NLP Tools for Code-Mixed texts

by

Bapiraju Vamsi Tadikonda

A dissertation submitted in partial satisfaction of the

requirements of

Undergraduate Thesis (BITS F421T)

in

Computer Science

in the

Department of Computer Science & Information Systems

of

Birla Institute of Technology and Science Pilani, Pilani

Committee in charge:

Dr. Yashvardhan Sharma, Chair

May 2017

The dissertation of Bapiraju Vamsi Tadikonda, titled NLP Tools for Code-Mixed texts, is approved:

Chair _____ Date _____

_____ Date _____

Birla Institute of Technology and Science, Pilani

NLP Tools for Code-Mixed texts

Bapiraju Vamsi Tadikonda

Abstract

NLP Tools for Code-Mixed texts

With a wide range of Indian dialects present in India, multilingualism has influenced online networking content. While examining online networking content, code mixing is one of the significant issue confronted by researchers in current situation, as users tend to blend their provincial dialects to convey their views. As a result, analysis of social media texts is hindered.

Part of Speech Tagging is one the most important phase of code mixed analysis as it provides the linguistic structure of the sentence as well as provides an insight on the form of code mixed sentence formed. Named Entity Recognition is also an important tool used for further text analytics. The following tools were developed using Deep Learning, Machine Learning and Neural Network approaches.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my sincere gratitude to Dr.Yashvardhan Sharma, Assistant Professor, CSIS Department for giving me this wonderful opportunity to work on some cutting edge research problems as part of my undergraduate thesis. His constant motivation, enthusiasm and immense knowledge in the subject of Natural Language Processing made sure that he always gave the best advice on how to proceed. I could not have imagined having a better advisor for my thesis.

I want to thank the Dean, Academic Resource and Counselling Division and the Head of the Department, Computer Science and Information Systems Department for giving me the opportunity to pursue a thesis. I would also like to thank Ms. Rupal Bhargava, research scholar in the Web Intelligence & Social Computing Laboratory, BITS Pilani for making me understand the background of the work as well as for constantly checking in on my progress. I would also like to thank my fellow lab mates Shubham, Gargi, Daivik and Deep for helping me in times of need. I would like to thank all my sister who has helped me during the process of annotating the dataset.

Finally, I would like to thank my parents, my grandparents and friends for constantly motivating me and also being there for me during adverse conditions.

Chapter 1

Introduction

The introduction of World Wide Web, has brought out a significant change in the field of computer science directly or indirectly. From offline products to cloud based products, the internet has affected the design and development of applications. The internet has been of use in many areas such as source of information, entertainment and communication. In the area of entertainment, there has been the introduction of audio files being used online and in the present days, there has been online platforms developed for high quality videos of movies, TV series etc. The internet is still the best source of information available. Most of the famous books, novels can be found on the internet. With the establishment of search engines and online encyclopedias finding information on the internet is always a few clicks away. The first and foremost reason why the internet was created is to find a medium of communication. The communication among people has increased a lot with the introduction of internet. Initially emails replaced the traditional mailing system. Later on internet chats, voice and video calls have brought out immense changes in the communication systems. One of such by products of internet communication is the introduction of social media.

The growth of social media is majorly due the trait of human beings to know about others and follow social practices followed by a lot of people. While Social media platforms like Facebook, Twitter and Snapchat capitalize on this very idea it has a lot caused a lot of change in the mindset of people. For example, previously people use to post about major events like engagements, graduation, birth etc on Social media. But now a days people just post their everyday activities on social media. Now with introduction to platforms like twitter where the number of characters is limited to 140, the way people communicate on all platforms has changed a bit. There has been an increase in the number of emoticons and short forms used. In other words, people are resorting to informal means of writing sentences. When language used becomes informal, the linguistic tools present become less reliable as there is a change in the grammar rules. For example, **Ritter etal** has designed a Parts Of Speech tagger for twitter sentences as the traditional POS taggers fail to perform up to the mark.

1.1 Code Mixing and Code Switching

The informal sentences present in social have certain characteristic differences from region to region. In countries having English as their major language, an informal sentence generally consists of shortcuts for specific words, acronyms, emoticons and hashtags. In other countries where English is not the major language, there exists another problem along with the usage of the above tokens. It is the usage of words from their native languages into a sentence along with English words. This tendency of using native language words in sentences along with English is called code-mixing. Code mixing generally happens when the writer is more comfortable explaining certain phrases in a sentence in his native language. For example, consider the sentence *You have that DJ wala look.* In this sentence, the word *wala* is of Hindi language while the other words are in English.

Multilingual Code mixing can also be present where a sentence may have more than two languages present. For example, *You have that DJ wala look kani, konchem hairstyle change chesi unte thoda acha hoga.* In this sentence there are three languages mixed up namely English, Telugu and Hindi. To deal with a bilingual code mixed sentence, one needs to consider the grammar patterns for both the languages along with the hashtags, emoticons and acronyms. With multilingual code-mixed sentences one needs to deal with the patterns observed in all the languages along with the other.

1.2 Motivation

Analyzing social media texts made by a users can state many issues such as the state of mind of the person, the opinion of the person on a certain issue or event or product, etc. For example, let us consider a hypothetical ad campaign for a famous brand on social media platforms. For the company to analyze the response about the campaign, it needs tools which can analyze the text written on the social media platform. In the current scenario, there are tools for basic English and other monolingual texts. There are tools for handling monolingual informal sentences for some of the largely spoken languages. But unfortunately, there aren't a lot of tools present for handling multilingual sentences as well as code mixed sentences.

1.3 Objective

In my thesis I would like to concentrate on exploring and developing various methods for Named Entity Recognition (NER) and Parts of Speech (POS) Tagging for code-mixed Indian languages. The systems developed would be mostly on Hindi-English, Telugu-English and Bengali-English texts but the approach can be extended to other code- mixed texts as well.

