Pak. J. Engg. & Appl. Sci. Vol. 21 July, 2017 (*p.* 15–22)

Urdu Named Entity Recognition System using Hidden Markov Model

Muhammad Kamran Malik, Syed Mansoor Sarwar

1. *Punjab College of Information Technology, University of the Punjab Lahore, Pakistan*

* **Corresponding Author:** E-mail: kamran.malik@pucit.edu.pk

Abstract

Named Entity Recognition (NER) is the process of identifying names of Persons, Organizations, Locations and other miscellaneous information like number, date, and measure in a given text. In this paper, we describe the development of a NER system for Urdu Language using Hidden Markov Model (HMM). First, we show a comparison of IOB2 and IOE2 tagging schemes. Second, we show the preprocessing of Urdu before feeding data to the HMM model for training using the IOE2 tagging scheme. Finally, we use the Part of Speech (POS) information, gazetteers, and rules to improve the accuracy of the system. Our system yields 66.71%, 71.70%, and 69.12% as the values for precision, recall, and f-measure, respectively. This system will help us improve the results of Urdu Information Retrieval, Machine Translation, and Questing and Answering systems.

Key Words: Statistical NER, Indian language NER, Resource poor language, Language independent NER, Urdu NER, Urdu HMM

1. Introduction

Urdu is the national language of Pakistan and written from right to left using the Arabic script, and 69.1 million people worldwide speak Urdu [19]. The task of a Named Entity Recognition and Classification (NERC) system is to identify proper names from the given text and then classify them into person, organization, and location names. Sometimes the process of identifying time, date, money, and percent expression can also be considered as part of the NERC system. A NERC system has many applications, including intelligent Information Retrieval (IR), Machine Translation (MT), and Questioning and Answering (Q&A) systems.

We face the following challenges in Urdu for the development of Urdu NER system [3]:

1. There is no concept of capitalization in Urdu, which is a major clue for NEs in English.
2. Urdu is free word-order language.
3. Urdu is Agglutinative in nature.
4. In Urdu, sometimes words are written with diacritic and sometimes without diacritic, which causes multiple variations of a single word.
5. Urdu contains words of different languages, including Arabic, Persian, Sanskrit, and English.
6. In Urdu, very less work has been done from computational perspective.
7. In Urdu, there is an issue of word segmentation.
8. There is also the problem of lack of standardization and spelling variations in Urdu.
9. In Urdu, many words, depending on their context, can be considered as common nouns as well as proper nouns (i.e., candidate for NE). For example, Shan, Kamran, Fazal, Kiran, Aftab, Manzoor, etc can be NEs, i.e., Person can be considered as common noun. The context may help in identifying proper nouns against common nouns but due to no concept of capitalization in Urdu, disambiguation becomes harder than English.
10. There is serious lack of labeled data in Urdu for machine learning.
11. There is a huge variation in writing numbers in Urdu.

In our Urdu NERC system, we first perform experiments using two different types of tagging schemes, i.e., Inside-Outside-Begin (IOB2) and Inside-Outside-End (IOE2). The main purpose of using these two tagging schemes for experimentation is that IOB2 is considered suitable for prepositional languages like English and IOE2 is considered suitable for postpositional languages like Japanese. Urdu is also a postpositional language; that's why we compare the results of the two tagging schemes and results show that IOE2 produces better results than IOB2. We use character level, word level normalization, Part of Speech, and Regular expressions to improve accuracies.

2. Literature Review

The NERC system has been developed for different languages including Chinese, Dutch, English, French, German, Hindi, and Italian using supervised learning algorithms, semi-supervised learning algorithms, unsupervised algorithms, and hand crafted rules as discussed in [5]. Little work has been done on Urdu NER compared to other languages. Focused work on Urdu NER started after International Joint Conference on Natural Language Processing (IJCNLP)-08 [20] where five Indian languages were targeted for study: Bengali, Hindi, Oriya, Telugu, and Urdu.

[10] describes development of NER systems for the Urdu, Hindi, Bengali, Telugu, and Oriya languages using IJCNLP workshop NER data. Language specific rules and Maximum Entropy (ME) approach along with gazetteers are used to develop NER systems for these languages. Experiment was conducted on 12 types of NEs and the overall accuracy for the Hindi, Bengali, Oriya, Telugu, and Urdu NER systems in terms of f-measure were 65.13%, 65.96%, 44.65%, 18.74%, and 35.47% respectively.

