Extraction of Named Entities from Punjabi-English Parallel Corpora

Kapil Dev Goyal
[1] *Research Scholar, Department of Computer Science,*
Punjabi University, Patiala, Punjab, India

Vishal Goyal
[2] *Research Scholar, Department of Computer Science,*
Punjabi University, Patiala, Punjab, India

Abstract- Names of persons/objects or places are known as named entities and transliteration of named entities play a vital role in the performance of all Natural Language Processing (NLP) tasks. This work is first ever work done on of parallel extraction Named Entities (NEs) from Punjabi-English corpus. We use a transliteration approach to meet our goal. We transliterate Punjabi text to English using the n-gram language model. Then extractions of the parallel Named Entities are done. To develop the transliteration system, we have to train our system copiously, as it is a training-based approach. In our experiment, we had used more than one million parallel Named Entities in Punjabi and English script as a training corpus. We generated Punjabi to English n-gram databases from the corpus. Our n-gram database consists of more than 10 million n-grams and each n-gram having multiple mappings of the other script. The toughest part of the experiment was to find the mapping for the given n-gram from the parallel Named Entity while creating n-gram databases, because the same combination of letters may have different pronunciation depending upon its location in the word. In the extraction of parallel Named Entities from Punjabi-English parallel corpus, we achieved 98.86% accuracy, 79.34% recall, 87.17% f1-score using the gold standard, and 99.37% accuracy, 90.93% recall, 93.45% f1-score accuracy using minimum edit distance.

Keywords – n-gram model, Named Entities, Natural Language Processing, Transliteration

I. INTRODUCTION

Names of persons/objects or places are known as named entities. For example, “Boota Singh”, “New Delhi”, “Knight Riders”, etc. Named Entities in English are basically represented by capital letters, but in Punjabi, it is a very hard task to identify them due to lack of capitalization. NEs play a vital role in performance of many NLP tasks such as machine translations (MT) and cross-lingual information retrieval. Parallel extraction of NEs links the source NE to target NEs, that is the first step to train the NE translation model. In Punjabi there are more than one meaning of a single word so it is difficult to recognize the actual meaning of the word by machine whether given word is NE or other word in given context. E.g.

□□□□ □□□□ □□ □□□□ □□□□□)Punjabi) Transliteration: “Būṭā sigha nē būṭā lagā'i'ā”

Gloss: Plant Singh planted the plant Translation: Buta Singh planted the plant.

In this Punjabi sentence, □□□□ comes at two places. At first place □□□□ acts as a NE and at second place, it acts as Noun. If someone ignores the importance of NE, then translation will not be correct.

Our main objective is to extract parallel named entities from Punjabi-English bilingual corpus using n-gram transliteration system. Transliteration means to convert text of one script to another without effecting pronunciation [1]. Transliteration is not only concerned with representing the sounds of the original language but also represents the characters accurately and unambiguously. In this paper, we use the transliteration system for extracting parallel Named Entities.

- In the first phase, we train our system using a Punjabi-English parallel named entities corpus and create an n-gram database.
- In the second phase, these n-gram database is used to develop a Punjabi-English transliteration system.
- In the third phase, we extract parallel Named Entities from the Punjabi-English parallel corpus using a transliteration system.

This research paper is organized as follows: Related Work is discussed in section II. In section III, Methodology is illustrated. Results are discussed in section IV and Finally, Conclusion and future scope are summed up in section V.

II. RELATED WORK

[2] presented a novel algorithm for translating named entity phrases from Arabic to English using a limited amount of monolingual and bilingual resources. There had been limited work done on the extraction of parallel NEs. Mainly three approaches had been used for extraction of NEs. These approaches are linguistic approaches (Rule-

based approaches), machine learning (ML) based approaches, and the hybrid approach. Most of the researchers used the linguistic approach [3]. Linguistics approaches require a large set of rules, experience, and grammatical knowledge of the related domain, and also this approach is language-specific and cannot be transferred to other domains or language [4]. [5] used aligned parallel texts to extract the candidates. After the texts are word-aligned, they extract sequences of length two or more in the source language that is aligned with sequences of length one or more in the target. Candidates are then filtered out of this set if they comply with pre-defined part-of-speech patterns, or if they are not sufficiently frequent in the parallel corpus.