Chapter 2

Related Work

2.1 Code Mixing

The concept of code mixing has been researched a lot in a linguistic point of view by (Hoffman 1991, Muysken, 2001; Poplack, 2004; Senaratne,2009; Boztepe, 2005). Hoffman classified code mixed sentences based on the location and scope of the word in the mixing. There were three types of code mixed sentences majorly. The first one was Intra sentential mixing where, the mixing occurs within a phrase or a sentence. The second one is lexical intra mixing of sentences where, the code mixing happens within the boundary of a word. The third and final type was involving in the change of a pronunciation.

After Hoffman, the other significant classification of code mixed sentences was done by Muysken. Muysken's book on bilingual speech has covered various scenarios where code mixing tends to occur and also deals with various forms of code mixing and code switching. He classified code mixing into three kinds Insertion of word phrase, Alternation and Congruents lexicalization.

All these efforts have strengthened the understanding of code mixing and code switching but, there weren't many tools available present to deal with code mixed sentences. One of the major reasons could be due to the unavailability of natural data collection but, with the introduction of text communication via social media platforms, it has become quite easy to collect sentences and scripts required.

The initial attempt to design systems identifying code switching was done by (Solorio and Liu 2008), They developed the system for Spanish and English corpus. They also developed a Parts of Speech tagger for the same dataset using an heuristic function over two traditional Parts of Speech tagger. Similar efforts were made by (Hughes et al.,2006; Baldwin and Lui, 2010; Bergsma et al.,2012) to identify word level language identification in code mixed sentences. All of these systems were not related scripts prevalent in the indian subcontinent.

(Barman, Amitava das etal) have developed a system for identifying language of words in the texts containing English, Bengali and Hindi words.

In order to broaden the tools for code mixing and code switching, there have been several conferences held internationally. In 2014, Conference on Empirical Methods in Natural Language Processing (EMNLP) has conducted the first workshop on code switching. They have organized a shared task for language identification of code switched tweets. The language of tweets were Spanish-English (11,400 tweets), Nepali-English (9993 tweets), Arabic (5,838 tweets) and Mandarin-English (1,000 tweets). The shared task had good spanish, nepali and mandarin tools developed. The arabic tools developed performed poorly comparatively.

Following that, The Eleventh International Conference on Natural Language Processing (ICON 2014) also had their main theme on developing tools and studying linguistics for indian social media texts.There were systems developed for Parts of Speech tagging , Named Entity recognition for indian tweets.

Similar conferences were conducted in 2015 and 2016 as well. In FIRE 2015, there were shared tasks organised to develop Named Entity Recognizers (ESM-IL) for pure indian languages. In FIRE 2016, there was a shared task on Mixed Script Question classification (MSIR) and a named entity recognition task for code mixed languages (CMEE-IL).

2.2 Parts of Speech Recognition

In the past decade, there has been a lot of progress in the field of Natural Language Processing. At present there are Parts of speech Taggers for English with accuracy of 97.64% by Choi (2016) [1] using Dynamic Feature Induction and tested on wall street journal dataset. There are less accurate systems present in indic languages. This can be attributed to the fact that there are very less datasets available publicly for indic languages. According to Antony et al (2011) " Earlier work in POS tagging for Indian languages was mainly based on rule based approaches." [2] In hindi Pushpak bhattacharya et.al [3] designed a POS tagger based on exhaustive morphological analysis and high-coverage lexicon followed by a decision tree based learning algorithm (CN2). This system has achieved an average accuracy of 93.45% . In Bengali, Ekbal et.al [4] has applied a Applied a voted approach on three classifiers SVM, CRF and Me to gain a Tag precision of 92.35%. Rama Sree et.al (2007) [5] have designed a rule based POS Tagger for Telugu corpus and have claimed to have achieved an accuracy of 98.016%. There are POS taggers for tamil,marathi, punjabi, malayalam also developed.

For developing any further linguistic tools for code mixing, Parts of speech is considered the most important as all the patterns and forms of code mixed sentences can be identified with the help of a parts of speech tagger. In fact the initial attempts to POS tag code mixed scripts was done by Solorio and Liu (2008) [6]. They have used different machine learning techniques on a Spanish-English corpus and found out that SVM performs the best. With

SVM they have achieved an accuracy of 93.48%. For indic Languages, Vyas et.al (2014)[7] have developed a POS Tagger for Hindi-English corpus based on combining the output of language-specific POS taggers.

2.3 Named Entity Recognition

Information Extraction and Natural Language Processing is a field that is widely researched upon from time to time in English Language as well as other native Indian languages. Particularly, Named Entity recognition has had significant research done so far in English and multilingual corpuses. Initial attempts to identify Named Entities in multilingual corpuses was done in shared tasks of CoNLL 2002-03. The best systems of that conference were based on machine learning techniques like Hidden Markov models, Maximum Entropy models etc. The CoNLL 2002 [16] was based on Spanish and Dutch language data sets while CoNLL 2003 [18] was based on English and German data sets. In 2013, [14] have designed a system called POLYGLOT for massive multilingual NLP applications. This system can handle 40 major languages including some Indian languages like Bengali, Hindi, Marathi, Punjabi, Tamil and Telugu for NER classification. The NER system was developed based on a Massive Wikipedia Database and applied semi-supervised approach for classification.

In the context of Indian languages, a hybrid maximum entropy model for Hindi and Bengali languages was designed by [15]. Authors used gazetteer lists and language specific rules in the system. For Tamil language a NER system based on Conditional random fields (CRF) was designed by [8]. When it comes to NER systems for code-mixed texts in Indian Languages, initial Attempts were made by ESM-IL subtask of FIRE 2015 [12]. The task consists of identifying Named Entities in Hindi and English social media text. A Conditional Random Field baseline system was built and other systems were compared with respective to that system. It was observed by them that most of the systems designed had similar precision but most of them have improved the recall making them better systems. [10] proposed a system by creating a set of features and training them using Conditional Random Fields(CRF) identifying Named Entities. A system based on Support Vector Machine (SVM) classifier using Brown clusters along with POS tags and Clusters for identifying Named Entities has been developed by [7]. [17] has used POS tag as a state and developed a system using Hidden Markov Model (HMM) Classifier for identifying Named Entities. Later on in CMEE-IL, FIRE 2016 the Named Entity Recognition was reintroduced for Hindi-English and Tamil-English code mixed sentences. [4] have introduced a neural network for identifying the entities. They have used contextual features like prefix and suffix of words in context window along with some binary features. [6] have used a CRF trained on context features like prefix, suffix and character n-grams of context window words along with other binary features.

