Chapter 4

SENTENCE LEVEL NAMED ENTITY IDENTIFICATION AND CLASSIFICATION

4.1 Introduction

This chapter illustrates our model of multiword named entity identification and classification system. What is developed is an extensive multi word named entity identification and classification system for Telugu. Named entities are subcategories of nouns. In Indian languages it gets very difficult to make a distinction between common nouns and proper nouns the reason they are very similar. We need proper restructuring tool which would enable proper noun identification and classification.

This chapter describes what we have tried to develop i.e. our computational model of named entity identification and classification that involves a number of features and heuristic. It also speaks about the performance assessment of NE systems and provides some quantitative evaluations of our system. It is already known that the Telugu language is highly inflectional. Another fact is that about 50-70% of words in any written text occur in inflected and/or derived forms. Inflection often results in elision and also declination of some akSharas. Due to this factor and also the high occurrence of inflectional and derivational words, it gets very difficult to analyze even the most minutes of texts without a suitable morphological analyzer (MA) and an associated lexicon or dictionary. The adoption of foreign language words from Hindi, Sanskrit, English and other languages further adds on to the already existing problems in the analysis of Telugu words.
4.2 Our Model for Telugu

This NERC module has been built to identify and categorize named entities in any given text. This chapter makes an attempt to identify multi-word nested named entities. Since names of people and organizations are mostly multi word named entities we have chosen to identify named entities at multi token level as in contrast to the token level. Once these words are taken together they form a named entity, however it should be noticed that individual words may or may not be named entities. Example: “viplava racayitala saMghaM”

The above words are individual Telugu words which have reference to the Telegu dictionary and are common nouns. Only when taken together they form an organization name.

Drawing out of word level features to produce patterns and test level data have been reference from Telugu-wiki and newspapers.

INPUT:

The Input to the NERC system is a plain text document which has been divided into sentences either typed in Telugu using roman script [Appendix A] or documents typed in Telugu language using Telugu script using UTF-8 [Appendix A] encoding and further converted to roman script.

OUTPUT:

The generated Output of the system is a file that contains NE, English words i.e. English words written in English script and dictionary tags tagged words.

It is to be noted that Named entities are open class words. As part of our research we had picked up 10 documents randomly and manually checked for named entities and spotted out that the existence of named entities in a document varies from 1% to 18% of the whole text depending on the nature of document that averages to
about 6% of named entities per document. This makes identification of named entities necessary in the field of Information Extraction.

Progressing ahead we next checked on the features of NE, which differentiates it from common nouns.

The following features of NE were then deduced.

1. Named Entities do not take plural forms unless they are preceded by another named entity that belongs to the same type.
2. Named Entities act as adjectival modifiers.
3. Named Entities take all possible case markers.
4. Ordinals and person entities comprise another named entity.
5. Named Entities do not fall in line behind quantifiers.
7. Named Entities may be succeeded by reflexive pronouns.
8. Named Entities are followed or preceded by verbs.
9. Named Entity and verb compounds are existent.
10. Person entities rarely assume the form locative case markers.
11. Organization entities do not include verbs or adverbs.

The above features helped us in recognition and classification of NEs and to eliminate non-NEs identified as NEs.
4.3 NERC Architecture

![NERC Architecture Diagram]

4.4 Non-Named Entity Identification

This non-Named entity identification includes identifying common Telugu words which can be found in dictionary, Telugu words that have undergone morphological inflections, numbers and alphanumeric words. The procedure that is adopted assumes three forms. Different components represent these filters and each filter analyses the untagged word paying no attention to the tagged words.

Stage-1:

Stage-1 is developed as a combination of non-Named entity identifiers.
Dictionary

Assumption 1: Words that can be referred to via the dictionary are not NEs or common Telugu words. A Dictionary files contains headword and its part of speech. All the words of input text file that find reference with dictionary words are categorized and their respective tags assigned. A Dictionary file contains 41,000 words and above. Also to mention that the Dictionary words which are inflected with case markers are also recognized.

Frequent Word List

Using reference to our corpora what we have done is extraction of words that occur more than 500 times. With the exception of nouns and adjectives all other words were hand filtered and added to frequent word list. Nouns and adjectives may occur as a part of named Entity and therefore could be ignored. This list contains words which are inflected i.e. the words which are not present in dictionary. Size of this list is 982 words. Words that are in the frequent word list cannot be NE.