[15] used Beaker-Riaz corpus for the development of a rule based NER system for Urdu. Hand crafted rules were developed due to the availability of a limited amount of annotated corpus for training and testing. They used 200 documents for the construction of rules for the identification of NEs like Person, Designation, Location, Date, Number, and Organization. These rules were tested on 2262 documents that contained 206 unique NEs. By using this approach, 187 NEs were extracted out of which 171 were true NEs with 90.7% recall, 91.5% precision and 91.1% f-measure. The same approach was also used in the IJCNLP 2008 NER workshop to achieve 72.5% f-measure without tuning and 81.6% f-measure with tuning.

[16] also used a rule for the extraction of numbers, non-numeral numbers, date, and time. For the identification of Person, Location, and terms, it used suffix matching along with gazetteer. Two datasets were used with 12032 and 150243 tokens. 12 NEs were used and accuracy in terms of f-measure for dataset 1 and dataset 2 were 60.09% and 88.1%, respectively.

[6] proposes a bootstrapped model for Urdu. This model has four levels of text processing. Conditional Random Field (CRF) is used for POS tagging and NE tagging. It uses three NEs for experimentation, i.e., Person, Organization, and Location. The f-measure of the two-stage model and four-stage model were 55.3% and 68.9%, respectively. The paper also describes these two-stage and four-stage models.

[9] used the n-gram models, i.e., Unigram and Bigram models, with different smoothing techniques for the development of an Urdu NERC system. Five NEs were used for experimentation, i.e., Person, Organization, Location, Date, and Time. The highest accuracy was achieved using the bigram model along with Backoff smoothing technique with 66.2% precision, 88.18% recall, and 75.83% f-measure.

[13] highlights the challenges in the development of an Urdu NER system. Urdu and other South Asian languages are discussed in detail in this regard.

Several experiments have been conducted on the IJCNLP workshop Urdu NE data. [7] used the ME approach to build an Urdu NERC system with 37.58% precision, 33.58% recall, and 25.47% f-measure. [12] applied CRF for the development of an Urdu NERC system with 48.96% precision, 39.07% recall, and 43.46% f-measure. [8] used HMM and rules to build an Urdu NERC system with 56.21% precision, 37.15% recall, and 44.73% f-measure. [14] used CRF for the training and testing of an Urdu NERC system. The system yields 54.45% precision, 26.36% recall, and 35.52% f-measure. [17] used CRF on an Urdu language on only three NEs, i.e., person, organization, and location. The reported results are 64.11% precision, 66.98% recall, and 65.51% f-measure.

3. Tagging Problem

NER can be considered as a sequence labeling problems where we want to determine a vector $\boldsymbol{Z} = \{z_0, z_1, \ldots z_T\}$ of random variables given an observed vector $\boldsymbol{X} = \{x_0, x_1, \ldots x_T\}$. Each variable $z_s \in \boldsymbol{Z}$ can be NE of the word at position s, and $x_s \in \boldsymbol{X}$ is the word at positions.

Now we define the tagging problem for finding the most probable NE sequence $z_{1:n}$ for the word sequence $x_{1:n}$. More formally,

$$argmax_{z_{1:n} \in z} \; P(z_{1:n} \mid x_{1:n}) \tag{1}$$

3.1 Hidden Markov Model (HMM)

In HMM [4] we have two set of states and a triple (π, A, B). First is a set of observable states that is the input sentence or word sequence $X = \{x_{1:n}\}$ such that $X \in X_s$ with x_i be the i^{th} word in *X*. Second is the set of hidden states that is represented by NE $z_{1:n}$ for the word sequence $x_{1:n}$ with z_i be the i[th] NE in the sequence. Each NE represents one of the hidden states in HMM. The observable states (the word sequence) are probabilistically related (emission probabilities) to the hidden states (NE sequence) such that the sum of probabilities of all links outgoing from a single hidden state to all observable states is 1. In triple (π, A, B), we define π as the initialization vector containing the initial probabilities of all NEs z_i starting an NE sequence. We define A as a matrix of probabilities (transition or Prior Probabilities) when the underlying Markov Process transitions from one state (NE) to another. We define B as a matrix of probabilities (emission or Likelihood probabilities) of generating the word sequence $x_{1:n}$ from the underlying NE sequence $z_{1:n}$, i.e., the probability of generating or emitting a word x_i once the underlying Markov Process enters a state x_i. The triple (π, A, B) is learnt from our Urdu NE training data.