Chapter 3

Named Entity Recognition

Named Entities are names of famous places, persons, artifacts, names. For example, in the sentence *'I love staying in Manhattan'* the word *Manhattan* is a Named Entity(NE) as it represents a name of a famous location. The task of identifying Named Entities in a sentence is the task done by a Named Entity Recognizer (NER). Named Entity Recognition is very important in many applications such as sentiment analysis as it helps in removing words with no sentiment attached. Named Entiites also help in narrowing down the results in Document Retrieval. In English, there are a lot of systems designed to tackle the problem, prominent ones being Stanford NER tagger [5] and Illinois NER tagger [13]. Named Entity recognition is widely used in many applications such as Question Answering systems, Co-Reference Resolution , Query Labeling [3], Sentiment Analysis [1], Question Classification [2] etc.

3.1 Proposed Approach

The system had 3 subparts where the first part is rule based and the last part is based word matching using gazetteer. The second part is machine learning based. The model was a built using machine learning features shown in table 1. These features were trained on Decision Tree and Extremely Random Forest classifiers.

Features used throughout

The following features have been used in conjunction to the features specified in subsection 3.1 to develop the proposed new models developed. In all the proposed systems the NER gazetteer list feature has been removed. This feature requires a lot of Named Entity Data spanning from various domains and languages. The Named Entity list also requires constant update, hence the proposed approaches have removed the feature.

Table 3.1: Features used for building the previous system.

Sno	Feature
1	Presence of token in English Dictionary[a]
2	Prefixes of length 1 to 3
3	Suffixes of length 1 to 3
4	Capitalization related features [b]
5	Features based on presence or absence of special characters.[c]
6	Presence of emoticons
7	Token present in gazetteer list.
8	Is previous token a NE Tag.

[a] Dictionary created using Pyenchant [b] Features like starting letter capital, all letters capital, other letters capital.[c] like #,@,numbers, other symbols.

1. **Parts Of Speech (*POS*) :** The Parts of speech for each word in a sentence is determined and used as a feature. The POS tag is determined using CMU Ark tagger [9].

2. **Char N-grams (*c_ngram*):** Character n-grams of length 1 to 4 have been used as features. For example for the word *football* the c_ngrams would be an array { *f, fo, foo ,foot* }

3. **Word Normalization (*norm*):** The word is classified based on the following method.

   Each small alphabet is given *"a"* and capital letter is given *"A"*. Numbers are given *"0"* and symbols are given *"$"*. After replacing all characters with the following letters. If neighbouring characters are the same then they are removed. Eg: *'AAaa0'* $\rightarrow$ *'Aa0'*

   'AaAAaAA' $\rightarrow$ *'AaAaA'*

4. **Word Embeddings (*embed*):** Each word in the sentence was represented as a numerical vector of size 50. For embedding the words, a skip-gram based Word2Vec model was used. The word2vec was trained using a custom training data.

Machine Learning Model

As shown in Table 2, the proposed approach has the following new features added apart from the previous features as explained in subsection 3.1.

Among the following, the first two features are always retained and the gazetteer list feature was removed in all the models. The other features were not added to all the models. Based on the following features four different classifiers (random forests, decision tree, extremely random forests and support vector machines) were trained and tested.

Table 3.2: Additional Features used for building the Machine Learning System.

Sno	Feature
1	If word length greater than 4.
2	If the word starts with punctuation or digit
3	*POS* tag of the current word, previous word and previous to previous word [a]
4	*c_ngram* of the current word, previous word and previous to previous word
5	*norm* of the current word, previous word and previous to previous word
6	*embed* of the current word, previous word and previous to previous word.

[a] Used *CMU Ark Tweet Tagger* for this purpose

Neural Network Model

In this approach a Neural Network is used to solve the problem of Named Entity recognition. In machine learning techniques, a better model requires a lot of handmade features.At times, this approach may not work well because finding new features requires a lot of brainstorming. This problem can be solved by using a neural network model. A Neural Network can detect any complex relationships between inputs and outputs, thus helps in reducing the number of features required by the model.

The proposed Neural Network model has only one hidden layer of 20 nodes along with an input and output function. This layer helps in determining different secondary features like capitalization, punctuation, alphanumeric, word length etc. with the help of the embeddings and the new features as explained in 3.2 section . The *ReLu* function was considered for activation function in both layers. The model used one-vs-rest approach. The input layer had a combination of features (in section 3.2).While building the model various combinations were experimented. The system was trained by a gradient descent with cross entropy objective function using *RMSprop* optimization algorithm.

Deep Learning Model

In this proposed approach, a LSTM network is used for identifying Named Entities. LSTM is a better version of the recurrent neural network as it has the concept of forgetting unwanted information.Let w_t represent the word features of the t^{th} word in a sentence. The word features includes some of the features mentioned in section 3.2 such as word vectors, Character N-grams, POS Tags, and Normalized word class. A context window X_t of size *2n+1* for w_t is considered as a concatenation of $w_{t\text{-}n}$, $w_{t\text{-}n+1}$, ... $w_{t\text{-}1}$, w_t, w_{t+1}, ... w_{t+n}. In the proposed system, context window X_t is sent as input to model. The input X_t is passed along with $h_{t\text{-}1}$ to form the input gate activation function i_t and forget gate activation function f_t. Similarly candidate state value is also calculated.