Common Noun and Verb Suffixes

List of common noun suffixes which do not generate NEs are prepared. If it is identified that a word ends with any of the suffixes from the list it is identified as a common noun. These suffixes therefore help us in identifying common nouns. The number of the list is reference to consist of 100 words.

A List of verb suffixes was also compiled from dictionary. If it was found that a word matches with this list or ends with any of the words from the dictionary it is identified as a verb. These words help in identifying verb compounds.

Symbols

Words which are symbols i.e. special characters or combination of special characters are identified.
Stage-2:
In stage-2, we check for morphological inflections of a word.

4.5 Morphological Analyzer

Assumption 2: Words that are analyzed by morph are not NEs.

Morphological analyzer [iiit] developed at University of Hyderabad over the last few decades has been used to obtain the root word and the POS category for the given word. Morphological analyzer helps us to recognize inflected forms which is otherwise not be available in the dictionary [MK97].

Ex: For the word “puurtikaadu” its root word is identified as “puurti” and its tag is assigned as N words from the morphological analyzer that contains

- Named entities
- Loanwords.

Loan words are borrowed from other languages and incorporated into Telugu language. On similar lines, Telugu has also borrowed many words from the English language and most of the regular usage and domain specific words have been borrowed from English. Some of these words constitute named entities, such as kaMpenii (company), paarTii (party). Then there are other loan words which are used by individuals like common Telugu words.

e.g. Teliphoon (telephone). There are other domain specific words such as vairas (virus), eMDooplaasmik (endoplasmic).

Examples: skuul => school

- elakTraan => electron
- boorD => board
- DaakTar => doctor
English Vocabulary.

The Telugu-wiki documents contain English words as a part of text or to support the words which cannot be understood. This supportive English text is written in brackets.

Ex: cassettes, subjects

- prati (fascimile)
- vistiirNaM (area)
- kaMThasvaraM (Tone)

Assumption 3: Since they are not formed using Telugu aksharas, these words are not NEs.

External saMdhi words

Words that contain external saMdhi are not handled by morphological analyser.

Ex: vaaraMdarikii, atanipeeru, viraamamerugaka

Nouns, Verbs, Noun compounds and verb compounds.

Few other nouns and compounds were also not analysed by morphological analyser.

Ex: satkaaraM, baalyamitruDu, sudhacellelu, bhiimaarjunulu

Spelling errors

Ex: kaaryadarsi => kaaryadarshi
kadhaanika => kathaanika

Stage-3:

In stage 3, we check for English word and orthographical information of words. If the word contains numbers then signifies it is a Named entity.

**Numbers or Alphanumeric Word**

Named entities do not contain numbers and alphanumeric characters and this information is useful in recognizing named entities at the contextual level. All the words that contain numbers are identified and tagged. Likewise, words that contain...
numbers are checked for the year pattern are identified and tagged as a year. This information helps in identifying name entities and all other words are tagged as numbers and alphanumeric words.

Our Named entity recognition and classification (NERC) system is built to identify people, location and organization entities which imply any word that contains numbers cannot be a part of a Named entity.

4.6 NERC at Word Level

The next process that is followed is checking for Untagged words from earlier non NE filers and verifying if they are named entities. We tried to identify person, location and organization names using NE gazetteer, NE suffix gazetteer and word level features such as punctuation marks, case markers to identify and classify named entities and then tried to handle Named entity variations that occur with the last akshara of word.

Ex: siitaaraaM, siitaaraam

maheesh. MaheeSh

The next method adopted was to identify abbreviations which could be or part of a person, location or organization entity.

Ex: Ti.Di.pi., en.Ti.raamaaraavu, es.aar.nagar. The identification of Person entities is known by the postpositions which are particular to that person. Words that end in consonants are tagged as loanwords, which are identified as NEs or as unknown words based on the contextual information in the next steps.

SaMdhi Filter

SaMdhi is the fusion of sounds across word boundaries and the alternation of sounds due to neighboring sounds or due to grammatical function of adjacent words. SaMdhi can be both internal and external. Internal saMdhi features the alternations of sounds
within words at morpheme boundaries. External saMdhi refers to changes found at word boundaries.