Apart from this, ML approach is also known as statistical approach and it requires a large volume of data to develop an analytical model. ML approach involves the supervised learning approach, which is mainly used to automatically develop annotation rules. [6] proposed a linear chain Conditional Random Field method which projects features between English and Chinese through word alignment. The information is transferred on the feature-level. The model combined both monolingual and bilingual features and performed decoding on two languages simultaneously to help improve the tagging process. [7] coined an integrated approach that was used to extract a bilingual named entity translation/transliteration dictionary from a bilingual corpus for Chines-English language pair, also improved the named entity annotation quality. First NEs were extracted from bilingual corpus independently for each language and then using a statistical alignment model, NEs were aligned and extract NEs pair having higher alignment probability and improved F-score from 73.38 to 81.46 and annotation quality from 70.03 to 78.15 for Chinese. [8] proposed a method that formulates the problem of exploring complementary cues about entities on an unannotated parallel corpus between English and Chinese. They used integer linear programming to enforce entities to agree through bilingual constraints. This method could jointly tag named entities in both languages without any annotated data. [9] presented intuitive and effective heuristics to project English named entities into Chinese ones. Results showed that the generated corpus achieved comparable results to a manually annotated corpus in Named Entity Recognition task. This method could be expanded to different domains to solve the common domain over-fitting problem. [10] used support vector machine for extracting Named Entities while [11] used Hidden Markov Model (HMM) which is graphics-based modelling approach. [12] use maximum entropy approach.

The hybrid approach uses both linguistic and ML approaches. [13] use a hybrid approach for their research. [8] presented a joint approach by combining two conditional random fields (CRF) NER taggers and two Hidden Markov Model (HMM) word aligners and improved in both NER and word alignment. [14] used a hybrid NER system using conditional random fields (CRF), which integrates Rule-based and Machine learning methods. Named Entities lexicon were extracted from DBpedia linked datasets to improve the rule-based system and ML was used to improve the rule-based component. [15] explore the use of bilingual resources to improve monolingual Named Entity Recognition systems of English and Chinese. Their proposed system managed to improve in Chinese NER performance. In particular, the F1-score of Chinese NER increase significantly from 42.83% (StanfordNER) and 57.65% (Che2013) to 63.64%. Regarding the English side, they managed to outperform StanfordNER, in which F1-score increase from 75.75% to 76.08%.

In our approach, we extracted parallel Named Entities using the transliteration system. Work-related to transliteration is as follows.

Rule-based machine transliteration was the first-ever technique used in the transliteration. In this technique Mapping of patterns of the source language to the patterns of the target language is done according to the set of predefined rules [16]. Grapheme based models are popular models in expression transliteration. They are further categorized as the rule-based approach, statistical approach, HMM (Hidden Markov Model) approach, and FST (Finite State Transducers) approach [17]. In SMT (statistical machine transliteration) we assume that every sentence in the target language has some probability to represent the given sentence in the source language. We choose the sentence with the highest probability. FST (finite-state transducers) are automation to covert the string of source language to the target language. The string is fed token by token to the finite state machine and while transitioning from one state to the next state, letters of the source language are mapped to the letters of destination or target language. Finite state machines were used by Stall et al. for Arabic to English transliteration [18]. [19] used the HMM model to transliterate Russian to English. The Viterbi algorithm was used where the observed sequence of source language text is mapped with the hidden or unknown sequence of the target language. [20] developed a rule-based system for Punjabi to Hindi transliteration. Due to, many to one mapping this system cannot be simply reversed from Hindi to Punjabi. [21] developed a web-based application for Hindi to Punjabi translation system. They also added Hindi to Punjabi transliteration module for the words which are not found in the parallel dictionary. [22] used the bi-gram tables for Punjabi-English transliteration. The bi-gram tables have different probabilities for names (person and location) and simple texts. Therefore, first of all, they tag the Named Entities in the given text and transliterate them separately. The version with the least perplexity according to the n-gram table is chosen as an acceptable transliterated sentence. [23] proposed a rule-based model for Punjabi to English machine transliteration. They use proper nouns as a key. They trained the system using the parallel corpus and created the bi-gram, tri-gram, 4-gram, 5-gram, and 6-gram tables. They defined a mapping between Punjabi and English script. The input script is first looked up in the dictionary, then n-gram tables are consulted. They claimed 96% accuracy.

Above were the different techniques used by different researchers for different languages for the extraction of the parallel name entities and transliteration system. There is no work done on Punjabi to English transliteration system using n-gram model, so we have used n-gram model to transliterate Punjabi to English. There is a no work done on Punjabi language in extraction of parallel Punjabi-English named entities, so we are using our own hybrid approach for extraction of parallel Named Entities (NEs) from Punjabi-English corpus. In our approach, we extracted parallel Named Entities using n-gram and transliteration system.

III. METHODOLOGY

Our system works in three phases.

- In the first phase, we train our system using a Punjabi-English parallel named entities corpus and create an n-gram database.
- In the second phase, these n-gram database is used to develop a Punjabi-English transliteration system.
- In the third phase, we extract parallel Named Entities from the Punjabi-English parallel corpus using a transliteration system.