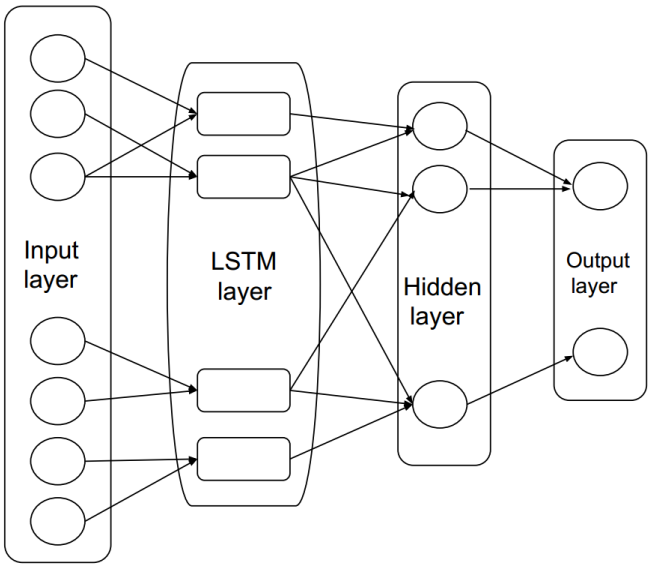

Figure 3.1: Proposed Deep Learning Architecture

$$i_t = sigmoid(W_i X_t + U_i h_{t\text{-}1} + b_i) \tag{3.1}$$

$$C_t = tanh(W_c X_t + U_c h_{t\text{-}1} + b_c) \tag{3.2}$$

$$f_t = sigmoid(W_f X_t + U_f h_{t\text{-}1} + b_f) \tag{3.3}$$

These activation functions are used to calculate the current memory state M_t and with the new state of the memory cell we can calculate another element in the model.

$$M_t = i_t * C_t + f_t * M_{t\text{-}1} \tag{3.4}$$

$$O_t = sigmoid(W_o X_t + U_o h_{t\text{-}1} + V_o M_t + b_o) \tag{3.5}$$

$$h_t = O_t * tanh(C_t) \tag{3.6}$$

This computation is done for all the words in a sentence. The new created elements are passed though a hidden layer of neural nodes. the idea behind using an LSTM and hidden layer is to prune the unwanted features by forgetting them and passing only the relevant features to the output layer where the prediction happens. In th proposed system the hidden layer has a size of 35 nodes and the output The output follows a one-vs all pattern to determine the particular tag of Named Entity. The system is trained for minimum cross entropy error and *RMSprop* as gradient descent optimization function.

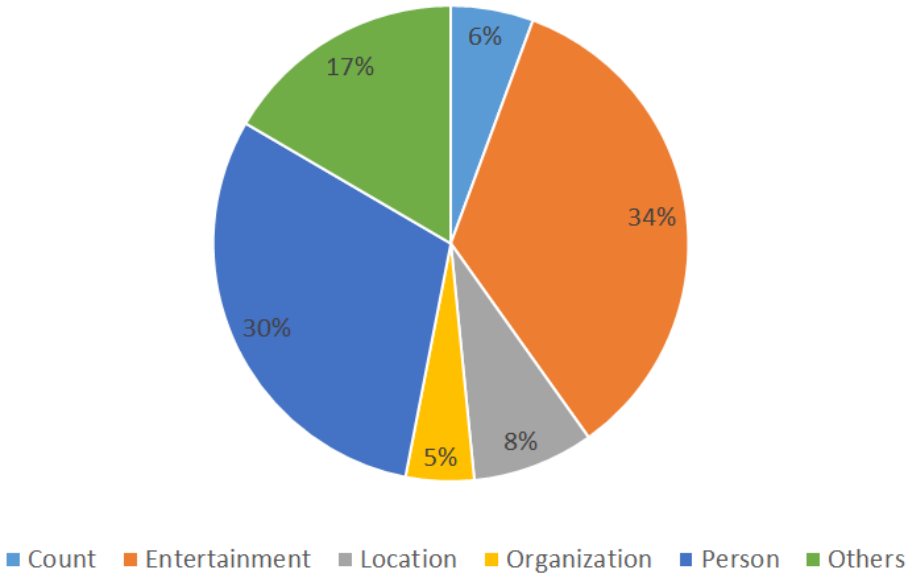

Figure 3.2: Chart depicting the distribution of tags in the training data

3.2 Dataset Analysis

The data set used for training the system was part of the task CMEE-IL of FIRE 2016 [11]. The dataset contained code mixed sentences in Hindi-English and Tamil-English but, the proposed systems are trained and tested only on Hindi-English dataset. The dataset consists of 2700 tweets in the Hindi-English corpus and 3200 tweets in the Tamil-English corpus. Most of the Hindi-English corpus tweets are already romanized in the dataset but the same didn't follow for the Tamil-English corpus. It had a mixture of both Tamil script and romanized script. For the proposed system only Hindi-English dataset has been considered. The dataset has 22 tags present. As shown in Figure 3, The 5 major tags present in the dataset are *Count, Entertainment, Location, Organization and Person.*

In the dataset, NE tags *Person*, *Entertainment* and *Location* constitute majority of the tags in the corpus. *Person* tag represents names of popular film stars, politicians, news anchors and social media celebrities. The *Entertainment* tag consists of names of famous serials, TV shows and movies. The Location tag consists of names of popular cities, towns in India and names of countries. *Location* tag consists of names of cities, countries, states etc.

Organization is another tag present in the dataset which consists of names of organizations. The count tag consists of numbers other than dates and years. The rest of the tags are present in small quantities and hence are merged into *Others*. *Others* consist of

tags *Artifacts, Locomotive, Date, Disease, Facilities, Time, Materials, Year, Month, Living Things, Money, Period etc.*

As observed, the tags *Entertainment, Location, Organization* and *Person* are present in larger amounts while the rest of the tags are present in very less quantities for which training a classifier is not possible. Hence, they are excluded from the dataset, as a result only the above four tags are considered for detection.

3.3 Experiments

All the proposed models were trained on the CMEE-IL dataset [11] using a train to test ratio of 70:30. The Stanford NER tagger [5] was considered as a base system for all the models. Models were evaluated on the basis of four parameters Precision, Recall, F-Measure and Accuracy. In general, there exists a trade-off between Precision and Recall for any two models. Hence, in order to determine the better model the geometric mean of both the parameters (F-Measure) was considered.

In case of the first model, three systems were designed with variations of features. The first system was based on the common features and word embedding. In the second system character n-grams were also appended. In the third system, instead of character n-grams, word normalization were used. Each of the system was trained with four different classifiers - decision tree, extremely randomized trees, random forest and support vector machines(SVM).