External saMdhi analyser has been developed as a part of a spelling error detection module [umh]. It identifies saMdhi in a word and checks if a word contains external saMdhi based on saMdhi rules defined and checks if the saMdhi splits are valid or not by checking the resulting two words with dictionary and correct words analyzed by morph. If the word split is valid then it is assigned a saMdhi tag and treated as common noun.

Examples: paacceppulu=paata + ceppulu
pattappulu=padi + tappulu
reeppodduna=reepu + podduna
nivaasabhuumi=nivasa + bhuumi
vyaktipeeru=vyakti + peeru
sarvaadhunika=sarva + aadhunika

Assumption 4: The words that undergo external saMdhi are assumed to be common nouns.

External saMdhi analyzer is placed after NERC word level module because named entities also contain saMdhi. For example, consider “raamanna”, this word can be split into two words “raamu” and “anna”.

4.6.1 Proper Noun Identification from the Words Identified as Common Nouns by Non-NE Filters

The suppositions (1) & (2) made at stage-1 and stage-2 need not be true in the real case as in a few cases common nouns and adjectives act as proper nouns based on the context. It should also be borne in mind that these words have already been tagged as common nouns. In this module we tried to identify the words that act both as a
common Telugu word and a proper noun as well. If a word has been identified as NE then its preceding and succeeding 3 words that are marked as nouns or adjectives by Telugu dictionary or morphological analyzer are checked for NE list and word features. If it appears that they act as NE a corresponding tag is assigned. Progressing ahead common and proper noun disambiguation is done in the next module.

4.7 NERC at Sentence Level

This segment chiefly deal with identifying person, location and organization entities as well as multi-word named entities. Named entities have been identified using contextual information. NE patterns are identified from the corpus to recognize named entities and rules are constructed based on these patterns. All the tagged and untagged words are considered in this section and rules make use of part of speech and NE tag information of previous and next words and word position. NEs recognized using this list and word features are now checked for their correctness using disambiguation and non-NE identification rules. (Here, non-NE identification means recognizing NE that is actually a Non-NE). Named entity features that were observed during earlier stages of this project helped in building disambiguation and non-NE identification rules. These rules make use of previous and next word POS information

The identified NEs are stored in another lexicon along with their category for future use. This will help if the next sentence contains the same word then tag will be assigned to it.

Disambiguation rules resolve the issue of NE ambiguity discussed in Telugu NERC issue. Non-NE identification rules can handle common and proper noun disambiguity to some extent.

Some common issue (6) is addressed here.
Reputation of the above process is done over varied iterations to extract patterns to identify more named entities and we checked nature of the documents is thereby checked. This process that thus developed a named entity database which includes 26,188 person entities, 26,468 location entities and 1600 organization entities. NE suffix gazetteer consists of 3,369 person entity suffixes, 211 location entity suffixes, 409 organization entity suffixes and 8 person designation suffixes which together constitutes 3,948 named entity suffixes. Tags used in NE gazetteer and suffix gazetteer and no. of NEs corresponding to each type are listed below.

Tags used in NE gazetteer:

<table>
<thead>
<tr>
<th>NE TAG Description</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE-PER-F Person Name</td>
<td>21,296</td>
</tr>
<tr>
<td>NE-PER-suf Person Suffix</td>
<td>1034</td>
</tr>
<tr>
<td>NE-PER_FN Person First Name</td>
<td>3806</td>
</tr>
<tr>
<td>NE-PER_Title Person Context</td>
<td>74</td>
</tr>
<tr>
<td>NE-PER_DES Person Designation</td>
<td>577</td>
</tr>
<tr>
<td>NE-LOC-F Location Name</td>
<td>26290</td>
</tr>
<tr>
<td>NE-LOC-suf Location Suffix</td>
<td>211</td>
</tr>
<tr>
<td>NE-LOC-PREF Location Prefix</td>
<td>2</td>
</tr>
<tr>
<td>NE-ORG-F Organization Name</td>
<td>1172</td>
</tr>
<tr>
<td>NE-ORG-suf Organization Suffix</td>
<td>412</td>
</tr>
<tr>
<td>NE-ORG-PREF* Organization prefix</td>
<td>18</td>
</tr>
<tr>
<td>NE-OTH* Miscellaneous names (Ex: iMDiyan)</td>
<td>202</td>
</tr>
<tr>
<td>NE-DIR* Directions</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 4.1 List of Tags used in Named Entity Gazetteer
*These entities help in determining multi-word named entities.

