International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-11, September 2019

NLP: Rule Based Name Entity Recognition

N.Kannaiya Raja, Naol Bakala, S. Suresh,

Abstract: ***Named Entity Recognition (NER) is an information extraction task aimed at identifying and classifying words of a sentence, a paragraph or a document into predefined categories of Named Entities (NEs). NEs are terms that are used to name a person, location or organization. They are also used to refer to the value or amount of something. NER is an important tool in almost all NLP application areas out of which it is very essential in Search Engines (Semantic based), Machine Translation, and Question-Answering, Indexing for Information Retrieval and Automatic Summarization systems. This paper presents Rule-based approach for the development of Named Entity Recognition (NER) system for Afan Oromo language.***

Key Words: ***Natural Language Process, Named Entity Recognition, Rule Based Approach.***

I. INTRODUCTION

Natural Language Processing (NLP) is one of the important fields in Computer Science. Basically, it analyzes text that is based on both a set of theories and a set of technologies [1]. NLP initially started at the late 1940s when machine translation was first used to decrypt enemy codes during World War II. However, not many researches in NLP were conducted until the 1980s. There are a lot of fields that apply the NLP technologies such as Information Retrieval, Information Extraction, Question-Answering and etc. [1]. Most recent studies focus on Information Extraction (IE). There are three types of input files in IE which are structured, semi-structured or free text which is as shown as Fig. 1 [2]. Structured inputs refer to HTML pages while semi-structured inputs refer to XML pages and records. News articles are considered as unstructured input texts and they are written and understandable by human. News articles are hard to be understood by machines. A computer machine will not be able to comprehend the content of these articles. Nowadays, a huge volume of articles can be easily retrieved and extracted from websites. Hence, it would take a long time for human to manually process these articles in a short time. Besides that, the process of annotating articles manually often provides biased results. Hence, an automated process is needed and such process is known as Information Extraction.

Manuscript published on 30 September 2019.
*Correspondence Author(s)
Dr.N.Kannaiya Raja, M.E., Phd.,,Professor, Department of Computer Science ,Ambo University, Ambo, Ethiopia,. Email: kannaiyaraju123 @gmail.com.
Mr. Naol Bakala, M.Sc., Head/Department of Computer Science, Ambo University, Ethiopia.naolbakala@gmail.com.
Mr. S. Suresh, M.E., Asst. Professor, Asst Professor/ Department of Computer Science & Engineering, C. Abdul Hakeem College of Engineering & Technology, Vellore, Tamil Nadu, India, surevitcahcet@gmail.com.

IE is a process that extracts information from unstructured articles to provide more useful information. Given an article, a machine will learn on how to answer certain questions (e.g., How can we determine who is the CEO of a company? What is that name of the company?) One of the sub-tasks of IE is to help the process to identify and extract such information called named-entity and it is known as a Named Entity Recognition (NER) process. Named Entity Recognition process was a popular discussion at the Sixth Message. Understanding Conference (MUC-6) [3], [4]. The NER process helps users to produce a more meaningful corpus by identifying proper names in the corpus and classifying them into groups such as person, organization, locations and etc. For example, the query "Steve Job" should not check only on the word "Steve" or "Job". The word "Job" may lead to the process of searching for similar word such as "occupation". Hence it will lead to other meaning instead of "Steve Job," the co-founder of Apple Inc One of the early named entity studies was introduced by Lisa F. Rau [5]. Rau proposed an automated way to recognize company names from financial news. Company entities are extracted based on heuristics rules. The main target of this research paper is to information extraction of NEs in plain text is of key importance in numerous NLP applications. In information extraction systems, NEs generally carry important information about the text itself, and thus are targets for extraction. Since entity names form the main content of a document, NER is a very important step toward more intelligent information extraction and management (Zhou & Su 2002). The task has also important significance in the Internet search engines and is an important task in many of the language engineering applications such as Machine Translation, uestion-Answering systems, indexing for Information Retrieval and Automatic Summarization.