For the Neural Network and Deep learning models, all combinations of the features in section 3.2 were sent as input features to the models and their respective systems were tested. The best five systems from each model were considered for evaluation.

Results

As shown in Table 3, all the machine learning classifiers have better results compared to the base NER system but, the third system outperformed all the other systems in most of the parameters. When the F-Measure was compared among the classifiers, it can be clearly concluded that the random forest performed the best among the four classifiers for all the three systems. One possible explanation for this result can be the features used. The current machine learning features may not be completely linearly separable hence the SVM had lesser performance compared to the Random Forest. Extremely Random forests also performed fairly well compared to the other two classifiers as shown in Figure 4.

For the Neural Network model explained in section 3.4 the following results were obtained as shown in Table 4. The second system (*embed+POS*) performed the best among the other

Table 3.3: Results of various Machine Learning Models

Model	Precision	Recall	F-Measure	
Previous System	0.589	0.369	0.424	0.967
Stanford Tagger	0.3745	**0.618**	0.433	0.960
embed				
Decision Tree	0.445	0.473	0.456	0.957
Random Forest	**0.942**	**0.502**	**0.583**	**0.970**
Extra Tree	0.9399	0.4965	0.5799	0.968
SVM	0.693	0.452	0.511	0.969
embed+c_ngram				
Decision Tree	0.4596	0.474	0.466	0.962
Random Forest	**0.944**	**0.508**	**0.589**	**0.971**
Extra Tree	0.9276	0.483	0.558	0.968
SVM	0.616	0.4325	0.489	0.961
embed+norm				
Decision Tree	0.4615	0.4757	0.467	0.963
Random Forest	**0.951**	**0.497**	**0.573**	**0.971**
Extra Tree	0.645	0.398	0.462	0.969
SVM	0.472	0.4476	0.415	0.903

Table 3.4: Results of the various Neural Network models

Model	Precision	Recall	F-Measure	
embed	0.65496	0.46936	0.53055	0.96931
embed+POS	0.62866	**0.50514**	**0.55111**	0.97068
embed+norm	0.66035	0.487833	0.54334	**0.971049**
embed+ c_ngram+ norm	0.65054	0.41501	0.4738	0.9624
embed +c_ngram	**0.67875**	0.42994	0.49556	0.963251
Stanford Tagger	0.3745	**0.6179045**	0.432871	0.9600972

systems (based on F-Measure). It has been observed that all the neural network systems performed better than the base system (Stanford NER tagger).

Table 5 shows the performance of LSTM models compare to the base system. All models had great accuracies and F-Measures compared to the base system. The models *embed* and *embed+norm+c_ngram* had better results among the rest. The usage of LSTM helped in increasing the accuracy of the system. Infact it has the best accuracy among all the systems

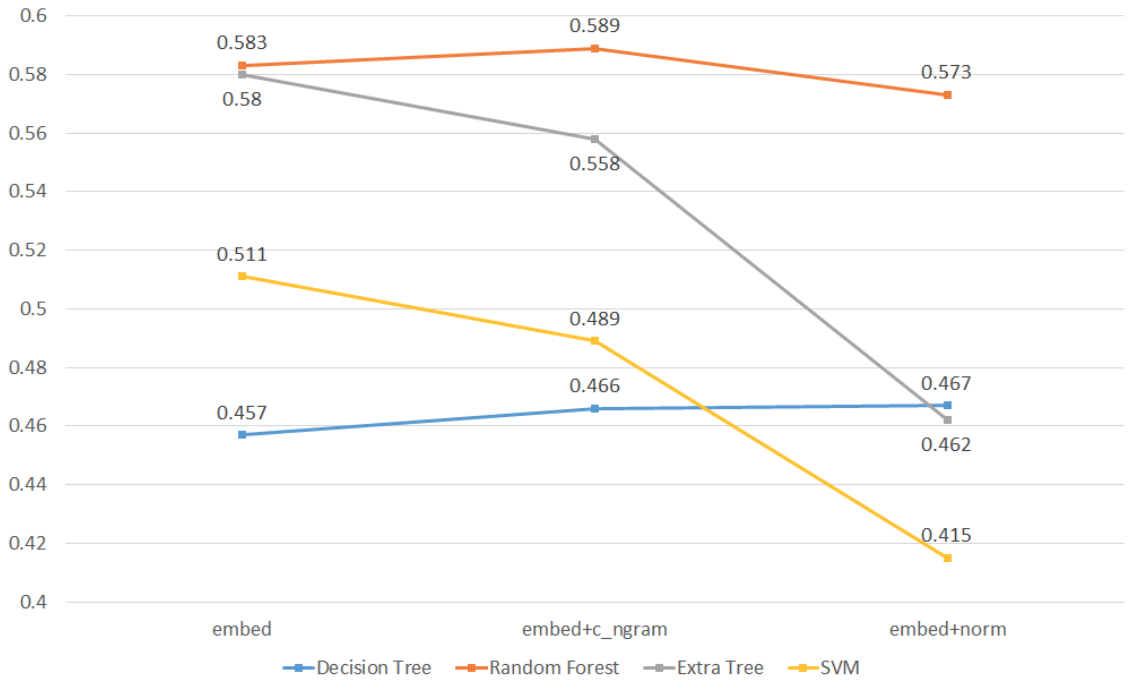

Figure 3.3: Chart comparing the performance of Machine Learning Classifiers

Table 3.5: Results of the various Deep Learning models

Model	Precision	Recall	F-Measure	
embed	0.62721	**0.51109**	**0.55698**	**0.972059**
embed+ c_ngram	0.6574	0.4459	0.5123	0.964334
embed+ norm	0.65741	0.44593	0.5123	0.964334
embed+ c_ngram+ norm	**0.68069**	0.432959	0.502936	0.96412
embed +norm + POS	0.647804	0.4999	0.551042	0.972059
Stanford Tagger	0.3745	**0.6179045**	0.432871	0.9600972

in all models.