Tags used in NE suffix gazetteer:

**NE TAG Description SIZE**

<table>
<thead>
<tr>
<th>NE TAG</th>
<th>Description</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE-PER-S</td>
<td>Person suffix</td>
<td>3362</td>
</tr>
<tr>
<td>NE-LOC-S</td>
<td>Location suffix</td>
<td>221</td>
</tr>
<tr>
<td>NE-ORG-S</td>
<td>Organization suffix</td>
<td>409</td>
</tr>
<tr>
<td>NE-PER_DES</td>
<td>Designation suffix</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.2 List of Tags used in Named Entity Suffix Gazetteer

**4.8 Algorithms**

Below algorithms explain the design and flow of control of the tool developed.

**Non-NE Identification**

**Stage-1: (Dict_Check)**

Input: Plain text file divided into sentences in roman script.

Output: Dictionary, noun, verb and frequent words tagged file.

dict_file(String inputFile)

```
{
    //Read inputFile line by line
    s12 = dict_sent(line);
    //Write s12 to output file
}
```

String dict_sent(String sentence)

```
{
    //pre-process sentence
    // replace , with , and singlespace
    // replace ( with space and ( 
```
// replace ) with ) and space
// replace * with empty string
// replace tab space with single space
// replace double spaces with single space
// tokenize sentence
for each word in sentence
word = current word
if(word.length()>1)
word = stripSplChar(word); //remove special characters
if (word is found in either dictionary or frequent word list)
assign tag
else if (word endsWith noun suffix)
assign tag
else if (word endsWith verb suffix)
assign tag
else if (word endsWith case marker)
remove the case
if (root word is found in dictionary)
assign tag
else if (root word endsWith noun suffix)
assign tag
else if (root word ends With verb suffix)
assign tag
else if(word is a symbol)
assign tag
else assign ||UNK tag
//end of for loop
return tagged sentence;

Stage - 2: (Morph Filter)

Input: dict_file() method output file
Output: output file contains morph tagged words along with input file tags

morph_file(String inputFile)
{
//Read inputFile line by line
s12 = morph_sent(line);
//Write s12 to output file
}

String morph_sent(String sentence)
{
for each word in sentence
if (word contains “||UNK”)
if (word is analyzed by morph)
assign tag
else do not alter tag
else do not alter tag
//end of for loop
return sentence;
}

Stage - 3: (Number Filter)
Input: morph_file() method output file

Output: output file contains English, number tagged words along with input file tag

num_file(String inputFile)
{
    // Read inputFile line by line
    s12 = num_sent(line);
    // Write s12 to output file
}

String num_sent(String sentence)
{
    for each word in sentence
    if (word contains "||UNK")
        if ((word is found in English dictionary) || (word starts with a capital letter except 'T', 'D', 'Sh'))
            assign tag
        else if (word contains digits)
            assign tag
        else if (word is alphanumeric word)
            assign tag
        else do not alter tag
    else do not alter tag
    return sentence
}

NERC at Word Level: (Word_Tagger1)

Input: num_file() method output file
Output: output file contains NE tagged words along with input file tags

```java
run(String inputfile)
{
    //Read inputfile line by line
    s12 = run_sent(line);
    //Write s12 to output file
}

String run_sent(String sentence)
{
    for each word in sentence
    if (word contains "||UNK")
        tag=word_operations(word)
    if(tag is not empty string)
        assign tag
    else do not alter tag and store the word without tag in test_sandhi_word.lst
    else do not alter tag
    return sentence
}

String word_operations(String word)
{
    if word is found in NE list or generated NE list
        assign tag
    else if word is found to have NE variations
        assign tag
    else if word endsWith NE suffix
```
assign tag
else if word contains punctuation marks and is found to be NE
assign tag
else if word is an abbreviation
assign tag
else if word endsWith consonant
assign tag
else if word endsWith case marker and is found to be NE
assign tag
else if word endsWith person case marker
assign tag
}