HMM defines the joint probability distribution over a word sequence paired with an NE sequence as

$$P(x_{1:n}, z_{1:n}) \tag{2}$$

The output of HMM is a tag sequence that maximizes this joint probability distribution

$$argmax_{z_{1:n} \in z} \; P(x_{1:n}, z_{1:n}) \tag{3}$$

To model this joint probability we consider our basic NE problem from Equation (1) as

$$argmax_{z_{1:n} \in z} \; P(z_{1:n} | x_{1:n}) \tag{4}$$

Bayes Rule of probability dictates us that we can calculate the probability of $(z_{1:n} | x_{1:n})$ if we know the probability of $(x_{1:n} | z_{1:n})$ and it says

$$P(z_{1:n} | x_{1:n}) = \frac{P(z_{1:n}) P(x_{1:n} | z_{1:n})}{P(x_{1:n})} \tag{5}$$

By applying Bayes Rule to Equation 3 we get

$$argmax_{z_{1:n} \in z} \; \frac{P(z_{1:n}) P(x_{1:n} | z_{1:n})}{P(x_{1:n})} \tag{6}$$

We drop the denominator for being a constant for all NEs and hence Equation 6 becomes

$$argmax_{z_{1:n} \in z} \; P(z_{1:n}) P(x_{1:n} | z_{1:n}) \tag{7}$$

This means that for each NE sequence we need to calculate the product of likelihood $P(x_{1:n} | z_{1:n})$ and prior probability $P(z_{1:n})$. We make two simplifying assumptions to estimate the probability of the NE sequence. First assumption says that the probability of a word is dependent only on its own underlying NE.

$$P(x_{1:n} | z_{1:n}) \approx \prod_{i=1}^{n} P(x_i | z_i) \tag{8}$$

Since we have used both the Bigram and Trigram HMM to formulate our results, therefore, for Bigram HMM we assume that the probability of NE is dependent only on the previous NE (First Order Markov Assumption). Thus, $P(z_{1:n})$ is expressed as shown below.

$$P(z_{1:n}) \approx \prod_{i=1}^{n} P(z_i | z_{i-1}) \tag{9}$$

For Trigram HMM we assume that the probability of NE is dependent only on previous two NEs (Second Order Markov Assumption). Thus, Equation (9) may be expressed as given below.

$$P(z_{1:n}) \approx \prod_{i=1}^{n} P(z_i | z_{i-2}, z_{i-1}) \tag{10}$$

After these two assumptions, we can rewrite Equation (2) as

$$P(x_{1:n}, z_{1:n}) \approx \prod_{i=1}^{n} P(z_i | z_{i-1}) \prod_{i=1}^{n} P(x_i | z_i) \tag{11}$$

Where $P(z | z_{i-2})$ and $P(z_i | z_{i-2}, z_{i-1})$ are called the Bigram and Trigram parameters, respectively, and $P(x_i | z_i)$ is called the emission parameter of HMM. Some of the details are also mentioned in [3].

4. Hybrid Approach For Urdu NER System

For our experiments, we have used training and testing data from IJCNLP workshop. The training data consisted of 2584 NEs, 35447 tokens, and 1508 sentences, and the testing data consisted of 1027 NEs, 12805 tokens, and 498 sentences. Details of the training and testing data are given in Table 1.