As shown in Figure 5, there is a clear drop in precision from the machine learning systems to the rest of the systems, but all the systems had performed better than the base Stanford tagger. In case of the Recall, the base system performed better than the other systems. The comparison of Accuracy and F-Measure is shown in Figure 6. All the systems had similar accuracies but better than the base system. When we compare F-Measures, the Random forest model *rf+embed+c_ngram* performed the best. Even here the same trend of having better results than the base system continues. The machine learning models had better

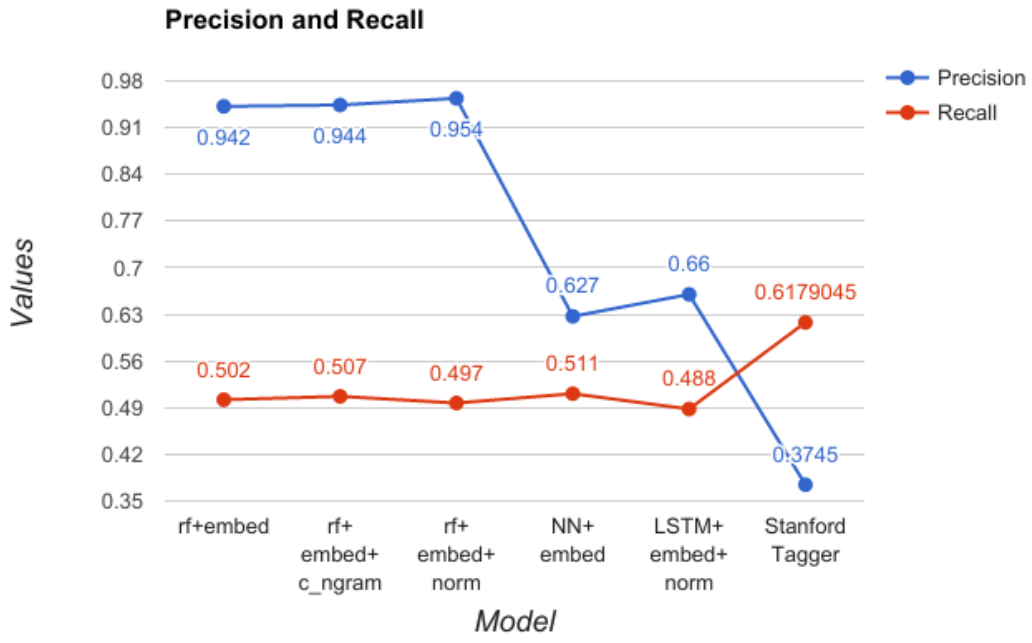

Figure 3.4: Precision and Recall Comparison of Different Systems

results compared to the neural network and deep learning models in terms of F-Measure but the vice-versa is observed in case of accuracy.

Error Analysis

The Machine Learning model used POS Tags as feature. The tags were provided by *CMU Ark Tagger*[13] but, the tagger was designed to tackle only social media texts in English. A POS Tagger for Code- Mixed texts would probably be helpful in improving the system. There might be a possibility of having better results if there was more data for training the deep learning models as deep learning models and neural network models tend to perform well with large data.

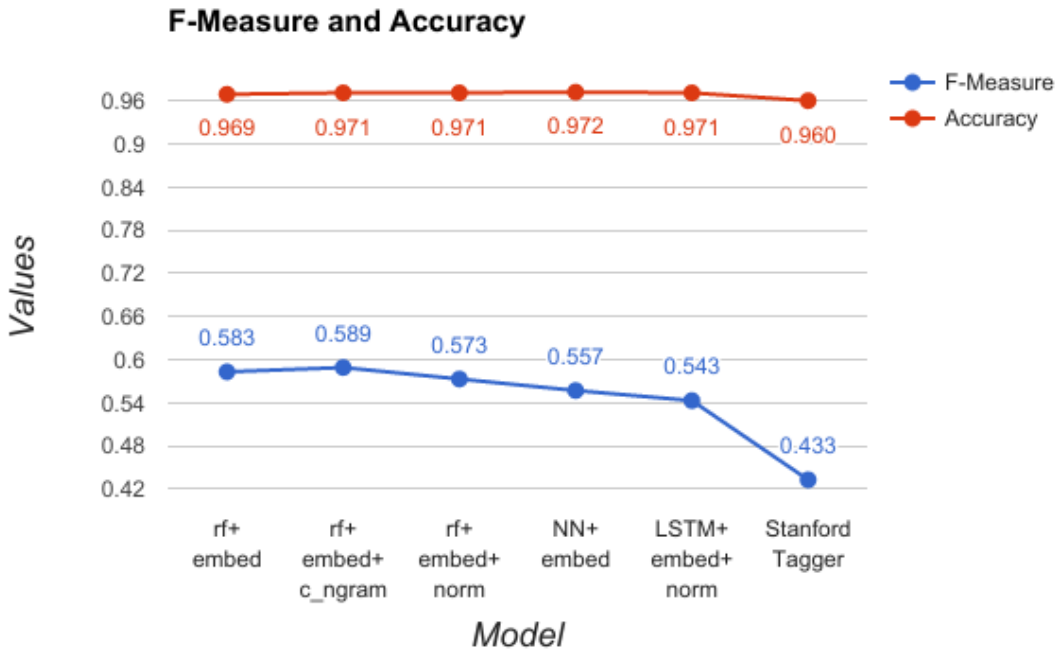

Figure 3.5: F-Measure and Accuracy Comparison of Different Systems

Chapter 4

Parts Of Speech Recognition

4.1 Proposed System

Three methods have been proposed for Parts of speech Tagging

Previous System

The previous system consists of a set of Features that comprise the feature vector for each token. The following are the features used.

1. **IsNumberPresent:** This is a binary feature which check for the presence of Digits in the given token.It returns 1 if present and 0 if not present.
2. **IS HashTag Present:** This is a binary feature which check for the presence of Hashtags in the given token. It returns 1 if present and 0 if not present.
3. **IS@Present:** This is a binary feature which check for the presence of the symbol '@' in the given token. It returns 1 if present and 0 if not present.
4. **IsSymbol Present:** All Characters which weren't either digits or numbers or @ or hashtag were considered to come into the category of symbols. This is a binary feature which check for the presence of the symbols in the given token. It returns 1 if present and 0 if not present.
5. **Language of Prev word(L-1), Prev Prev Word(L-2), Current word(L0) Next Word(L+1)** were considered as they help in analysing the way the words were used.
6. **POS Tag of Prev word(P-1), Prev Prev Word(P-2)** were considered as they help in analysing the structure of the sentence.

