are also excellent sources of features.

3.5 Deep Learning Approach

Deep learning is set of machine learning algorithms that attempt to learn layered model of inputs, commonly know as neural nets. Each layer tries to learn a concept from previous input layer. With each subsequent layer deep learning algorithm attempts to learn multiple levels of concept of increasing complexity/abstraction. Most of the current machine learning algorithm works well because of humandesigned representations and features. Algorithm then just becomes a optimization problem to adjustweights to best make a final prediction. Deep learning is about representation learning with good features automatically.

3.5.1 Neural Network

Most successful deep learning methods involve neural networks. Deep learning is just a fancy name given to deep neural networks. Neural networks are inspired by central nervous system of animals. In neural network, primary computation node is know as neuron. Neurons are connected to one another using synapses. The strength of connection defines the importance of an input from the receiving neuron.Neural connection adjust the connection strength to learn new pattern in input.

Artificial neural networks are mathematical modelling of biological neural network. In artificial neural network, each neuron is a perceptron with weighted inputs. Weights are analogous to connection strength in biological neural networks. For computational eficiency, layered architecture is used for connections.There are connections only among consecutive layers.With each layer, network tries to learn input pattern that help to correctly classify new examples.

Primary computation node in neural network is a perceptron. Perceptron does a supervised classification of an input into one of the several possible non-binary outputs. It is a linear classi

er that makes prediction using weighted sum of input feature vector. Mathematically, perceptron can be written as

$$\begin{aligned} y &= f(z) \\ \text{where } z &= \sum w_i * x_i = W^T X \end{aligned} \tag{3.1}$$

If f is unity function then a perceptron is equivalent to a linear regression model. If f is sigmoid function then the preceptron acts as logistic regression.

3.5.2 Backpropogation algorithm

Backpropogation algorithm, proposed by Rumelhart, is one of the most successful training algorithm to train a multilayer feed forward network. A backpropogation algorithm learns by example.Algorithm takes examples as input and it changes the weights in the network links. When the network is fully trained, it will give the required output for a particular input.Basic idea of backpropogation algorithm is to minimize the error with respect to input by propogating error adjustments on the weight of the network. Backpropogation algorithm uses gradient decent method that calculates the squared error function with respect to the weights of the network. The squared error function is

$$E = \frac{1}{2} \sum_{n \epsilon training} (t^n - y^n)^2 \tag{3.2}$$

where, t = target output

y = actual output of output node

Differentiating the above error function gives

$$\frac{\partial E}{\partial W_i} = \frac{dE}{dy^n} \frac{dy^n}{dz^n} \frac{\partial z^n}{\partial W_i} \tag{3.3}$$

Major problems with backpropogation algorithm can be summarized as

1. As we backpropogate deep into the network, gradient progressively gets diminished after each layer. Below certain depth of output neuron, correction signal is very minimal.
2. Since the weights are initialized randomly, backpropogation is susceptible to getting stuck local minima.
3. In usual setting, we can only use labeled data. It defeats the motivation for neural network as brain can learn from unlabeled data.

3.5.3 Distributed Representation

In cognitive science, central problem is to understand how agents represent information that enables them to behave in sophisticated ways. One big contention is whether the representation is localized or distributed. Contention remains whether knowledge is stored in specific, discrete region of brain or entire cortex. But with advent of connectionist models in mathematics, distributed representation has found great attention.Major benefit of using distributed representation is sharing of features to represent instance a knowledge. In most basic sense, a distributed representation is one that is spread out over a set of features for representation as opposed to localized approach where each feature is independent of each other.

3.5.3.1 Distributed representation for words

A word representation is a mathematical object associated the each word, often a vector. Each dimension of the vector represents a feature and might even have a mathematical interpretation. Value of each dimension represents the amount of activity for that particular feature.

In machine learning, one of the most obvious model of representing a word is one-hot vector representation. In this representation only one of the computing element is active for each entity element. For example, if the size of vocabulary is $|V|$ then word w can be represented as vector of size $|V|$ in which the index of word w is only active and rest are set to zero.

$$Home : [0, 0, 0, 0, ..., 1, ..., 0, 0]$$

$$House : [0, 0, 0, 0, .., 1, .., 0, 0, 0, 0]$$

This representation is known as local representation. It is easy to understand and implement on hardware. But this representation has many flaws of itself. As in example shown above, if we want the correlation between Home and House, the representation fails to show any correlation between the terms.