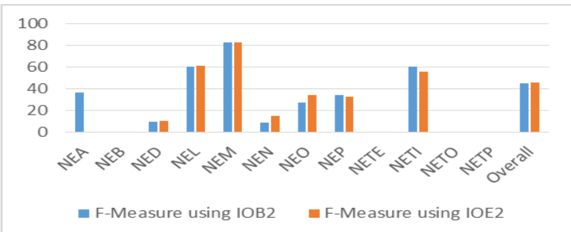

Figure 1: Comparison of IOB2 and IOE2 tagging scheme

We used two tagging schemes, i.e., IOB2 and IOE2, on 12 NEs for experimentation. First, we converted the Urdu NE data from the SSF format to the IOB2 and IOE2 formats. We used HMM for training and testing of Urdu NER. The precision, recall, and f-measure values using IOB2 are 38.38%, 54.04%, and 44.89%, respectively. Using IOB2 no entry for NEB, NETE, NETO, and NETP is found, whereas the highest f-measure value of NEM is found. Similarly, the precision, recall, and f-measure using IOE2 are 39.00%, 54.65%, and 45.52%, respectively. No word is correctly classified against NEA, NEB, NETE, NETO, and NETP. Detailed results of IOB2, IOE2, and comparison of each NE against IOB2 and IOE2 is shown in Table 2, Table 3, and Figure 1, respectively.

Table 1: Details of Urdu Corpus

Named Entity	Training Data (tokens)	Testing Data (tokens)
NEP (Person)	365	145
NED (Designation)	98	41
NEO (Organization)	155	40
NEA (Abbreviation)	39	3
NEB (Brand)	9	18
NETP (Title-Person)	36	15
NETO (Title-Object)	4	147
NEL (Location)	1118	468
NETI (Time)	279	59
NEN (Number)	310	47
NEM (Measure)	140	40
NETE (Terms)	30	4
Total NEs	2584	1027
Total Words	35447	12805
Total Sentences	1508	498

We used IOE2 for further experimentation. We normalized the whole input and then trained it using HMM. We performed two types of normalization:

1. Character level normalization
2. Word level normalization

In the character level normalization we convert different equivalent forms of a character to a standard form of character. In Urdu some of the characters that are visually the same can be written in different ways. For example آ can be written using 0622 Unicode or it can be written using two Unicode (0627+0653) ا+ٓ. We need to standardize such characters for correct matching and for better learning the parameters of the supervised learning algorithm. If we do not standardize such characters then there is the possibility that the same location name, organization name, or person name may be assigned different NEs. For example, آگرہ (Agrah), آگرہ (Agrah) may be assigned different NEs. For the purpose of Normalization, we use the Center for Language Engineering (CLE) [20] utility for Urdu character level normalization using the NFC implementation.

Table 2: Results of using IOB2 tagging scheme

NE	Precision	Recall	F-Measure
NEA	50	28.57	36.36
NEB	0	0	0
NED	11.76	8	9.5
NEL	54.1	67.57	60.01
NEM	85.26	80.20	82.65
NEN	8.57	8.11	8.33
NEO	32.01	23.78	27.31
NEP	26.07	49.55	34.16
NETE	0	0	0
NETI	79.71	48.67	60.44
NETO	0	0	0
NETP	0	0	0
Overall	38.38	54.04	44.89

Table 3: Results of Using IOE2 tagging scheme

NE	Precision	Recall	F-Measure
NEA	0.00	0.00	0.00
NEB	0.00	0.00	0.00
NED	11.76	9.09	10.26
NEL	54.77	68.84	61.01
NEM	87.37	78.30	82.59
NEN	14.29	14.71	14.49
NEO	42.45	28.13	33.83
NEP	24.64	48.15	32.60
NETE	0.00	0.00	0.00
NETI	72.46	45.05	55.56
NETO	0.00	0.00	0.00
NETP	0.00	0.00	0.00
Total	39.00	54.65	45.52

In Urdu users may write a single word with different sequence of characters like Lahore can be written as لاھور or لاہور. This is an example of word normalization where we need to standardize such words into a single form for correct learning of HMM. We perform word level normalization by using four steps, as described below:

4.1 Assignment of Part of Speech (POS) Tags

The POS information for each word is useful to improve the results of a NER system. For this purpose, we assign the POS tags to the training and testing data by using an online POS tool available at http://cle.org.pk/index.htm, which uses the POS tagset of [11]. The output of this tool is not up to the mark with respect to identifying Proper Noun (NNP) because the location name Lahore can be written using two ways لاھور and لاہور, as stated above, but this tool assigns the NN tag to the first word and NNP tag to second word.

4.2 Transliteration of Each Word

The transliteration task is carried out using the tool mentioned in [18]. This tool converts Urdu in Arabic script into Roman Urdu. For example, Roman Urdu of لاھور and لاہور is Lahore and, similarly, Roman Urdu of شملہ and شملا are Shimlah and Shimla, respectively. We take all Nouns and Proper Nouns as input and generate their transliteration.