A. Generating N-Grams Databases

In the first phase, Punjabi-English parallel named entities corpus is used to train our system and create n-gram database. A corpus of 1,020,660 parallel Punjabi-English Named Entities from P.S.E.B. Mohali was used to create n-gram database. n-gram, in this context, refers to the sequence of n contiguous letters of Gurmukhi script. n-gram database contains all possible n-grams mapping from Punjabi to English. In this process, we created all possible n-grams from bi-grams to till 30-grams for the language pair Punjabi to English.

1) Arrangement of English and Punjabi Names

A parallel corpus provided by PSEB Mohali was arranged in the following way:

aman@□□□
amita@□□□□□
anjali@□□□□□
ankit@□□□□□

2) Generating Punjabi-English N-gram Database

For generating the Punjabi-English n-gram database, first of all those strings which are not valid names are filtered out. The names which contain numerals or other symbols except Punjabi and English letters are considered as invalid names. The next process is explained step by step as below.

1. We separate all English names and Punjabi names by symbol '@'.
2. Then iteratively take one named entity at a time and repeat steps 3 to 7.
3. Split the Punjabi name into all possible n-grams (bi-gram to n-gram Maximum 30-gram).
4. For each Punjabi n-gram, we scan the Punjabi n-gram left to right, character by character and try to find corresponding English characters from English name using the Punjabi-English Unigram mapping table.
5. If we successfully find all corresponding English characters from English name, then we cut the English name from the first mapped character to the last mapped character.
6. If Punjabi n-gram occurs at the beginning of Punjabi name, then append _S, if it occurs at the end of Punjabi name, then append _E, otherwise append _M.
7. Add Punjabi n-gram and corresponding English substring to n-gram dictionary database as key-value pair, in which Punjabi n-gram was taken as key and English substring was taken as value.

While adding key-value pairs into the n-gram dictionary, there may be three cases.

Case 1: If the key does not exist in the n-gram database, then add a key-value pair and set the frequency of value as one.

Case 2: Otherwise if the key-value and corresponding value exists, then simply increment the frequency of corresponding value by one.

Case 3: If the key already exists and corresponding value does not exist, then simply add corresponding value as the new value and set frequency as one.

Thus n-gram can store more values corresponding to the one key along with their frequencies.

8. In the last step, for each key, sort all values in descending order by their frequencies.

Table 1 shows the all possible n-grams of Punjabi name "□□□□" and Punjabi-English n-gram database.

Table -1 Possible n-grams of Punjabi Name: ਭਾਰਤ

Type	Key	Value
Bi-gram	**□□_S**	**BHA_1879 BHHA_2**
Bi-gram	**□□_M**	**AR_7538 AAR_231 AHAR_152**
Bi-gram	**□□_E**	**RAT_708 RT_166 RET_58 RRAT_40**
Tri-gram	**□□□_S**	**BHAR_300 BHAAR_6**
Tri-gram	**□□□_E**	**ARAT_52 ART_18 AHARAT_2 AHRAT_2 AARAT_2**
4-gram	**□□□□_S**	**BHARAT_54 BHART_50 BHAARAT_2 BHAERT_2**

Table 1 shows the all possible n-grams of Punjabi Name □□□□ and also shows key-value pairs of Punjabi n-grams. The length of Punjabi name □□□□ is four, so maximum possible n-gram is 6-gram. Total numbers of possible n-grams can be calculated using the following expression 1:

$$\frac{n(n-1)}{2} \quad (1)$$

Here n is the length of a Punjabi name.

In the case of Punjabi Name □□□□, the length is 4 and the total numbers of n-gram are 6 (□□, □□, □□, □□□, □□□, □□□□). Table 1 shows that □□_S, □□□_S and □□□□_S occur at the beginning of □□□□, so that _S is appended to each n-gram. Similarly, □□ and □□□ have occurred at the end of the Punjabi words, so _E is appended and _M is appended to all other remaining n-grams. From Table 1 BHA_1879 means BHA is mapped 1879 times at the beginning for Punjabi n-gram □□_S. Similarly, RAT_708 means RAT is mapped 708 times in the middle for n-gram □□_M.

B. Implementation of Punjabi to English Transliteration System

In Punjabi to English transliteration, our system takes Punjabi names as inputs and generates all possible English transliterated names for each Punjabi name. The whole step by step process of Punjabi to English transliteration is described below.

1. It splits all Punjabi names by new line character and by blank space into a list of Punjabi names and set output list is Empty.
2. For each Punjabi name in List of Punjabi names, repeat steps 3 and 4
3. Append "_S" string to the end of Punjabi Name. Suppose name is □□□□, after appending it becomes "□□□□_S".
4. Call NGram function from Algorithm 2 and pass Punjabi name as argument, then NGram returns all possible transliteration for Punjabi Name and save to list of English Names. In this step, a list of all English Names is appended to the output list.
5. Print or return the output list.