Distributed representation would represent these words in some lower dimensional dense vector of real values with each dimension representing a latent feature for word model. Distributed representation could be like :

$$Home : [0.112, 0.432, ..., 0.341]$$

$$House : [0.109, 0.459, ..., 0.303]$$

Distributed representation helps to solve the problem of sparsity. For words that are rare in the labeled training corpus, parameters estimated through one-hot representation will be poor. More over, the model cannot handle the word that do not appear in the corpus. Distributed representation are trained using large unlabeled corpus using an unsupervised algorithm. Hope is that the distributed representation would capture semantic and syntactic properties of word and would have a similar representation for syntactically and semantically related words.

3.5.4 The Window Approach

All the NLP tasks can be seen as tasks assigning labels to words. The traditional NLP approach is: extract from the sentence a rich set of hand-designed features which are then fed to a standard classification algorithm.The choice of features is a completely empirical process, mainly based first on linguistic intuition, and then trial and error, and the feature selection is task dependent, implying additional research for each new NLP task.

Instead,use a radically different approach: as input we will try to pre-process our features as little as possible and then use a multilayer neural network (NN) architecture, trained in an end-to-end fashion. The architecture takes the input sentence and learns several layers of feature extraction that process the inputs. The features computed by the deep layers of the network are automatically trained by backpropagation to be relevant to the task.Such general multilayer architecture suitable for all our NLP tasks, which is generalizable to other NLP tasks as well.

The first layer extracts features for each word. The second layer extracts features from a window of words or from the whole sentence,treating it as a sequence with local and global structure (i.e., it is not treated like a bag of words).The following layers are standard NN layers.

3.5.4.1 Transforming Words into Feature Vectors

One of the key points of this architecture is its ability to perform well with the use of raw words. The ability for this method to learn good word representations is thus crucial to our approach. For efficiency, words are fed to our architecture as indices taken from a finite dictionary. Obviously, a simple index does not carry much useful information about the word. However,the first layer of our network maps each of these word indices into a feature vector, by a lookup table operation. Given a task of interest, a relevant representation of each word is then given by the corresponding lookup table feature vector, which is trained by backpropagation, starting from a random initialization.Such an architecture allow to take advantage of better trained word representations, by simply initializing the word lookup table with these representations.

3.5.4.2 Window concept

A window approach assumes the tag of a word depends mainly on its neighboring words. Given a word to tag, we consider a fixed size k_{sz} (a hyper-parameter) window of words around this word. Each word in the window is first passed through the lookup table layer producing a matrix of word features of fixed size $d_{wrd} * k_{sz}$. This matrix can be viewed as a $d_{wrd}k_{sz}$-dimensional vector by concatenating each column vector, which can be fed to further neural network layers.

HardTanh Layer: Several linear layers are often stacked, interleaved with a non-linearity function, to extract highly non-linear features. If no non-linearity is introduced, our network would be a simple linear model. We chose a "hard" version of the hyperbolic tangent as non-linearity. It has the advantage of being slightly cheaper to compute (compared to the exact hyperbolic tangent), while leaving the generalization performance unchanged. The corresponding layer l applies a HardTanh over its input vector.

Scoring: Finally, the output size of the last layer L of our network is equal to the number of possible tags for the task of interest. Each output can be then interpreted as a score of the corresponding tag

3.5.4.3 Training

All our neural networks are trained by maximizing a likelihood over the training data, using stochastic gradient ascent. If we denote θ to be all the trainable parameters of the network, which are trained using a training set *T* we want to maximize the following log-likelihood with respect to θ.Let x corresponds to a training word window and its associated features, and y represents the corresponding tag. The probability $p(y|x, \theta)$ is computed from the outputs of the neural network.

In this approach, each word in a sentence is considered independently. Given an input example x, the network with parameters θ outputs a score $[f_\theta(x)]_i$, for the i^{th} tag with respect to the task ofinterest.This score can be interpreted as a conditional tag probability $p(i|x, \theta)$ by applying a softmax operation over all the tags.While this training criterion, often referred as cross-entropy is widely used for classification problems, it might not be ideal in our case, where there is often a correlation between the tag of a word in a sentence and its neighboring tags.

CHAPTER 4

SYSTEM ANALYSIS AND SPECIFICATION

4.1 Required Packages

4.1.1 NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

NLTK includes more than 50 corpora and lexical sources such as the Penn Treebank Corpus, Open Multilingual Wordnet, Problem Report Corpus, and Lin's Dependency Thesaurus.

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, opensource, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

4.1.2 Spacy

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python.spaCy is designed specifically for production use and helps you build applications that process and “understand” large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning.