4.3 SOUNDEX Code Generation

This module takes Roman Urdu form of all Nouns and Proper Nouns as input and generates their SOUNDEX codes. For example, the SOUNDEX code generated for Shimla and Shamila is S540.

4.4 Conversion into a Standard Word

In this module words with different spelling variations are converted into one form on the basis of highest frequency.

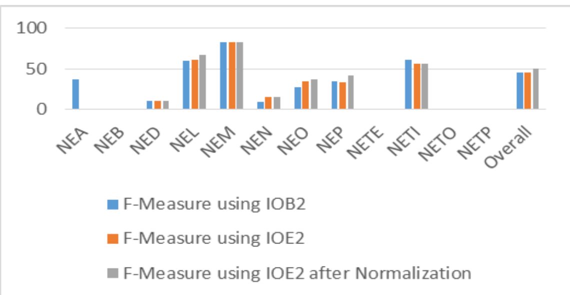

Figure 2: Comparison of IOB2, IOE2 and IOE2 after normalization

After performing character level and word level normalizations, the results improved significantly with precision, recall, and f-measure as 43.91%, 57.94%, and 49.96%, respectively. However, our model is unable to recognize NEA, NEB, NETE, NETO and NETP. Details of the results are given in Table 4 and a comparison with other two approaches is show in Figure 2.

Table 4: Results of Using IOE2 tagging scheme after normalization

NE	Precision	Recall	F-Measure
NEA	0.00	0.00	0.00
NEB	0.00	0.00	0.00
NED	11.76	9.09	10.26
NEL	63.00	72.28	67.32
NEM	87.37	79.05	83.00
NEN	14.29	14.71	14.49
NEO	45.28	30.00	36.09
NEP	32.70	55.20	41.07
NETE	0.00	0.00	0.00
NETI	74.63	45.05	56.18
NETO	0.00	0.00	0.00
NETP	0.00	0.00	0.00
Total	43.91	57.94	49.96

The last experiment is performed using some rules, i.e., regular expressions of NEN, NETI, NEM and NEA. Also, a small Gazetteer of person name, location name, organization name, Terms, Title Persons, and Title object is prepared using the Internet. The partial details of these rules are given below.

For NEN (number) we define regular expressions like Urdu_Digit = ایک (aik) (one) | دو (do) (two)| تین (theen) (three) | چار (char) (four)| پانچ (panch) (five) | چھے (chey) (six) | سات (saat) (seven) | آٹھ (aath) (eight) | نو (nao) (nine)| دس (das) (ten)| گیارہ (giarah) (eleven) | بارہ (barah) (twelve) | تیرہ (terah) (thirteen) | چودہ (chodah) (forteen) | پندرہ (pandrah) (fifteen) | سولہ (solah) (sixteen) | سترہ (satrah) (seventeen) | اٹھارہ (atharah) (eighteen) | انیس (unees) (nineteen) | بیس (bees) (twenty) | اکیس (akees) (twenty one) | باایس (baees) (twenty two) | تےایس (tayees) (twenty three) | چوبیس (chobees) (twenty four) | پچیس (bachees) (twenty five) | چھبیس (chabees) (twenty six) | ستاایس (sataees) (twenty seven) | اٹاایس (athaees) (twenty eight) | انتایس (unatees) (twenty nine) | تیس (tees) (thirty) | اکتیس (aikatees) (thirty one) | بتیس (bates) (thirty two) | تینتیس| (tantees) (thirty three) | چونتیس (chontees) (thirty four) | پینتیس (pantees) (thirty five) | چھتیس (chatees) (thirty six) | سنتیس (santees) (thirty seven) | اٹھتیس (athatees) (thirty eight) |

انتالیس (untalees) (thirty nine) | چالیس (chalees) (forty) | اکتالیس (aktalees) (forty one) | بیالیس (bialees) (forty two) | تینتالیس (tartalees) (forty three) | چوالیس (chowalees) (forty four) | پینتالیس (pantalees) (forty five) | چھیالیس (chialees) (forty six) | سینتالیس (santalees) (forty seven) | اٹھتالیس (artalees) (forty eight) | انچاس (unchas) (forty nine) | پچاس (pachas)