II. METHODOLOGY

A number of methods (techniques) are employed for the successful completion of this study. Some of them are discussed below. 2.1. Review of Literatures a number of related works and resources have been reviewed. This consists of conference and journal articles, white papers and NER systems developed for other languages. The large portions of reviewed materials are conference and journal articles. The nature, operational background and performance of NER systems like Stanford-ner, MENE and LingPipe's neDemo are also studied. In addition, a discussion was made with Afan Oromo Linguistic. 2.2. Corpus data collection and Development Afan Oromo does not have publicly available annotated corpus text for any NLP task including NER so far. As a result, we collected an electronic text (a total of 4014 articles) from two state owned news papers: „Bariisaa" and „Kallacha Oromiyaa". The collected articles were further edited manually by removing those sentences which does not have NEs at all.

The resulting data is then tokenized and tagged manually in accordance with the CoNNL 2002 standard. Finally, NE corpus of size more than 23,000 tokens out of which 3,575 are NEs was developed. Since we used part-of-speech tagger (POST) in our system, for feature extraction, we did additional work of developing a corpus for training the POST. Accordingly, we developed an additional corpus of size 4588 words with each word tagged with their part-of-speech.

2.3. Prototype Development A prototype was developed in order to ensure whether our study works in accord with the ideas and theories of NER.

2.4. Conducting research after the formal system was being developed it was tested and evaluated with different parameters by performing an objective testing. The performances obtained throughout the conducted experiments were given in three evaluation metrics: Precision, Recall and F-measure.

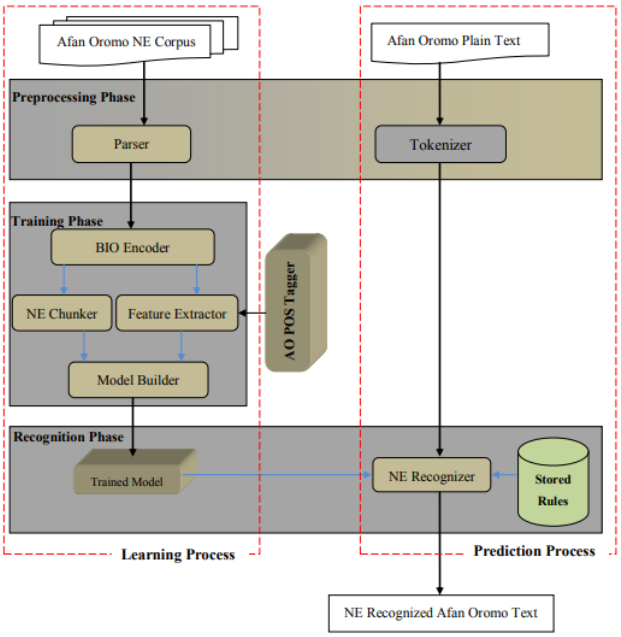
III. RULE BASED/LINGUISTIC APPROACH

Ralph Grishman in 1995 developed rule-based NER systems. The Rule based approach is a classical approach to NER. It uses rules manually written by linguists. Rule based systems parse the source text and produce an intermediate representation which may be a parse tree or some abstract representation. To use this, a large gazetteer list has to be built for different Named Entity classes.

Hand-made Rule-based NER systems extract names entities using lots of human made rule sets. Generally, the systems consist of a set of patterns using grammatical (e.g. part of speech), syntactic (e.g. word precedence) and orthographic features (e.g. capitalization) in combination with dictionaries (Mansouri et al. 2008). An example for this type of system is: “Atileeti Daraartuu Tulluun Boqqojjiitti dhalatte.” (Athlete Derartu Tulu was born in Bokoji.) In this example a proper noun (Daraartuu Tulluu) follows a person's title (Atileeti), and then the proper noun that is started with capital letter and ends with suffix (-tti) (Boqqojjii) is a location's name. These approaches are relying on manually coded rules.

These kinds of models have better results for restricted domains, are capable of detecting complex entities that learning models have difficulty with. However, the rule-based NE systems lack the ability of portability and robustness, and furthermore the high cost of the maintenance of the rule increases even when the data is slightly changed. These type of approaches are often domain and language specific and do not necessarily adapt well to new domains and languages.