While some of spaCy’s features work independently, others require statistical models to be loaded, which enable spaCy to predict linguistic annotations – for example, whether a word is a verb or a noun. spaCy currently offers statistical models for a variety of languages, which can be installed as individual Python modules. Models can differ in size, speed, memory usage, accuracy and the data they include. The model you choose always depends on your use case and the texts you’re working with. For a general-purpose use case, the small, default models are always a good start.

spaCy provides a variety of linguistic annotations to give you insights into a text’s grammatical structure. This includes the word types, like the parts of speech, and how the words are related to each other. For example, if you’re analyzing text, it makes a huge difference whether a noun is the subject of a sentence, or the object – or whether “google” is used as a verb, or refers to the website or company in a specific context

4.1.3 Scikit-learn

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Scikit-learn is used to build models. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g. NumPy, Pandas etc.)

To implement Scikit learn, we first need to import the above packages. If you are not familiar with these libraries, you can have a look at my previous blogs on Numpy and Matplotlib. You can download these two packages using the command line or if you are using PyCharm, you can directly install it by going to your setting in the same way you do it for other packages.

Next, in a similar manner, you have to import Sklearn. Scikit learn is built upon the SciPy (Scientific Python) that must be installed before you can use Scikit-learn. You can refer to this website to download the same. Also, install Scipy and wheel package if it's not present, you can type in the below command:

```
pip install scipy
```

Scikit learn comes with sample datasets, such as iris and digits. You can import the datasets and play around with them. After that, you have to import SVM which stands for Support Vector Machine. SVM is a form of machine learning which is used to analyze data.

4.1.4 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric, also variously called Numerical Python extensions or NumPy. Hugunin, a graduate student at Massachusetts Institute of Technology (MIT), joined the Corporation for National Research Initiatives (CNRI) to work on JPython in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer. Other early contributors include David Ascher, Konrad Hinsen and Travis Oliphant.

A new package called Numarray was written as a more flexible replacement for Numeric. Like Numeric, it is now deprecated. Numarray had faster operations for large arrays, but was slower than Numeric on small ones,so for a time both packages were used for different use cases. There was a desire to get Numeric into the Python standard library, but Guido van Rossum decided that the code was not maintainable in its state then.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

4.1.5 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term ”panel data”, an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling if not THE most used one. Pandas is an open source, free to use (under a BSD license) and it was originally written by Wes McKinney (here's a link to his GitHub page). What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

In order to "get" Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like Numpy) and has optional dependencies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the Anaconda distribution , "a cross platform distribution for data analysis and scientific computing." There you can download the Windows, OS X and Linux versions. If you want to install in a different way, these are the full installation instructions.

In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it's there for you to work with. In order to import Pandas all you have to do is run the following code:

$import\,pandas\,as\,pd\,import\,numpy\,as\,np$

There are different commands to each of these options, but when you open a file, they would look like this:

$pd.read_filetype()$

There are different file types Pandas can work with, so you would replace "filetype" with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all you them, you would have to read the documentation (for example, the documentation for $pd.read_csv()$ would contain all the arguments you can pass in this Pandas command).

4.1.6 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

The micro framework Flask is based on the Pocoo projects Werkzeug and Jinja2.

Werkzeug is a utility library for the Python programming language, in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom software framework on top of it and supports Python 2.6, 2.7 and 3.3.

Jinja, also by Ronacher, is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it provides that templates are evaluated in a sandbox.

Python 2.6 or higher is usually required for installation of Flask. Although Flask and its dependencies work well with Python 3 (Python 3.3 onwards), many Flask extensions do not support it properly. Hence, it is recommended that Flask should be installed on Python 2.7.

virtualenv is a virtual Python environment builder. It helps a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries.

The following command installs virtualenv

$pip\, install\, virtualenv$

This command needs administrator privileges. Add sudo before pip on Linux/Mac OS. If you are on Windows, log in as Administrator. On Ubuntu virtualenv may be installed using its package manager.

$sudo\, apt - get\, install\, virtualenv$

Once installed, new virtual environment is created in a folder.

$mkdir\, newproj\, cd\, newproj\, virtualenv\, venv$

To activate corresponding environment, on Linux/OS , use the following:

$venv/bin/activate$

We are now ready to install Flask in this environment.

$pip\, install\, Flask$

The above command can be run directly,without virtual environment for system-wide.

CHAPTER 5

RESULT

5.1 User Interface

CHAPTER 6

FUTURE WORK

Named Entity Recognition is an important method in order to extract relevant information. For domain specific entity, we have to spend lots of time on labeling so that we can recognize those entity. For general entity such as name, location and organization, we can use pre-trained library like Stanford NER, spaCy.