(fifty) | اکاون (akwan) (fifty one) | باون (bawan) (fifty two) | ترپن (trapen) (fifty three) | چون (chawan) (fifty four) | پچپن (pachpan) (fifty five) | چھپن (chapan) (fifty six) | ستاون (satawan) (fifty seven) | اٹھاون (athwan) (fifty eight) | انسٹھ (unsath) (fifty nine) | ساٹھ (saath) (sixty) | اکسٹھ (eksat) (sixty one) | باسٹھ (basath) (sixty two) | ترسٹھ (treseth) (sixty three) | چوسٹھ (chonsath) (sixty four) | پینسٹھ (panseth) (sixty five) | چھیاسٹھ (chaysiat) (sixty six) | سڑساٹھ (sarsath) (sixty seven) | اٹھسٹھ (arsath) (sixty eight) | انھتر (unhather) (sixty nine) | ستر (sather) (seventy) | اکھتر (ekhather) (seventy one) | بھتر (bather) (seventy two) | تھتر (theather) (seventy three) | چوھتر (choather) (seventy four) | پچھتر (pachater) (seventy five) | چھہتر (chather) (seventy six) | ستتر (satater) (seventy seven) | اٹھتر (athter) (seventy eight) | اناسي (unasi) (seventy nine) | اسی (asy) (eighty) | اکاسي (ekasi) (eighty one) | بیاسي (biayasi) (eighty two) | تراسي (tarasi) (eighty three) | چوراسي (chorasi) (eighty four) | پچھاسي (pchasi) (eighty five) | چھیاسي (chiasi) (eighty six) | ستاسي (satasi) (eighty seven) | اٹھاسي (athasi) (eighty eight) | نواسي (nawasi) (eighty nine) | نوے (navay) (ninty) | اکانوے (akanway) (ninty one) | بانوے (banway) (ninty two) | ترانوے (taranway) (ninty three) | چورانوے (churanway) (ninty four) | پچانوے (pachanway) (ninty five) | چھیانوے (chianway) (ninty six) | ستانوے (satianway) (ninty seven) | اٹھانوے (ethanway) (ninty eight) | ننانوے (ninway) (ninty nine) | سو (so) (hundred)| ہزار (hazar) (thousand) | لاکھ (laakh) (lack) | کروڑ (crore) | ارب (erab) | کھرب (kharab) | ملین (million) | بلین (billion)

NEN = Urdu_Digit + (to handle one or more occurrences of a digit)

For NETI (Time), we define regular expressions like U_Digit = " ایک (aik) (one) | دو (do) (two) | تین (theen) (three) | چار (char) (four) | پانچ (panch) (five) | چھ (chey) (six) | سات (saat) (seven) | آٹھ (aath) (eight) | نو (nao) (nine) | دس (das) (ten) | گیارہ (giarah) (eleven) | بارہ (barah) (twelve) | تیرہ (terah) (thirteen) | چودہ (chodah) (forteen) | پندرہ (pandrah) (fifteen) | سولہ (solah) (sixteen) | سترہ (satrah) (seventeen) | اٹھارہ (atharah) (eighteen) | انیس (unees) (nineteen) | بیس (bees) (twenty) | اکیس (akees) (twenty one) | بائیس (baees) (twenty two) | تےایس (tayees) (twenty three) | چوبیس (chobees) (twenty four) | پچیس (bachees) (twenty five) | چھبیس (chabees) (twenty six) | ستائیس (sataees) (twenty seven) | اٹھائیس (athaees) (twenty eight) | انتیس (unatees) (twenty nine) | تیس (tees) (thirty) | اکتیس (aikatees) (thirty one)";

M_Digit = [1-31]

Urdu_Month = جولائی (July) | مارچ (March) | اکتوبر (October)| فروری (February) | نومبر (November) | مئی (May)| دسمبر (December) | اپریل (April) | اگست (August) | ستمبر (September) | جون (June) | جنوری (January) | ربیع الاول (Rabi-ul-Awal) | شعبان (Shaban) | شوال (Shawal) |ذوالقعد (Zulqad) | ذوالحجہ (Zulhaj) | رجب (Rajab) | جمادی الثانی (Jamad-ul-Alsani) | جمادی الاول (Jamad-ul-Awal) | ربیع الثانی (Rabi-ul-Sani) | محرم (Muharam) | صفر (Safar)

Year_Number = (single digit | double digit | three digit | four digit)