String punctuation_mark(String word)
{
    wds[] = tokenize word on . (period)
    if wds.length > 1
        if (all the tokens of word are alphabets except last token)
            token = last token
            if the token is also an alphabet
                assign tag
            else if the token is a NE
                assign tag
            else if the token has NE variations
                assign tag
            else if the token ends With case marker
remove case
if the root word is found in NE list
assign tag
else if the root word ends With NE suffix
assign tag
else if the root word has NE variations
assign tag
else if the root word is an alphabet
assign tag
else assign “OTHC” tag
else
remove punctuation mark
if word is found in NE list
assign tag
else if word has NE variations
assign tag
else if word ends With NE suffix
assign tag
else if word ends with case
repeat case Check method
else if word is an alphabet
assign tag
else if word is an abbreviation
assign tag
else if word ends With consonant
assign “LOANWD” tag
else assign empty string
return tag
}

**Sandhi Filter: (Sandhi_Filter)**

/*All the unidentified words are checked for external saMdhi. All the words that contain valid external are tagged.*/

Input: run() method output file and test_sandhi_words.lst

Output: output file contains sandhi tagged words along with input file tags

sandhi_mark_file(String inputfile)
{
for each word in test_sandhi_words.lst
split the word into sandhi cases based on rules
if all the sandhi cases match with dictionary or morphology
output contains word along with possible sandhi cases
else store them in no_sandhi_words list.
//Read inputfile line by sentence
s12 = sandhi_mark_sent(sentence, no_sandhi_words list);
//Write s12 to output file
}

String sandhi_mark_sent(String sentence, List no_sandhi_words )
{
for each sentence, tokenize
for each **UNK tagged** word
if word is found in no_sandhi_words list do not alter UNK tag
else assign sandhi tag
return NE sandhi tagged sentence
}

**Proper Noun Identification from Common Noun (Word_Tagger2)**

The previous and next 3 words of NE or untagged (UNK)/English words are considered. These words are referred to as context words. If the context words are identified as common nouns or as adjectives by dictionary, or if they are analysed by a morphological analyser then they are also checked for the NE word feature. All these words identified as common nouns or as adjectives, or analysed by morphological analyser are then passed through word_operations() method of NERC at word level.

Input: sandhi_Fliter.class output file

Output: The output file contains R1D(if the word is found in dictionary and also acts as NE) and R1M (if the word is analyzed by morph and acts as NE) tagged words along with input file tags

```java
inputFile

//Read inputFile sentence by sentence
s12 =w_r1_sent(sentence);

//Write s12 to output file

String w_r1_sent(String sentence)
{
    for each sentence
tokenize
if the word which is identified by dictionary as noun or adjective is identified as NE by word_operations() method, NE tag is assigned.
```
else if the word that was analysed by morphological analyser is identified by word_operations() method as NE, tag is assigned.
}

NERC at Sentence Level

Input: Word_Tagger2.class output file
Output: NE tagged file along with input file tags.

//patterns are identified from data. Following rules were built based on observed patterns.
for each sentence
tokenize
Rules under s_r2_sent_Word(String) method are language oriented and makes use of specific words as well as tags
If a sentence contains any of the patterns described in Appendix C under s_r2_sent_Word(String sentence) method, then tag the word and if it is an unknown word or a loanword, sandhi word store the word along with its assigned NE tag in generated NE list
Else do not alter the tag
This method output is given as input to next method(s_r2_sent(String)).
Rules under s_r2_sent(String) method are based on tags and very few rules are based on words.
These rules are based on tag information of previous, current and next words.

Disambiguation rules for person and location and organization are then identified. Non NEs that were recognized as NEs from class Word_Tagger1.class are identified as NEs.
If the current word has any pattern described in Appendix C under s_r2_sent(String) method, then tag the word and if it was unknown word or loanword or saMdhi word store the word along with its assigned NE tag in generated NE list, Else do not alter the tag.

This method output is given as input to next method(s_r2_sent_Final(String)). Rules under s_r2_sent(String) method are based on tags and very few rules are based on words.

These rules are based on tag information of previous, current and next words. If the current word has any pattern described in Appendix C under s_r2_sent_Final(String) method, then tag the word and if it was unknown word or loanword or saMdhi word store the word along with its assigned NE tag in generated NE list.
else do not alter the tag.

Return the tagged sentence and write to output file.