The algorithm 1 explains the process in more detail

1) Algorithm 1: PE_TransliterationSystem( ) Generating Punjabi to English Transliteration System

```
Input: PunjabiNames
Output: resultOutput
resultOutput:= Empty
ListOFPunNames:= PunjabiNames. splitByLinesAndSpaces()
foreach PName in ListOFPunNames do {
        PName.append( "_S" )
        listOfEngNames = call Algorithm 2: NGram(PName)
        resultOutput.append( "   ")
        resultOutput.append(listOfEngNames)
}
return resultOutput
```

2) Algorithm 2: NGram(NameStr) Recursive function to transliterate NameStr

```
Input: PE_ngramDatabase, UniMapTable, NameStr
Output: listOfNamesStr
if PE_ngramDatabase.findKey[NameStr] <> Null then {
return PE_ngramDatabase[NameStr]. Values
}
NameLength = NameStr.length -2
if NameLength = 1 then {
```

```
return UniMapTable.find( NameStr[0])
}
strPosition = NameStr.subStr( NameLength, NameLength +1)
for index = NameLength – 1 to 0 do {
        newStr = NameStr.subStr( 0,index).append(strPosition)
        if PE_ngramDatabase.findKey[newStr] <> Null then {
        break
 }      }
if strPosition = "_S" then {
        StrLeft = NameStr.subStr( 0,index).append( "_S" )
        StrRight = NameStr.subStr( index, NameLength -1).append( "_E" )
}
else if strPosition = "_E" then {
        StrLeft = NameStr.subStr( 0,index).append( "_M" )
        StrRight = NameStr.subStr( index, NameLength -1).append( "_E" )
}
else {
        StrLeft = NameStr.subStr( 0,index).append( "_M" )
        StrRight = NameStr.subStr( index, NameLength -1).append( "_M" )
}
list1 = call Algorithm 2: NGram(StrLeft)
list2 = call Algorithm 2: NGram(StrRight)
listOfEngNames = combineList( list1,list2)
return listOfEngNames
```

3) *Working of recursive n-gram function*

In the case of algorithm 2, it shows the working of recursive NGram function. This algorithm finds the longest possible n-gram from n-gram database for given name string, then split the name recursively until the match is found. After that, all transliterated names are merged back to generate the final list of output.

1. Set the output list as Empty.
2. Check whether the given name string is already existing in the n-gram database or not. If it exists then append all corresponding transliterated names to the output list for given Name String and return the output list. Otherwise, go to step 3.
3. Calculate the length of the name string without including its position string ("_S", "_E", "_M") to NameLength variable.
4. If the length of the name string is one, then search the corresponding transliterated character/s from the Unicode Mapping Table and save it to the output list. Then return the output list.
5. In this step, the algorithm finds the longest substring from beginning to maximum possible length from n-gram database and store maximum possible length to the Index variable. Otherwise set Index as 1.
6. In this step, the algorithm splits the name string into two parts using the Index value. The left string will be from starting to index and the right string will be form Index to the last character of name string. Both left and right strings contain index character. In the left string index character will be last and in right string index character will be first.
7. Call function NGram recursively for left string and right string and save results to list1 and list2 respectively.
8. In this step, the algorithm combines the name list1 and list2 into the output list. Suppose list1 contains m names and list2 contains n names, then the output list has m*n names. While combining, if the last character of the name of list1 and the first character of the name of list2 is the same, then it removes one common character, then combines.
9. Return output list.

4) *Executing sample input for Punjabi to English transliteration*

For Punjabi to English transliteration consider a sample input "□□□□□□□□□□□□□". First, it calls Algorithm 1 (PE_Transliteration System), and appends "_S" to sample text and it becomes "□□□□□□□□□□□□□_S". Then, it calls NGram function for the name string "□□□□□□□□□□□□□_S" and searches the n-gram from n-gram database. If this n-gram does not exist in n-gram database, then it finds the longest possible string which is present in n-gram database i.e. "□□□□□□□_S" which is already presented in our database.

Now, a recursive process begins that splits this name string into two substrings as “□□□□□□□□_S” and “□□□□□□□□ _E”. Now,the name string “□□□□□□□□_S” exists in n-gram database and its corresponding transliteration is saved to output list and returns. In the case of name string “□□□□□□□□ _E” which does not exist in our database, it splits it further into two substrings “□□_M” and “□□□□□□□_E” and again calls the Ngram function for each. Now both substrings exist in our database and the corresponding outputs are returned. After that returned outputs are merged back. The recursion tree in figure 1 shows the splitting process from level 1 to level 4 of transliteration system for sample input “□□□□□□□□□□□□□□” and Table 2 also shows whether the name sting exists in n-gram database or not. If it does not exist, then it shows “not found” and otherwise shows the top two results returned by the n-gram database.