IV. ARCHITECTURE OF A TYPICAL NER SYSTEM PHASES OF NER



In NER architecture, a phase is just a module that holds related components together. Components are related to each other based on the task they perform, their respective input and their general contribution in a process.

4.1.1. Pre-processing Phase

The pre-processing phase is the phase where necessary pre-processing tasks are performed on the input data, the training corpus and the plain text, so that it can effectively be utilized by the rest of the phases. Parser and Tokenize are the components placed in the pre-processing phase.

4.1.2. Parser

During the development of the corpus, we first went through all the obtained news articles and selected those sentences that have at least one NE. The obtained sentences are then tokenized and each token was tagged with its NE tag in accordance with the CoNLL‟s 2002 standard. NE tagging during this time was done manually. This makes the task prone to errors. Some of the errors that might arise are errors like failing to be consistent with the rules of CoNLL and mistyping. If the data is victim of these errors, it will mislead the other tasks that depend on the data or even it will halt the progress into the next component. Parser is the essential component that checks the NE tagged training corpus and report if there are errors. It mainly checks the NE tagged data against the CoNLL‟s 2002 rules and standards. We used CoNLL‟s BIO scheme for tagging the training data. CoNLL has (Erik 2002) developed a legal sequence of tags which every NE corpus should abide by. The Parser will check the data whether it is legally tagged or not. For example, for the following two tags created for the sentence

Retrieval Number: K20470981119/19©BEIESP
DOI: 10.35940/ijitee.K2047.0981119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

"Yunivarsitiin Amboo magaalaa Ambootti argama.", the tagging on the left is legal while the tagging on the right violates the rule of CoNLL 2002 standard. Because, it has to be preceded by B-LOC tag or it has to be B-LOC. Once the training data is checked by the parser and everything is found to be correct, the data will be made ready for generating the tokens and the tags, token/tag sequence, which is done by the BIO Encoder.

Yunivarsitiin B-ORG	Yunivarsitiin B-ORG
Amboo I-ORG	Amboo I-ORG
Magaalaa O	magaalaa O
Ambootti B-LOC	Ambootti I-LOC
Argama O	argama O
. O	. O

4.1.3. Tokenizer

Tokenization can basically be defined as the process of breaking up a text into its constituent elements, tokens. Tokenization can occur at a number of different levels: a text could be broken up into paragraphs, sentences or words. For any given level of tokenization, there are many different algorithms for breaking up the text depending on the intention of the system and the language under consideration. For example, at the word level, it is not immediately clear how to treat such strings as "can't," "$22.50," "New York," and "so-called". Having these confusing facts beside, mostly in languages like English, tokenization at the word level is done based on white space.

In Afan Oromo, tokenization is a trivial problem as its writing fashion is identical to English since words are separated by a white space. In fact, there are circumstances in which two words are treated as a single word when they are joined to each other by a hyphen (-) and they are commonly called „jecha tishoo". For example „gara-jabeessa" and „bu'a-qabeessa" are treated as a single word. As shown in the architecture of the system, tokenization is done on the input plain text during the prediction process. Tokenization on the training corpus is done during corpus development. Tokenization for the input text is performed by a component called tokenizer. The tokenizer takes the input text supplied from a user and tokenizes it into a sequence of tokens that can make it easy to recognize NEs. Token is explained in detail in the implementation part of our system . The tokenizer will treat double words that are connected to each other by a hyphen (-) as separate words and the hyphen mark is treated as a token. Though a hyphen does not have a direct effect on NEs, treating it as a token will enable us group all punctuations under a single category. For example: the word „gara-jabeessa' has three tokens. The tokenizer treats words which contain apostrophe („) as a single word as in „Ya'aaballoo". „Obbo" and the case of the word „Barkeessaan" and its suffix „n" can be used as features to identify „Barkeessaan" as a person name. The feature extractor is designed to extract features from the training data and store (cache) them for future use. The features are extracted from an input sequence of tokens those supplied by the encoder"s tokenizer and the tags those supplied by the encoder.