Year = Year_Number سـہ | ء

NETI = M_digit Urdu_Month Year_Number | U_digit Urdu_Month Year_Number | M_digit Urdu_Month Year | U_digit Urdu_Month Year | M_digit Urdu_Month | U_digit Urdu_Month | Year | Urdu_Month Year | Urdu_Month Year_Number

Examples are 14 اگست (August) 1947, چودہ (chodah) (forteen) اگست (August) 1947, 14 اگست (August), چودہ (chodah) (forteen) اگست (August), 1971ء, ۱۸۸۶سـہء, 73ء, ۱۸۸۶, and جون (June) ء 2000 etc.

The following rule is used for disambiguation between NETI and NEN. For example, in the sentence “Main nay yeh cheez 2000 main lee” (I bought this thing in 2000). Here 2000 could be NETI or NEN. We used the rule that if a digit is greater than 3000 and less than 100 then it will be most like a NEN, not NETI.

if digit < 3000 and digit > 100 then
NETI
Else
NEN
endif

For NEM (measure), we define regular expressions like Units = کلومیٹر (kilometer), فی گھنٹہ کلومیٹر (kilometer per hour), کلو (kilo), سال (saal) (year), فٹ (feet), میل (mile), ہیکٹر (hector), ٹن (ton), میگاواٹ (mega watt), انچ (inch) etc.

CC = سے (say) | یا (ya) | تا (ta)

NEM = Urdu_Digit+ Units | Digit + Units | Urdu_Digit+ CC Urdu_Digit+ Units | Digit + CC Digit+ Units

Some of examples are 4 سے 9 سال (4 say 9 saal) (4 to 9 years), 40 سے 50 کلومیٹر فی گھنٹہ (40 say 50 kilometer fi ghunta) (40 to 50 kilometer per hour), ایک میل (aik meel) (one mile), ایک یا دو میل (aik ye do meel) (one or two mile), 60 تا 100 فٹ (60 ta 100 foot) (60 to 100 feet), ایک ایک انچ (aik aik inch) (one one inch)

For NEA (abbreviation) we define regular expression like Abb = | آے (A) | بی (B) | سی (C) | ڈی (D) | ئی (E) | ایف (F) | جی (G) | ایچ (H) | آئی (I) | جے (J)| کے (K) | ایل (L) | ایم (M) | این (N) | او (O) | پی (P) || کیو (Q) | آر (R) | ایس (S) | ٹی (T) | یو (U) | وی (V) | | ڈبلیو(W) | ایکس (X) | واے (Y) | زی(Z)

NEA = Abb Abb+

Table 5: Results of Using IOE2 tagging scheme after executing normalization step, Regular expression, names list and POS

NE	Precision	Recall	F-Measure
NEA	100.00	58.33	73.68
NEB	0.00	0.00	0.00
NED	89.47	44.74	59.65
NEL	88.27	77.84	82.73
NEM	100.00	89.91	94.69
NEN	86.00	81.13	83.50
NEO	63.21	37.64	47.18
NEP	63.55	73.12	68.00
NETE	14.29	14.29	14.29
NETI	95.24	76.92	85.11
NETO	2.34	100.00	4.58
NETP	54.55	75.00	63.16
Total	66.71	71.70	69.12

By using the above rules, names list, and the POS information, the improved precision, recall, and f-measure are achieved as 66.71%, 71.70%, and 69.12%, respectively. The detailed results are given in Table 5. For example, دو (do) can represent digit 2 or it can represent word "give" that is a verb. We use the POS information to correctly assign an NE to the word دو (do). Similarly, سو (so) can represent the number 100 or it can represent word "sleep" that is a verb. Thus, we use the POS information to correctly assign the NE to the word سو (so). Likewise, صفر (safar) can represent digit 0 or it can represent a month in the Islamic calendar. Hence, we use the POS information to correctly assign an NE to the word صفر (safar). The same is the case with the word بدھ (budh) that could be a day of the week, i.e., Wednesday or it could be the name of person, i.e., Mahatma Budh (Buddah). Again, we distinguish between the two with the help of the POS information.