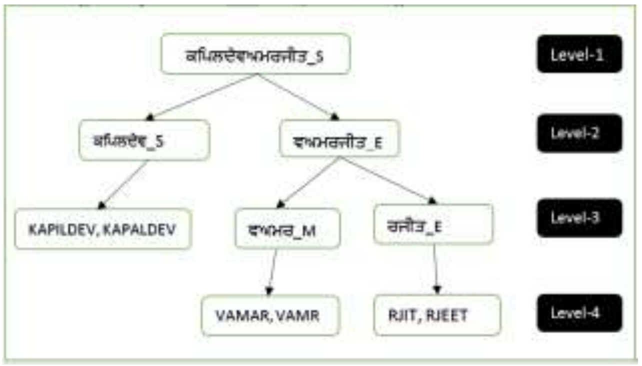

Figure 1. Recursive process for splitting for Punjabi to English Transliteration System.

Table -2 Top two results of Ngram Database for Punjabi to English Transliteration System

Steps	**Input String**	**Most 2 Output String**	
1	□□□□□□□□□□□□□□**_S**	**Not Found**	
2	□□□□□□□**_S**	**KAPILDEV**	**KAPALDEV**
3	□□□□□□□**_E**	**Not Found**	
4	□□□□**_M**	**VAMAR**	**VAMR**
5	□□□□**_E**	**RJIT**	**RJEET**

Figure 2 shows the recursive process of merging back results into the final output. As principle of stack, it starts from level 4 and ends at level 1 by combining the results of each recursive process using a bottom-up approach.

Final output is
KAPILDEVAMARJIT
KAPILDEVAMARJEET
KAPALDEVAMARJIT
KAPALDEVAMARJEET

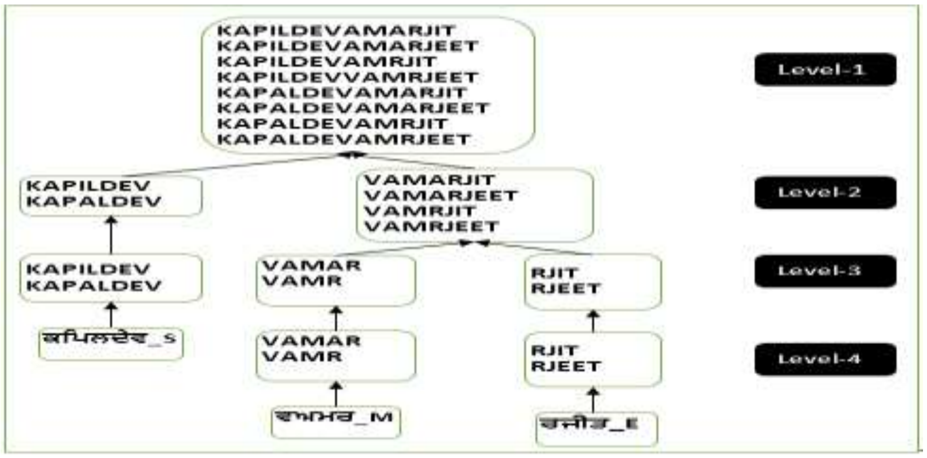

Figure 2. Recursive process of merging back the results into Final output for Punjabi to English Transliteration System.

C. Extraction of parallel named entities from Punjabi-English parallel corpus

The final phase of our approach is the extraction of parallel named entities phase. Algorithm 3 illustrates the step by step process for extracting parallel Named Entities from Punjabi-English corpus.

Punjabi-English parallel corpora are given as input. In the pre-processing step, it cleans the parallel sentences by removing extra white spaces and separate punctuations marks. Mostly all punctuations marks are attached with a word and become part of that word, and it poses problems in the transliteration system. Therefore, punctuation marks should be separated from a word by space.

After pre-processing, Punjabi and English corpora are split into sentences. If number of Punjabi and English sentences are the same then it proceeds for further execution, otherwise, it stops. Next, it iteratively takes Punjabi and English sentence one by one and splits each sentence in words and save them to the list of Punjabi and English words respectively. Then, for each Punjabi word, first, it checks whether the given Punjabi word is in the list of Punjabi stop words. A list of Punjabi stop words is a collection of most frequent Punjabi words, which cannot be part of any Named Entity. If a given Punjabi word is in the list of Punjabi stop words, then it simply ignores this given Punjabi word and goes for next Punjabi word, otherwise, it passes to the next step. This Punjabi stop word check is implemented to increase the efficiency of our algorithms because it rejects many of the Punjabi words without involving any transliteration process which cannot be any valid Named Entities.

If a given Punjabi word is not in the list of Punjabi stop words, then it passes to the transliteration system. After passing the Punjabi word to the transliteration system, it returns the list of possible English transliterated words. Next, it matches each English transliterated word within the list of English words to each word of English sentence using a minimum edit distance (MED) algorithm. Minimum Edit distance computes the minimum numbers of edit operations (insert/delete/substitute) to transform one string to another. For example, if string1 = "AMAN" and string2 = "NAMAM", then to transform string1 to string2 only one insertion operation is required. Similarly, to convert string2 to string1, only one delete operation is required. Apart from MED, we also calculate the MED Percentage using expression 2.