**Sentence_Tagger2:**

Input: Sentence_Tagger1.class output file

Output: NE tagged file along with input file tags. //Final output.

for each sentence

tokenize

if current word has any pattern described in Appendix C under s_r2_sent_Final(String) method, then tag the word
else do not alter the tag

Return the sentence and write it to output file.
NERC:

/*All the above steps are iterated for each sentence in input text file. NEs identified are stored in NE dictionary which will help recognizing that NE in next sentences.
Input: Plain untagged text is divided into sentences and in roman script
Output: NE tagged word

main(){
for each line in input file
//pre-process sentence
// replace , with , and space
// replace ( with space and ( 
// replace ) with ) and space
// replace * with empty string
// replace tab space with single space
// replace double spaces with single space
s1=Dict_Check.dict_sent(sentence);
s1=Morph_Filter.morph_sent(s1); 
s1=Number_Filter.num_sent(s1)
s1=Word_Tagger1.run_sent(s1)
s1=Sandhi_Filter.sandhi_mark_sent(s1)
s1=Word_tagger2.w_r1_sent(s1)
s1=Sentence_Tagger1.s_r2_sent_Word(s1);
s1=Sentence_Tagger1.s_r2_sent(s1);
s1=Sentence_Tagger1.s_r2_sent_Final(s1);
s1=Sentence_Tagger2.s_r3_sent(s1);
write s1 to output file

//end of for

Print output filename

}

4.9 Results

The execution of the NERC system is appraised by implementing the system on test data. The Performance of NERC is then evaluated based on the F-measure score obtained. The capability of NERC to run on different platforms (Operating systems) is also taken into account. It is also necessary to specify the hardware and software requirements to run the particular application and the execution process is explained comprehensibly followed by the performance results tested on the files selected randomly from News paper and Telugu.wiki corpus.

Two standard measures, Precision and Recall are adopted. While precision (P) measures the number of correct NEs in the answer file (Machine tagged data) over the total number of NEs in the answer file, recall (R) measures the number of correct NEs in the answer file over the total number of NEs in the key file (gold standard). F-measure (F1) is the harmonic mean of precision and recall.

\[
P = \frac{\text{Number of correct tags assigned}}{\text{Total number of tags assigned}}
\]

\[
R = \frac{\text{Number of correct tags assigned}}{\text{Total number of tags in the annotated test corpus}}
\]

\[
F1 = \frac{2RP}{R + P}
\]

The current NERC system handles entities as multiword tokens. Though it is intricate to identify accurate named entity boundaries, we are still able to identify part
of the named entity. Partial matches are also considered correct in our analysis. Only a partial attempt has been made to focus on Nested entities. Nested entities have been built only for only names of people. Our main focus here however is to identify all person, location and organization entities present in the text.

The performance of the rule based NERC system has been presented over randomly chosen text files from News paper and Telugu.wiki corpus. The total number of words and number of named entities in the test data sets are given in Table-4.3 Performance of the system is measured in terms of F-measure. The recognized named entity must belong to the accurate type or category for it to be counted as correct.

The notation used is as follows: NE – named entity, PER–person entity, LOC–location entity, ORG–organization entity.

<table>
<thead>
<tr>
<th>#Files</th>
<th>NE</th>
<th>PER</th>
<th>LOC</th>
<th>ORG</th>
<th>#Words</th>
<th>%NEs in files</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>526</td>
<td>301</td>
<td>79</td>
<td>146</td>
<td>3689</td>
<td>14.25</td>
</tr>
<tr>
<td>2</td>
<td>483</td>
<td>149</td>
<td>198</td>
<td>136</td>
<td>9586</td>
<td>6.67</td>
</tr>
<tr>
<td>3</td>
<td>587</td>
<td>370</td>
<td>181</td>
<td>36</td>
<td>7978</td>
<td>13.48</td>
</tr>
<tr>
<td>4</td>
<td>502</td>
<td>318</td>
<td>146</td>
<td>38</td>
<td>14699</td>
<td>9.92</td>
</tr>
</tbody>
</table>

Table 4.3 Number of Entities in the Test Data

Experiment 1:

Originally the system is designed with plain text file as an input instead of sentences. This file is then passed through each of filters or components of the NERC system described in section (3.2). All the named entities that were identified from the
context rules are now stored in another lexicon but their information is not used within the same file. Results of this experiment on different datasets are given below.