5. Conclusion and Future Work

We are not aware of any research in which experimentation on Urdu NER has been performed using the character and word level normalizations, POS information, gazetteers, and rules. Literature review shows that our system has produced the best-known results using 12 NEs of IJCNLP workshop data. To the best of our knowledge, no one has previously carried out experimentation on Urdu data with IOE2 tagging scheme. In future, we can use Urdu NP chunker by [1, 2], another tagging scheme mentioned in [3], CRF, and deep learning approaches to show how these techniques affect results.

References

[1] Siddiq, S., Hussain, S., Ali, A., Malik, K., & Ali, W. (2010, December). Urdu Noun Phrase Chunking-Hybrid Approach. In Asian Language Processing (IALP), 2010 International Conference on (pp. 69-72). IEEE.

[2] Ali, W., Malik, M. K., Hussain, S., Siddiq, S., & Ali, A. (2010, September). Urdu noun phrase chunking: HMM based approach. In Educational and Information Technology (ICEIT), 2010 International Conference on (Vol. 2, pp. V2-494). IEEE.

[3] Malik, M. K., & Sarwar, S. M. (2016). Named Entity Recognition System for Postpositional Languages: Urdu as a Case Study. International Journal of Advanced Computer Science and Applications, 7(10), 141-147.

[4] Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. The annals of mathematical statistics, 37(6), 1554-1563.

[5] Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1), 3-26.

[6] Mukund, S., & Srihari, R. K. (2009, June). NE tagging for Urdu based on bootstrap POS learning. In Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies (pp. 61-69). Association for Computational Linguistics.

[7] Saha, S. K., Sarkar, S., & Mitra, P. (2008, January). A Hybrid Feature Set based

Maximum Entropy Hindi Named Entity Recognition. In IJCNLP (pp. 343-349).

[8] Kumar, P., & Kiran, V. R. (2008, January). A hybrid named entity recognition system for south Asian languages. In proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages (pp. 83-88).

[9] Jahangir, F., Anwar, W., Bajwa, U. I., & Wang, X. (2012, December). N-gram and gazetteer list based named entity recognition for urdu: A scarce resourced language. In 24th International Conference on Computational Linguistics (p. 95).

[10] Saha, S. K., Chatterji, S., Dandapat, S., Sarkar, S., & Mitra, P. (2008, January). A hybrid approach for named entity recognition in indian languages. In Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages (pp. 17-24).

[11] Tafseer, A., Urooj, S., Hussain, S., Mustafa, A., Parveen, R., Adeeba, F., & Butt, M. (2015). The CLE Urdu POS Tagset. In LREC 2014, Ninth International Conference on Language Resources and Evaluation (pp. 2920-2925).

[12] Gali, K., Surana, H., Vaidya, A., Shishtla, P., & Sharma, D. M. (2008, January). Aggregating Machine Learning and Rule Based Heuristics for Named Entity Recognition. In IJCNLP (pp. 25-32).

[13] Naz, S., Umar, A. I., Shirazi, S. H., Khan, S. A., Ahmed, I., & Khan, A. A. (2014). Challenges of Urdu Named Entity Recognition: A Scarce Resourced Language. Research Journal of Applied Sciences, Engineering and Technology, 8(10), 1272-1278.

[14] Ekbal, A., Haque, R., Das, A., Poka, V., & Bandyopadhyay, S. (2008, January). Language Independent Named Entity Recognition in Indian Languages. In IJCNLP (pp. 33-40).

[15] Riaz, K. (2010, July). Rule-based named entity recognition in Urdu. In Proceedings of the 2010 named entities workshop (pp. 126-135). Association for Computational Linguistics.

[16] Singh, U., Goyal, V., & Lehal, G. S. (2012). Named Entity Recognition System for Urdu. In COLING (pp. 2507-2518).

[17] Malik, M. K., & Sarwar, S. M. (2015) "Urdu Named Entity Recognition and Classification System Using Conditional Random Field" Sci-int. 27(5), pp (4473-4477).

[18] Kamran Malik, M., Ahmed, T., Sulger, S., Bögel, T., Gulzar, A., Raza, G., & Butt, M. (2010). Transliterating Urdu for a Broad-Coverage Urdu/Hindi LFG Grammar. In LREC 2010, Seventh International Conference on Language Resources and Evaluation (pp. 2921-2927).

[19] https://www.ethnologue.com/statistics/size

[20] http://ltrc.iiit.ac.in/ner-ssea-08/

[21] http://cle.org.pk/