MED percentage: (Cost of MED × 100)/ (Length of Source String) (2)

In the case of string1 = “AMARJEET”, string2 = “AMARJIT”, The cost of minimum edit distance from string1 to string2 is 2. Then using expression 2, MED Percentage = (2 × 100)/8 = 25

In the extraction of parallel Named Entities system, we set MED Percentage to 30%. While comparing two strings, if MED percentage is less than 30%, then we assume that these two strings are similar, otherwise, strings are dissimilar. Thus, if MED Percentage of each English transliterated word and each word of English sentence is less than equal to 30%, then we save the Punjabi word along with its transliteration to Punjabi to English Named Entities Database.

1) Algorithm 3: Extracting Name Entities from given Pun-Eng Parallel Corpra

```
Input: ListOfPunSentences, ListOfEngSentences, PunStopsWords, PE_TransliteratationSystem
Output: ParallelPunEngNameEntities
TotalPunSentences = ListOfPunSentences.count()
TotalEngSentences = ListOfEngSentences.count()
if TotalPunSentences <> TotalEngSentences then {
return Error
 }
for iIndex =1 to TotalPunSentences do {
          ListOfPunWords = ListOfPunSentences[iIndex]. splitIntoWords()
          ListOfEngWords = ListOfEngSentences[iIndex]. splitIntoWords()
          foreach PunWord in ListOfPunWords do {
                    if PunStopsWordsTable.contains(PunWord) then {
                              continue
                     }
                    ListOfPun2EngTransliterateResult = PE_TransliterationSystem( PunWord)
                    foreach EngWord in ListOfEngWords do {
                              foreach P2ETransWord in ListOfPun2EngTransliterateResult do {
                    MEDDist = MinEditDist(EngWord, P2ETransWord) MEDPercentage =
                    MEDDist*100/ P2ETransWord.Length
                                        if MEDPercentage <= 30 then {
                                        ParallelPunEngNameEntities.add( PunWord +”@ ” +EngWord )
}         }         }         }         }
```

return *ParallelPunEngNameEntities*

2) *Executing Sample Input For Extracting Parallel Named Entities*

Consider the following sample input for Punjabi and English sentences:

Punjabi Sentences: □□□□ □□□ □□□□ □□□ □□□□ □□

English Sentences: My Name is Kapil Dev Goel

After preprocessing, Punjabi and English sentences are split into words. For each Punjabi word, first, it will check for Punjabi stop word, if it is not Punjabi stop word, then it will go for transliteration steps. Table 3 shows that Punjabi word is 'stop word' or not and if it is not Punjabi 'stop word', then corresponding top most transliteration results.

Table - 3 Top most transliteration results

Punjabi Word	Is Punjabi Stop Words?	Top Most trasliteration results
□□□□	**Yes**	-
□□□	**No**	**NAM NAAM NAHM**
□□□□	**No**	**KAPIL KAPAL KAPEL KPIL KAPEEL**
□□□	**No**	**DEV DEW DEAV**
□□□□	**No**	**GOEL GOYAL GOYEL**
□□	**Yes**	-

In this sample, □□□□ and □□ are Punjabi stop words, whereas □□□, □□□□, □□□, and □□□□ are not 'stop words' and passed for the transliteration process. The top results of transliteration system are given in table 3. After that each transliterated word is searched in corresponding English sentence, if it is found in English sentence, then it saves to output database. Therefore only "Kapil", "Dev" and "Goel" transliterated words are matched and the result is

□□□□ □□□ □□□□@ Kapil Dev Goyal

IV. RESULTS OF EXTRACTION OF PARALLEL NAMED ENTITIES

Our extraction system is mainly based on a transliteration system in which pronunciation is the same but the script is different. Mostly these types of words are Named Entities. At last 1000 parallel Punjabi-English sentences are taken for testing purposes in which total Punjabi words are 11,648 and total English words are 9594. We manually found 1036 English and Punjabi Named Entities in which Punjabi Named Entities are 518 and English Named Entities are 518.