Currently, the named entity task is changing from tagging only proper names to tagging a broader range of words and expressions that are of interest to people with particular information needs. Here, our approach supports both domain specific and general entity recognition. This approach for NERC improves the accuracy and efficiency of the NER classification. Developing an advanced relation extraction system along with this NERC powered by deep learning will in turn leads to the development of an efficient Information Extraction (IE) system. Such an IE system will be very useful in the coming data age when enormous size of data are being created and need to be processed.

This system will be also useful in the field of big data, data analytics. They may be also useful in automated text summarization. Even, our approach can be rectified and improved for more efficiency and accuracy. The current accuracy rate of 90-94% may be improved to 95-98% in near future.

NER, though considered to be a basic NLP function, is challenged by various complexities that are inherent in any natural language. Few of the challenges are described below:

1. Ambiguity and Abbreviations -One of the major challenges in identifying named entities is language. Recognizing words which can have multiple meanings or words that can be a part of different sentences. Another major challenge is classifying similar words from texts. Multiple words or sentences can be written in different forms .Words can be abbreviated for ease of writing and understanding .Same words can be written in long

forms. Words which will sometimes require some label for identification is another major challenge.

2. Spelling Variations-The vowels (a, e, i, o, u) in English language plays a very important role. Words which do not make a major difference in phonetics but make a major difference in the way of writing and its spelling.
3. Foreign Words-Words which are not used very frequently these days, or words that are not heard by a lot of people, is another major challenge in this field. Words like person names, location names etc.

All these challenges need to be fixed in future for successful implementation of an efficient IE system with NER and RE.

CHAPTER 7

CONCLUSION

The Named Entity Recognition field has been thriving for more than fifteen years. It aims at extracting and classifying mentions of rigid designators, from text, such as proper names, biological species, and temporal expressions. Many new researches are going on in this field. An overview of the two basic approaches to develop a NER has been looked on. The statistical approach includes details of some of the existing models which are used. Also the various learning methodologies to train the system for this approach are also discussed. Finally the new approach, using deep learning has also been briefly stated.

Handcrafted systems provide good performance at a relatively high system engineering cost. When supervised learning is used, a prerequisite is the availability of a large collection of annotated data. Such collections are available from the evaluation forums but remain rather rare and limited in domain and language coverage. Recent studies in the field have explored semi-supervised and unsupervised learning techniques that promise fast deployment for many entity types without the prerequisite of an annotated corpus.

Named Entity Recognition will have a vital impact on our society as it will enable us extract more and more information from a piece of text by identifying and classifying the named entities which may be known or unknown to us. It is indeed the basis of a major advance in biology and genetics, enabling researchers to search the abundant literature for interactions between named genes and cells.

REFERENCES

[1] **David Nadeau, Satoshi Sekine**, *"A survey of named entity recognition and classification"*, NRCC, New York University, 2007

[2] **Kashif Riaz**,*"Rule-based Named Entity Recognition in Urdu"*,Proceedings of the 2010 Named Entities Workshop, ACL 2010, July 2010.

[3] **Christopher D. ManningHinrich Schütze**,*"Foundations of Statistical Natural Language Processing"*,MIT, May, 1999

[4] **Chris Callision-Burch**, "Learning useful representations in a deep network with a local denoising criterion", *Journal of machine learning research* , 11 (2010) 3371–3408.

[5] **S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio**, "Explicit invariance during feature extraction", *Proceedings of the 28th International Conference on International Conference on Machine Learning, Omnipress,* pp. 833–840.

[6] **M. Y. Azar, K. Sirts, L. Hamey, D. M. Aliod**, " Query-based single document summarization using an ensemble noisy auto-encoder", *in: Proceedings of the Australasian Language Technology Association Work-shop* 2015, pp. 2–10

[7] **G. Bebis, M. Georgiopoulos**, " Feed-forward neural networks", *IEEE Potentials* 13 (1994) 27–31.

[8] **M. T. Nayeem, et al.**, "Methods of sentence extraction, abstraction and ordering for automatic text summarization", Ph.D. thesis, Lethbridge,Alta.: Universtiy of Lethbridge, Department of Mathematics and Computer Science, 2017.

[9] **S. Hochreiter**, "The vanishing gradient problem during learning recurrent neural nets and problem solutions", *International Journal of Uncertainty,Fuzziness and Knowledge-Based Systems* 6 (1998) 107–116

[10] **R. Pascanu, T. Mikolov, Y. Bengio**, "Understanding the exploding gra-dient problem", CoRR, abs/1211.5063 (2012).