<table>
<thead>
<tr>
<th>Test File</th>
<th># of NEs Identified by the System</th>
<th># of NEs Correctly Identified by the System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NE</td>
<td>PER</td>
</tr>
<tr>
<td>F1</td>
<td>48</td>
<td>38</td>
</tr>
<tr>
<td>F2</td>
<td>235</td>
<td>71</td>
</tr>
<tr>
<td>F3</td>
<td>440</td>
<td>272</td>
</tr>
<tr>
<td>F4</td>
<td>518</td>
<td>329</td>
</tr>
<tr>
<td>F5</td>
<td>129</td>
<td>97</td>
</tr>
<tr>
<td>F6</td>
<td>305</td>
<td>173</td>
</tr>
<tr>
<td>F7</td>
<td>32</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.4 NE Identification without using Generated NE Information

<table>
<thead>
<tr>
<th>File Name</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NE</td>
<td>PER</td>
<td>LOC</td>
</tr>
<tr>
<td>F1</td>
<td>87.5</td>
<td>84.21</td>
<td>100</td>
</tr>
<tr>
<td>F2</td>
<td>92.76</td>
<td>80.28</td>
<td>96.63</td>
</tr>
<tr>
<td>F3</td>
<td>92.85</td>
<td>92.40</td>
<td>91.92</td>
</tr>
<tr>
<td>F4</td>
<td>93.63</td>
<td>94.85</td>
<td>98.55</td>
</tr>
<tr>
<td>F5</td>
<td>82.17</td>
<td>79.38</td>
<td>86.95</td>
</tr>
<tr>
<td>F6</td>
<td>79.67</td>
<td>76.87</td>
<td>90</td>
</tr>
</tbody>
</table>
Table 4.5 Performance of NERC system without using generated NE information

In a number of NEs correctly identified by the system column, number of NEs correctly identified may not equal to sum of correctly identified person, location, organization names. Since a NE may be identified correctly as NE but it might be assigned wrong category.

**Experiment 2:**

Presuming, precision to be 100% the initially built system is executed sentence by sentence instead of file by file. This now is our final NERC system and the method makes use of dynamically generated named entity lexicon built from earlier sentences of the same file. Named entities that were discovered from context rules and stored in lexicon are used in another sentence. Generated named entity information is used within the file in the subsequent sentences. There is however a drawback with this method. If a non-named entity is identified as NE from contextual rules and stored as NE in the lexicon then it may reduce precision and F-measure.

<table>
<thead>
<tr>
<th>Test File</th>
<th># of NEs Identified by the System</th>
<th># of NEs Correctly Identified by the System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NE</td>
<td>PER</td>
</tr>
<tr>
<td>E1</td>
<td>60</td>
<td>49</td>
</tr>
<tr>
<td>E2</td>
<td>244</td>
<td>74</td>
</tr>
<tr>
<td>E3</td>
<td>63</td>
<td>35</td>
</tr>
<tr>
<td>E4</td>
<td>531</td>
<td>340</td>
</tr>
<tr>
<td>E5</td>
<td>441</td>
<td>273</td>
</tr>
<tr>
<td>File Name</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>File Name</td>
<td>NE</td>
<td>PER</td>
</tr>
<tr>
<td>E1</td>
<td>71.66</td>
<td>65.30</td>
</tr>
<tr>
<td>E2</td>
<td>92.21</td>
<td>78.37</td>
</tr>
<tr>
<td>E3</td>
<td>53.96</td>
<td>28.57</td>
</tr>
<tr>
<td>E4</td>
<td>91.14</td>
<td>90.0</td>
</tr>
<tr>
<td>E5</td>
<td>93.65</td>
<td>94.87</td>
</tr>
<tr>
<td>E6</td>
<td>81.20</td>
<td>78.43</td>
</tr>
<tr>
<td>E7</td>
<td>79.17</td>
<td>78.14</td>
</tr>
<tr>
<td>E8</td>
<td>79.54</td>
<td>77.77</td>
</tr>
</tbody>
</table>

Table 4.6: NE Identification using Generated NE Information

Table 4.7: Performance of NERC System using Generated NE Information
Experiment 3:

Results of this system when used without morphological analyzer are given in below table:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NE  PER  LOC ORG</td>
<td>NE  PER  LOC ORG</td>
<td>NE  PER  LOC ORG</td>
</tr>
<tr>
<td>E1</td>
<td>71.66    65.30 88.88 100</td>
<td>58.10 50 100 100</td>
<td>64.17 56.63 94.11 100</td>
</tr>
<tr>
<td>E2</td>
<td>92.01    78.08 94.30 90.47</td>
<td>83.26 76 86.56 70.37</td>
<td>87.42 77.02 90.27 79.16</td>
</tr>
<tr>
<td>E3</td>
<td>53.96    28.57 78.29 60</td>
<td>94.44 100 78.26 100</td>
<td>68.68 44.44 78.26 75</td>
</tr>
<tr>
<td>E4</td>
<td>91.14    90 90.79 76</td>
<td>87.84 85 93.67 57.57</td>
<td>89.46 87.42 92.21 65.51</td>
</tr>
<tr>
<td>E5</td>
<td>92.97    94.20 98.55 82.10</td>
<td>77.94 86.37 86.07 53.42</td>
<td>84.79 90.12 91.89 64.73</td>
</tr>
<tr>
<td>E6</td>
<td>71.42    67.61 83.33 45.45</td>
<td>71.94 66.98 80 62.5</td>
<td>71.68 67.29 81.63 52.63</td>
</tr>
<tr>
<td>E7</td>
<td>78.15    76.47 87.25 44.11</td>
<td>82.46 76.47 90.81 65.21</td>
<td>80.25 76.47 88.99 52.63</td>
</tr>
<tr>
<td>E8</td>
<td>78.78    71.42 93.33 63.63</td>
<td>47.27 20 60.86 100</td>
<td>59.09 31.25 73.68 77.77</td>
</tr>
</tbody>
</table>

Table 4.8 Performance of NERC System without using morph and using generated NE information from the above results, it can be observed that the use of morphological analyzer improves the performance of NERC system.

All the above 8 test files together contains 35, 952 words. Out of them 22, 979 words are identified by dictionary, 3886 words are analyzed by morphological analyzer, 513 English words are identified by English word filter and 1159 words are identified by external saMdhi module. The percentages of words analyzed at each filter are given below.
<table>
<thead>
<tr>
<th>Filter type</th>
<th>Dictionary</th>
<th>Morphological Analyser</th>
<th>Loan Words</th>
<th>Sandhi Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>63.91</td>
<td>10.8</td>
<td>1.42</td>
<td>3.22</td>
</tr>
</tbody>
</table>

**Table 4.9 Words Identified at each Filter**

**4.10 Summary**

It can be inferred that not much had been done earlier in NER for Telugu compared to what was done in English. Various approaches available for NER have been discussed in this particular research effort. In this thesis, a platform independent rule based named entity recognizer and classifier is built for Telugu. The system is built using newspaper corpus and Telugu wiki corpus. Various experiments have been conducted and consequent results have been elucidated as well. Since the Telugu language suffers from the lack of labeled data, this system can be used to produce training data which can further be used for machine learning techniques or hybrid techniques that could eventually produce unswerving annotated data. Though this is a rule based system, it is built upon looking at different domains of data such as history, scientific documents, novels, stories, literature, articles etc. What can be said beyond a doubt is that this system can work across all of the mentioned domains with satisfactory results. An attempt is made to identify multi-word named entity which helps in identifying the multi word organization names which is otherwise not possible with token level named entity recognition. This system may be executed on entire sentence database to get a named entity tagged data with precision in the range of 79%-94%. There are a few observations and recommendations that emerge while building a named entity recognition and classification system for Telugu.
- Common and proper noun disambiguation can be made more accurately by efficient verb (action verbs which are specific to living beings) classification. A single verb root can lead to formation of a few hundred thousand word forms. If we can identify and classify verb forms in more detail we would be able to disambiguate between common nouns and proper nouns more efficiently.

- Ability to distinguish between domain based loanwords and loanwords which we use in our daily life but do not generate NEs enhances the performance of the system significantly.

- Corrections in the Telugu newspaper and wiki corpus for proper punctuation marks and word division may also generate highly dependable named entity tagged data.