To extract parallel Named Entities, two approaches are used. First is Gold Standard and the other is the minimum edit distance (MED) approach. In the gold standard approach, we consider the named entity as correct if transliterated word exactly matches the English word, whereas in MED approach, instead of exact match, if more than 70% of characters of both the strings are matched, then it to be considered as a correct match. Confusion matrix of gold standard approach and MED approach are given below:

Table - 4 Confusion Matrix using the gold standard

Confusion Matrix	Predicted Positive		Predicted Negative	
Actual Positive	**822**	**True Positive (TP)**	**214**	**False Negative (FN)**
Actual Negative	**28**	**False Positive (FP)**	**20178**	**True Negative (TN)**

Table - 5 Confusion Matrix using Minimum Edit Distance

Confusion Matrix	Predicted Positive		Predicted Negative	
Actual Positive	**942**	**True Positive (TP)**	**94**	**False Negative (FN)**
Actual Negative	**38**	**False Positive (FP)**	**20168**	**True Negative (TN)**

Table 4 represents the confusion matrix using the gold standard approach. It shows that using the gold standard approach, out of 1036 named entities, only 850 named entities are extracted by our system, in which 822 are correct named entities, but 28 are not.

Table 5 shows that using the MED approach, out of 1036 named entities, our system extracts 980 named entities, in which 942 are correct, but 38 are incorrect.

In Table 4 and Table 5, true positives are correctly extracted named entities and false positives are incorrectly extracted named entities. Whereas true negatives are correctly rejected Non-Named Entities and false negatives are incorrectly rejected Non-named entities.

To evaluate the performance of our system following four parameters are calculated:

Accuracy, precision, recall [24], and f1-score.

In expression 3, Accuracy is the ratio of the sum of correctly extracted and rejected to total numbers of words.

$$\frac{TP + TN}{TotalWords} \times 100 \quad (3)$$

Precision in expression 4, is the ratio of the correctly extracted named entities to the total extracted named entities

$$\frac{TP}{TP + FP} \times 100 \quad (4)$$

In expression 5, Recall is the ratio of the correctly extracted named entities to the actual total named entities.

$$\frac{TP}{TP + FN} \times 100 \quad (5)$$

F1-score is given in expression 6, is the weighted average of recall and precision

$$\frac{Recall \times Precision}{Recall + Precision} \times 2 \quad (6)$$

Table - 6 Results of Parallel Named Entities

Results	**Gold Standard**	**MED**
Accuracy	**98.86**	**99.37**
Precision	**96.70**	**96.12**
Recall	**79.34**	**90.93**
F1-Score	**87.17**	**93.45**

Using MED approach, we get relatively high performance as compared to the gold standard approach. MED approach increases the recall by more than 10% and F1-score by more than 6%, but it slightly reduces the performance in term of precision, because of increase in the extraction of the incorrect named entities.

V. CONCLUSION

We had extracted parallel Named Entities from Punjabi-English corpus by using n-gram approach. These extracted parallel NEs can be used to improve the performance of many NLP tasks, such as word alignment in bilingual corpus, machine translation (MT) and many more. Our system is mainly based on a transliteration system, in which pronunciation of words is the same but script may differ. Our n-gram database consists of more than 10 million n-grams and each n-gram having multiple mappings of the other script. The toughest part of the experiment was to find the mapping for the given n-gram from the parallel Named Entity while creating n-gram databases. Because the same combination of letters may have different pronunciation depending upon its location in the word. In our extraction process, by using a Gold standard approach, out of 1036 named entities only 850 named entities are extracted. From which 822 are correct named entities, but 28 are wrong. Similarly, by using MED approach, out of 1036 named entities, our system extracts 980 named entities, from which 942 are correct, but 38 are incorrect. Thus, we achieved 98.86% accuracy, 79.34% recall, and 87.17% f1-score using the gold standard. Also, by using minimum edit distance we achieved 99.37% accuracy, 90.93% recall, and 93.45% f1-score. In future this method can be used in bidirectional extraction and its recall and f1-score can be increased.

REFERENCES

[1] S. Karimi, F. Scholer, and A. Turpin, "Machine transliteration survey," *ACM Comput. Surv.*, vol. 43, no. 3, 2011, doi: 10.1145/1922649.1922654.

[2] Y. Al-Onaizan, K. K.-P. of the 40th A. M. On, and U. 2002, "Translating named entities using monolingual and bilingual resources," *dl.acm.org*, 2002, Accessed: Jun. 12, 2020. [Online]. Available: https://dl.acm.org/citation.cfm?id=1073150.

[3] K. Riaz, "Rule-based Named Entity Recognition in Urdu," Association for Computational Linguistics, 2010. Accessed: Jun. 13, 2020. [Online]. Available: https://dl.acm.org/citation.cfm?id=1870476.

[4] R. Alfred, L. C. Leong, C. K. On, and P. Anthony, "Malay Named Entity Recognition Based on Rule-Based Approach," *Int. J. Mach. Learn. Comput.*, vol. 4, no. 3, pp. 300–306, 2014, doi: 10.7763/ijmlc.2014.v4.428.

[5] H. de M. Caseli, A. Villavicencio, A. Machado, and M. J. Finatto, "Statistically-driven alignment-based multiword expression identification for technical domains," no. August, p. 1, 2009, doi: 10.3115/1698239.1698241.