7. **Position of Word in sentence:** This is a numerical feature. There were many ways of representing this feature. We used the formula

$$PositionofWord = \frac{NoofWordspresentbeforethecurrentword}{Totalnoofwordsinthesentence} \tag{4.1}$$

8. **Prefix of the Token(Pr1, Pr2, Pr3):** Presence of most common Prefix is considered as a feature. Prefixes of length 1 to 3 were considered. The list of most common prefixes were extracted from the given test data.

9. **Suffix of the Token(S1, S2, S3):** Presence of most common Suffix is considered as a feature. Suffixes of length 1 to 3 were considered. The list of most common prefixes were extracted from the given test data.

The Listed features were extracted for every token in a sentence and feature vectors were built. Finally the List of all feature vectors were extracted and trained using a Random Forest classifier using scikit-learn.

Features used throughout

In all the proposed approaches the following features were used in common.

1. **Word Embedding (*embed*):** Each word in the sentence was represented as a numerical vector of size 30. For embedding the words, a skip-gram based Word2Vec model was used. The word2vec was trained using data extracted from various sources.

2. **POS Dictionary (*dict*):** A dictionary containing the first and second probable POS tag is designed. The dictionary is initially Empty and the dictionary gets updated after the POS prediction for a given word. The concept is a dictionary is very useful in cases where a given word mostly has only one possible POS tag every time. For example, consider the following sentence

   DiCaprio is an excellent actor.

   In this sentence the words *DiCaprio*, *actor* and *excellent* have the Parts Of Speech Noun, Noun, Adjective respectively. These words will always have the same POS tag when used elsewhere. Hence when stored in a dictionary they tend to show the same prediction.

3. Apart from the above two features, **Char N-grams (*c_ngram*)** and **Word Normalization (*norm*)** have also been used. The explanation for the following features has been explained in section 3.1 .

Tree Based Approach

Parts Of Speech(POS) tagging has a lot of anomalies. so to tackle these abnormal rules tree based classification would be a better fit. In this Approach, along with the proposed new features 2 binary features and 1 numeric feature have been used. They are given below,

- Presence of any emoticons in the word.
- Check if the starting letter is an alphabet.
- length of the word

Feature vectors were generated for each word in a sentence. These feature vectors were trained on 3 tree based models namely Decision Trees, Random Forest, Extremely Randomized Trees.

Conditional Random Fields

In this proposed approach, a Conditional Random Field Classifier is used for identifying the Parts Of Speech tags for every word in a sentence. The proposed CRF model labels a tag t_i to a word w_i in a sentence S based on the sequence (T_{i-1}) of tags $\{t_{i-1}, t_{i-2}, \ldots. t_{i-k}\}$ by calculating the Conditional probability $P(y_i/ T_{i-1})$, where $y_i \varepsilon$ Y, the list of all POS Tags.

The feature function $\boldsymbol{f}$ for each word in a sentence is a binary function based on the Sentence S, sequence T_{i-1} and position k. The score of the sequence l ($t_i \cup T_{i-1}$) is calculated by considering the weighted sum of the feature functions for all labels along all the words in the sequence.

$score(l/T_{i-1}) = \sum_{j=1}^{m} \sum_{k=1}^{i} \lambda_j * f(S, k, T_{i-1})$

Here, m represents the number of POS tags and n represents the size of the sequence. Then the conditional probability is calculated as follows.

$P(y_i/T_{i-1}) = \frac{exp[score(l/T_{i-1})]}{\sum exp[score(l'/T_{i-1})]}$

For this approach, the following features were used to calculate feature vector for each and every word in a sentence.

- Word length, position of word in sentence
- *norm* of all words in context window of size 3 [-1,0,1]
- Binary features like is capital, is digit, is titlecase, is uppercase, presence of ,@or other symbols in words of context window
- Language of all words in context window.

Deep Learning Approach

For this approach, it takes into account the POS Dictionary not as a feature but as an output with majority given to the model. There were three models developed.

- Neural Network consisting of 2 hidden layers of dimensions 64 and 32 nodes were considered apart from the input and output layers. Except the output layer all other layers have ReLu as their activation function.The last layer has sigmoid Activation function.
- LSTM model consists of one LSTM layer along with a hidden layer of 250 nodes followed by an output node. All the activation functions are ReLu. A dropout of 0.2 was considered in the hidden layer.
- A Simple RNN model was designed similar to the LSTM model with the change in first layer.

4.2 Dataset Description

The dataset used for the training the system is obtained from ICON 2016. The dataset has code mixed sentences from three languages Telugu, Hindi and Bengali. For each language, the sentences are obtained from Twitter, Facebook and Whatsapp.

As shown in Table 4.1, the following tags were considered while designing the system. In total 12 tags were considered.The dataset was partitioned into a training to testing ration of 70:30 .

As shown in Table 4.2, the distribution of the POS tags in the training and testing partition is almost similar except for a few tags. In the training data there are a lot of residual tokens present followed by nouns with a slightly less no of tokens. In the case of testing data, the no of nouns present is very high compared to the no of residual tokens.

The sentences in the dataset of any language always had majority of the tokens to be their respective languages and the token were already transliterated to English. It is followed by the tokens from English language and Universal tokens.

The datasets of all the three languages contained inconsistencies in improper classification of words into language tags. Universal tags were classified incorrectly in many cases. For example, In The Telugu corpus, the names of some movie stars were classified as universal while universal was given only to token having symbols @ or #.