4.1.4. NE Chunker

The token/tag sequence supplied by the BIO encoder has to be chunked before building the model. This means, those sequences of tokens that follow each other and belong to similar category has to be considered as a phrase and assigned a single tag specifying their category. NE Chunker is designed to perform this task by detecting phrase boundaries. It creates a chunk set from a sequence of tagged tokens. This process is called Chunking. Chunking combines two or more continuous NEs that have identical tag (with the first B and the successive Is) and tag them with a single NE tag. Chunking is helpful for those NEs that comprise two or more words. For example, let us consider a person name „Abbaaduulaa Gammadaa" which is composed of two words. Following the BIO tagging scheme „Abbaaduulaa" will be tagged as B-PER and „Gammadaa" is tagged as I-PER. When this is provided to the chunker, it understands that both words follow each other and also belong to the same NE type, PERSON. Based on this fact, it combines them together and creates a single chunk: „Abbaaduulaa Gammadaa' with a tag PERSON. The sequence of chunks will be handed to the model builder in parallel with extracted features.

4.1.5.

5.1.5. Model Builder

Our main concern with the machine learning component is to build a trained model. All the previously discussed components in the training phase perform a task of generating required information that is used for the actual training procedure. The procedure of training the model is done by the model builder. Model builder is the component designed to estimate the model coefficients and then build a trained model. This means it trains a CRF model. Training in this case consists of estimating the model coefficient based on the input from the training corpus that is supplied in the form chunking by the NE Chunker, and the extracted features that are supplied by the Feature Extractor.

4.2. Recognition Phase

The recognition phase is the phase where the actual work of recognizing NEs from the tokenized plain text is done. Components included in this phase are the Trained Model, NE Recognizer and Stored Rules.

4.2.1. Trained Model

The trained model is what we look for as part of the machine learning component and it is the final output of the learning process. Since we used CRF algorithm in the training process and the model is trained with Afan Oromo data, we can call the trained model as Afan Oromo CRF Model. The model contains a set of values that are obtained from the calculations performed during the estimation. It corresponds to equation 5 of section the trained model, here, is a set of parameters (known as weights) which corresponds to the importance of the features used in the task. It is an estimation of all the parameters that are obtained through training. The trained model is the very essential component that assists the NE Recognizer to recognize NEs from the tokenized plain texts. While doing so, it performs the task of Inference. Inference is the procedure of finding the most likely sequence of tags for the tokenized input texts as described in section 6.1.2.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The model is the main component that plays a crucial role in supplying necessary information during NE recognition.

4.2.2. Stored Rules

Naturally, machine learning components require a huge amount of training data to perform well. Still with huge amount of training data one cannot be sure whether the data contains all the structure and organization associated to NEs. As a result, it can be possible to develop a reasonable amount of training data and then develop a set of rules that can work in collaboration with the trained model. The proposed architecture is designed based on this assumption. The stored rules are designed to focus on the general structure and organization of Afan Oromo NEs, so that they can cover those parts that are not covered by the model. The rules will be developed based on the idea of pattern matching. We have identified some of the words that are used to develop the pattern of NEs in Afan Oromo. They are found in appendix A. The rules are expected to assist the model in such ambiguities as when it gets equal probability for a token to classify it into different NE categories. By doing so, the rules will perform the task of disambiguation. The rules will act in the detection procedure of the recognition process.

4.2.3. NE Recognizer

NE Recognizer is the core component that takes a pre-processed input plain text from a user, recognizes NEs out of it and supplies NE recognized Afan Oromo text as an output. As depicted on the architecture, it is assisted by both the trained model and the stored rules. NER"s NE recognizer performs two main tasks: detection and classification. Detection is the first task which finds candidate tokens that can be NEs from the tokenized plain text. Based on the combined knowledge supplied from both the model and the rules, detection is performed as follows. The model, invoked by the recognizer, does a number of tasks turn by turn. The features that are extracted and stored during the training phase are supplied to the recognizer to identify NEs from the text.