[6] Q. Li, H. Li, H. Ji, W. Wang, J. Zheng, and F. Huang, "Joint bilingual name tagging for parallel corpora," *ACM Int. Conf. Proceeding Ser.*, no. 1, pp. 1727–1731, 2012, doi: 10.1145/2396761.2398506.

[7] F. Huang and S. Vogel, "Improved Named Entity Translation and Bilingual Named Entity Extraction," 2002. Accessed: Jun. 12, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1167002/.

[8] M. Wang, W. Che, and C. D. Manning, "Joint word alignment and bilingual Named Entity Recognition using dual decomposition," *ACL 2013 - 51st Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, vol. 1, pp. 1073–1082, 2013.

[9] R. Fu, B. Qin, and T. Liu, "Generating Chinese named entity data from parallel corpora," *Front. Comput. Sci.*, vol. 8, no. 4, pp. 629–641, 2014, doi: 10.1007/s11704-014-3127-5.

[10] K. Dashtipour, M. Gogate, … A. A.-2017 I. 16th, and U. 2017, "Persian named entity recognition," *ieeexplore.ieee.org*, Accessed: Jun. 13, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8109733/.

[11] D. Chopra, N. Joshi, I. M.-2016 S. International, and U. 2016, “Named entity recognition in Hindi using Hidden Markov model,” *ieeexplore.ieee.org*, Accessed: Jun. 13, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7546675/.

[12] M. Jiljo, M. P.-I. J. of Applied, and undefined 2016, “A study on named entity recognition for malayalam language using tnt tagger & maximum entropy markov model,” *pdfs.semanticscholar.org*, Accessed: Jun. 13, 2020. [Online]. Available: https://pdfs.semanticscholar.org/79fa/ab5cbc0b1dc79dfb879431292d276c261f95.pdf.

[13] K. S. Bajwa and A. Kaur, “Hybrid Approach for Named Entity Recognition,” in *2nd International Conference on Computer Science and Engineering, UBMK 2017*, 2015, vol. 118, no. 1, pp. 474–479, doi: 10.1109/UBMK.2017.8093439.

[14] E. Hkiri, M. Zrigui, and S. Mallat, “Integrating Bilingual Named Entities Lexicon with Conditional Random Fields Model for Arabic Named Entities Recognition,” *ieeexplore.ieee.org*, pp. 609–614, 2017, doi: 10.1109/ICDAR.2017.105.

[15] A. T. Dao, T. H. Truong, L. Nguyen, and D. Dinh, “Improving Named Entity Recognition using Bilingual Constraints and Word Alignment,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 435, no. 1, 2018, doi: 10.1088/1757-899X/435/1/012007.

[16] J.-H. Oh and K.-S. Choi, “An English-Korean transliteration model using pronunciation and contextual rules,” no. 1, pp. 1–7, 2002, doi: 10.3115/1072228.1072327.

[17] K. Kaur and P. Singh, “Review of Machine Transliteration Techniques,” *Int. J. Comput. Appl.*, vol. 107, no. 20, pp. 13–16, 2014, doi: 10.5120/18866-0061.

[18] B. G. Stalls and K. Knight, “Translating Names and Technical Terms in Arabic Text.” Accessed: Jun. 26, 2020. [Online]. Available: https://www.aclweb.org/anthology/W98-1005.pdf.

[19] P. Nabende, “Transliteration System using pair HMM with weighted FSTs,” 2009, Accessed: Jun. 26, 2020. [Online]. Available: http://www.dspace.mak.ac.ug/handle/10570/380.

[20] V. Goyal and G. S. Lehal, “Hindi-Punjabi Machine Transliteration System (For Machine Translation System),” *Georg. Ronchi Found. Journal, Italy*, vol. 64, no. 1, p. 2009, 2009.

[21] V. Goyal, G. L.-J. of E. T. in W. Intelligence, and undefined 2010, “Web based Hindi to Punjabi machine translation system,” *Citeseer*, Accessed: Jun. 26, 2020. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.348.3640&rep=rep1&type=pdf#page=72.

[22] D. Bhalla, N. Joshi, and I. Mathur, “Rule Based Transliteration Scheme for English to Punjabi,” *Int. J. Nat. Lang. Comput.*, vol. 2, no. 2, pp. 67–73, 2013, doi: 10.5121/ijnlc.2013.2207.

[23] A. Kaur, V. G.-2018 3rd I. C. O. Internet, and undefined 2018, “Punjabi to English Machine Transliteration for Proper Nouns,” *ieeexplore.ieee.org*, Accessed: Jun. 26, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8519877/.

[24] M. Junker, R. Hoch, and A. Dengel, “On the evaluation of document analysis components by recall, precision, and accuracy,” *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, pp. 717–720, 1999, doi: 10.1109/ICDAR.1999.791887.