Table 4.1: Parts Of Speech Tags present in the dataset

Sno	Tag	Description
1	G_N	Noun
2	G_PRP	Pronoun
3	G_V	Verb
4	G_J	Adjective
5	G_R	Adverb
6	G_SYM	Quantifier
7	G_PRT	Particles
8	G_X	Residual (punctuation, symbol,unknown etc.)
9	CC	Conjuction
10	PSP	Preposition
11	DT	Determiner
12	$	Numerical

Table 4.2: Distribution of POS tags in training and testing partitions

Tag	Count in Training set	Count in Testing set
G_N	5917	4935
G_X	6073	2826
G_V	1676	1218
G_J	953	537
PSP	740	507
G_PRP	689	427
G_PRT	419	263
DT	347	253
G_R	386	242
$	165	158
CC	194	150
G_SYM	47	15
null	270	8

4.3 Experiments

All the proposed systems were trained on the ICON 2016 dataset by partitioning the dataset into a training to testing ratio of 70:30. The previously submitted system for ICON 2016 was considered as the base system. All the models developed were evaluated on the four parameters namely Precision, Recall, F-Measure and Accuracy. For deciding the better model, F-Measure was considered as standard as it is the geometric mean of Precision and Recall.

In case of the models in First Approach, all the models have the word embedding (*embed*) and the previous features in common. The rest of the features such as norm, dict,c_ngram are used in various combinations.Finally the models performing better than the previous system are shown in Table 4.3 .

In case of the Neural Network & Deep Learning approach, all the models were trained on *embed, dict and c_ngrams* as features. The difference lies only in the basic system which was explained in previous sections. For the final approach using Conditional Random Fields, the input only consists of the features described in the respective section. It doesn't contain any of proposed common features such as *dict, embed, c_ngram* and *norm.*

Results

The Table 4.3 depicts the results of five better systems among all possible combination in the first approach. In overall, there isn't much change in the machine learning models proposed. But these features have brought a lot of change in Neural Networks and Deep Learning models as shown in Table 4.4. One possible explanation might be that the neural networks are able to extract more information than the normal binary features designed in the machine learning approach. The best performing system among all the proposed models is the conditional Random Fields. The best part of the CRF is that it doesn't even any word embedding, character n-grams or any dictionary. One possible explanation for the performance is the fact that CRF predicts the tag of a word based on the sequence of tags already tagged which is close to reality.In the other systems, the features are given such that the classifier knows the context of the word. In case of CRF that context is already being present with the classifier.

Table 4.3: Results of various Machine Learning Models

Model	Precision	Recall	F-Measure	
Previous System				
Decision Tree	0.4988	0.4918	0.4922	0.6618
Random Forest	0.5814	0.4778	**0.5174**	**0.6909**
Extra Trees	0.5741	0.4747	0.5135	0.6862
SVM	0.5765	0.3369	0.3936	0.6491
embed +ngram				
Decision Tree	0.4382	0.4493	0.4400	0.5746
Random Forest	0.4430	0.6604	0.5126	0.6426
Extra Tree	0.4627	0.6673	0.5294	0.6508
SVM	0.1651	0.2084	0.1639	0.4376
norm+embed+ngram				
Decision Tree	0.4472	0.4626	0.4500	0.5771
Random Forest	0.4414	0.6562	0.5103	0.6463
Extra Tree	0.4665	0.6543	0.5312	0.6532
SVM	0.1206	0.1161	0.0846	0.2831
embed+norm				
Decision Tree	0.4342	0.4594	0.4373	0.5656
Random Forest	0.4549	0.6505	0.5213	0.6501
Extra Tree	0.4724	0.6436	0.5335	0.6468
SVM	0.1727	0.3744	0.1787	0.5409
embed+dict				
Decision Tree	0.4660	0.4824	0.4654	0.5770
Random Forest	0.4640	0.6481	0.5239	0.6407
Extra Tree	0.4749	0.6489	**0.5347**	0.6501
SVM	0.2659	0.4932	0.2857	0.5669
embed+ngram+dict				
Decision Tree	0.4593	0.4792	0.4611	0.5798
Random Forest	0.4638	0.6527	0.5259	0.6442
Extra Tree	0.4745	0.6513	0.5333	0.6510
SVM	0.1198	0.2356	0.1128	0.3299

Table 4.4: Results of other Models

Model	Precision	Recall	F-Measure	
Previous Best	0.5814	0.4778	**0.5174**	**0.6909**
NN+*c_ngram+embed+dict*	0.7804	0.4797	0.5757	0.6644
LSTM+*c_ngram+embed+dict*	0.7662	0.4801	0.5521	0.5208
RNN+*c_ngram+embed+dict*	0.7662	0.4801	0.5521	0.5208
CRF	0.7327	0.5806	**0.6353**	**0.6855**

Chapter 5

Conclusion

In my Thesis, First I propose three different approaches to tackle the problem of Named Entity Recognition. The first approach uses a machine learning approach such as Decision Tree, Random Forest, Extremely Random Forest and Support Vector Machines.The second approach uses a Neural Network with hidden layers and the third approach uses a LSTM for the tagger. The Machine learning approach had better F-Measure and Precision compared to the other models while the other two approaches had better Accuracy (about 97%).

Work can be further improved by building a POS Tagger for code mixed sentences as there is a possibility of having better results. The proposed systems used a basic *word2vec* to create word embeddings. Experimenting with other embedding styles (like bilingual embeddings, character embeddings etc.) could probably benefit the systems performance.

Similarly, I propose three different approaches for Parts Of Speech Tagging. The first approach consists of Machine Learning models trained on a set of features. The Tree based classifiers such as Decision Tree, Random Forest and Extremely Random Forest perform better than the rest. In the second approach, there were three models developed based on neural network and deep learning methods. The first model was using a four layered Neural Network along with a dictionary.the second model was an LSTM model along with a hidden layer in combination with a dictionary. The Third model was a simple RNN instead of LSTM. The Neural Network model performed the best among the three models. In the final Approach, a Conditional Random Fields Classifier was used to tag the parts of speech of words. Among all the approaches proposed, the CRF outperformed all the models with a F-Measure of 63.53% Work can be improved by tagging the words with the help of linguistic rules developed for code mixing. The proposed systems did not use any clusters and tree banks, adding that feature might help in increasing the accuracy of the system. Also, By Normalizing the sentences there might be a possibility of better classification.