The model then selects candidate tokens based on the calculated probability. To check if there are wrongly detected tokens and there are found ambiguities by the model, the stored rules are used. This concludes the detection process. The classification portion is also done by a joint work from the model and the rules. The recognizer starts the classification process by tagging the detected candidate tokens. Each token will be tagged with their possible NE tags using the BIO scheme. This is done based on the knowledge obtained from the model. The stored rules will react to check if the tokens are tagged correctly. If there are wrongly tagged tokens, it corrects. The recognizer then chunks those sequence of tokens with identical tag extension and make them ready for final output. This completes the classification part and the prediction process. Finally, the recognizer will generate NE recognized Afan Oromo text and send it as an output.

V. IMPLEMENTATION OF NER

jupyter Name entity-Copy1 Last Checkpoint: 27 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [1]:
```
# coding: utf-8
```

In [2]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
```

In [14]:
```
from bs4 import BeautifulSoup
import requests
import re
def url_to_string(url):
    res = requests.get(url)
    html = res.text
    soup = BeautifulSoup(html, 'html5lib')
    for script in soup(["script", "style", 'aside']):
        script.extract()
    return " ".join(re.split(r'[\n\t]+', soup.get_text()))

ny_bb = url_to_string("https://www.bbc.com/afaanoromoo")

article = nlp(ny_bb)
len(article.ents)
```

Out[14]: 140

In [23]:
```
def score(self, y_true, y_pred):
        """Calculate f1 score.
        Args:
            y_true (list): true sequences.
            y_pred (list): predicted sequences.
        Returns:
            score: f1 score.
        """
        score = f1_score(y_true, y_pred)
        print(' - f1: {:04.2f}'.format(score * 100))
        print(classification_report(y_true, y_pred, digits=4))
        return score
```

In [24]:
```
 def on_epoch_end_fit(self, epoch, logs={}):
        X = self.validation_data[0]
        y = self.validation_data[1]
        y_true, y_pred = self.predict(X, y)
        score = self.score(y_true, y_pred)
        logs['f1'] = score
```

In [25]:
```
def on_epoch_end_fit_generator(self, epoch, logs={}):
        y_true = []
        y_pred = []
        for X, y in self.validation_data:
            y_true_batch, y_pred_batch = self.predict(X, y)
            y_true.extend(y_true_batch)
            y_pred.extend(y_pred_batch)
        score = self.score(y_true, y_pred)
        logs['f1'] = score
```

In [16]:
```
labels = [x.label_ for x in article.ents]
Counter(labels)
```

Out[16]:
```
Counter({'CARDINAL': 12,
         'DATE': 2,
         'EVENT': 2,
         'FAC': 2,
         'GPE': 4,
         'LOC': 2,
         'NORP': 8,
         'ORDINAL': 1,
         'ORG': 40,
         'PERSON': 57,
         'PRODUCT': 4,
         'TIME': 4,
         'WORK_OF_ART': 2})
```

In [17]:
```
items = [x.text for x in article.ents]
Counter(items).most_common(3)
```

Out[17]:
```
[('Afaan Oromoo', 5), ('Viidiyoo', 4), ('Keeniyaatti', 4)]
```

In [18]:
```
sentences = [x for x in article.sents]
print(sentences[20])
```

```
imale hin milkaa'in hafe
```

In [23]:
```
def score(self, y_true, y_pred):
        """Calculate f1 score.
        Args:
            y_true (list): true sequences.
            y_pred (list): predicted sequences.
        Returns:
            score: f1 score.
        """
        score = f1_score(y_true, y_pred)
        print(' - f1: {:04.2f}'.format(score * 100))
        print(classification_report(y_true, y_pred, digits=4))
        return score
```

In [24]:
```
 def on_epoch_end_fit(self, epoch, logs={}):
        X = self.validation_data[0]
        y = self.validation_data[1]
        y_true, y_pred = self.predict(X, y)
        score = self.score(y_true, y_pred)
        logs['f1'] = score
```

In [25]:
```
def on_epoch_end_fit_generator(self, epoch, logs={}):
        y_true = []
        y_pred = []
        for X, y in self.validation_data:
            y_true_batch, y_pred_batch = self.predict(X, y)
            y_true.extend(y_true_batch)
            y_pred.extend(y_pred_batch)
        score = self.score(y_true, y_pred)
        logs['f1'] = score
```

In [22]:
```
class F1Metrics(Callback):

    def __init__(self, id2label, pad_value=0, validation_data=None):
        """
        Args:
            id2label (dict): id to label mapping.
            (e.g. {1: 'B-LOC', 2: 'I-LOC'})
            pad_value (int): padding value.
        """
        super(F1Metrics, self).__init__()
        self.id2label = id2label
        self.pad_value = pad_value
        self.validation_data = validation_data
        self.is_fit = validation_data is None
```

```
In [19]: displacy.render(nlp(str(sentences[20])), jupyter=True, style='ent')

c:\python35\lib\runpy.py:170: UserWarning: [W006] No entities to visualize found in Doc object. If this is surprising to you, m
ake sure the Doc was processed using a model that supports named entity recognition, and check the `doc.ents` property manually
if necessary.
  "__main__", mod_spec)

imale hin milkaa'in hafe

In [20]: print([(x, x.ent_iob_, x.ent_type_) for x in sentences[20]])

[(imale, 'O', ''), (hin, 'O', ''), (milkaa'in, 'O', ''), (hafe, 'O', ''), (
, 'O', '')]

In [21]: import numpy as np
         from keras.callbacks import Callback
         from seqeval.metrics import f1_score, classification_report

Out[16]:: Polyglot NER Tool Metrics
          The accuracy is 0.8333333333333334
          The recall is 0.8333333333333334
          The precision is 0.8333333333333334
```

VI. CONCLUSIONS

This research proposed and designed a system called NER to solve Afan Oromo NE problem. The system is designed based on rule based. The rule based component is not implemented due to time constraints and we left it as a future work. We implemented the machine learning component using python NERs algorithm. For training the system, we developed Afan Oromo NE corpus for BBC by https://www.bbc.com/afaanoromoo. We believe this will be the largest Afan Oromo corpus for NLP for extraction of information. The input plain text entered by a user is first tokenized by the tokenizer and supplied to the NE Recognizer. The NE Recognizer recognizes the possible NE tokens from the tokenized text with the help of the trained model. The recognition was implemented using forward backward algorithm. This work is the first of its type for Afan Oromo language. The systems evaluation showed a promising performance. Being tested on a test data of 5,000 tokens, it obtained 85% Recall, 85% Precision and 85% F1-measure.

REFERENCES

1. E. D. Liddy, “Natural language processing,” in Encyclopedia of Library and Information Science, 2nd Ed. NY, Marcel Decker, Inc, 2001.
2. C. H. Chang, M. Kayed, M. R. Girgis, and K. Shaalan, “A survey of web information extraction systems,” IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 10, pp.1411-1428, Oct. 2006.
3. G. Ralph and S. Beth, “Message Understanding Conference-6: A Brief History,” in Proc.
4. G. Ralph, “The NYU system for MUC-6 or where’s the syntax?,” in Proc. Sixth Message Understanding Conference, MUC-6, 1995, pp. 167-175.
5. L. F. Rau, “Extracting Company Names from Text,” in Proc. Conference on Artificial Intelligence Applications of IEEE, 1991. [6] ABGene. [Online]. Available: ftp://ftp.ncbi. nlm.nih.gov/pub/tanabe/AbGene/

AUTHORS PROFILE

Dr.N.Kannaiya Raja, M.E., Phd., working as Professor in Department of Computer Science, Ambo University, Ambo, Ethiopia. His Major research interest is Natural Language Processing, Pattern Matching and data science.

Mr.Naol Bakala, M.Sc is working as Head in Computer Science Department in Ambo University, Ethiopia. His major research interest lies in Natural Language Processing

S. Suresh, M.E., currently working as Asst. Professor in C.Abdul Hakeem College of Engineering and Technology, Vellore, Tamilnadu, India. His research interest is Cognitive Radio Networks, Data Mining and Natural Language Processing.