I.J. Modern Education and Computer Science, **2018, 1, 36-45**
Published Online January 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.01.04

A Partial String Matching Approach for Named Entity Recognition in Unstructured Bengali Data

Nabil Ibtehaz
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh
Email: nabilibtehaz2@gmail.com

Abdus Satter
Institute of Information Technology, University of Dhaka, Dhaka 1000, Bangladesh
Email: bit0401@iit.du.ac.bd

Received: 30 October 2017; Accepted: 29 November 2017; Published: 08 January 2018

***Abstract*—**In today's data driven, automated and digitized world, a significant stage of information extraction is to look for special keywords, more formally known as 'Named Entity'. This has been an active research topic for more than two decades and significant progresses have been made. Today we have models powered by deep learning that, although not perfect, have near human level accuracy on certain occasions. Unfortunately these algorithms require a lot of annotated training data, which we hardly have for Bengali language. This paper proposes a partial string matching approach to identify a named entity from an unstructured text corpus in Bengali. The algorithm is a partial string matching technique, based on Breadth First Search (BFS) search on a Trie data structure, augmented with dynamic programming. This technique is capable of not only identifying named-entities present on a text, but also estimating the actual named-entities from erroneous data. To evaluate the proposed technique, we conducted experiments in a closed domain where we employed this approach on a text corpus with some predefined named entities. The texts experimented on was both structured and unstructured, and our algorithm managed to succeed in both the cases.

***Index Terms*—**Named Entity Recognition, Dynamic Programing, Trie, String Matching, Edit Distance.

I. Introduction

Named Entity Recognition problem (NER) holds a very important position in the domain of Natural Language Processing (NLP) and Information Retrieval (IR) [1]. In formal words, a Named Entity (NE) is some abstract or real object, which can be a person, a location, an organization or even numerical data that can be classified and denoted with a proper name. Named-entity recognition (NER) is a task of Information Extraction (IE) that identifies and tags named entities from a text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, numerical values etc. Early approaches to solve this problem used handcrafted algorithms whereas now with the advancement of data science, data mining and access to big data we are fortunate to employ the power of machine learning for solving this problem. However, we do not have much structured and annotated data for Bengali. So, we cannot use the state of art machine learning models to solve this problem. This is why our paper is limited to developing a partial string matching approach for solving the NER problem.

In today’s world scenario, a lot of our tasks are automated. Previously which were done by human agents are now being done by computers. A very popular example is scanning zip codes in USA by OCR technology. The time is not much far when all our day to day tasks will be governed by computers. Information plays a vital and inseparable role in our day to day life. A lot of our dealings is done by textual data. So, we need robust systems to retrieve information from textual data. These are active research areas of fields like NLP, IR etc. NER covers a fair part of retrieving information from textual data. If we manage to identify Named Identities from a text, then the text becomes structured and it becomes easier to parse a semantic meaning from it. Motivated from these needs, this paper tries to explore string matching based approach to solve the problem of NER for Bengali language.

Literature contains several methods to solve NER problem. From [1] we see that most of the NER research tasks have been devoted to the study of English language, since we have huge amounts of annotated data for English. Moreover English is a simple yet very well structured language which allows us to solve this NER problem for English with higher ease. For example, all nouns in English starts with a capital letter, a number of observations like this make the task much easier. However, some other research works address language independence and multilingualism problems. German is well studied in CONLL-2003 [2] and in earlier works. Similarly, Spanish and Dutch were also explored by a major devoted conference: CONLL-2002 [3]. Japanese had been studied in the MUC-6 conference [4], the IREX conference, and other works. French, Greek, Italian, Chinese is studied in an abundant literature. Many other languages also received some attention for example, Basque, Bulgarian, Catalan, Cebuano, Danish, Hindi, Korean, Polish etc. But unfortunately very few work has been done for Bengali language, a notable one being [5].

The technique proposed in this paper involves a partial string matching algorithm to solve Named Entity Recognition problem. It not only detects the named-entities correctly present in the text, but also estimates the probable named-entities from erroneous text. In brief, this algorithm involves a linear sweep of the suffixes of the text. Next, for each of the suffixes a Breadth First Search (BFS) is performed on a Trie data structure. Also to account for error in text, we modify the text and use the notion of edit distance. In order to ensure that the same computation is not repeated again and again, we apply dynamic programming. These steps extract the named-entities from a text.

To evaluate this technique a very robust and optimized implementation of the algorithm was made, and intensive experimentation was performed. We extracted unstructured textual data from Bengali newspapers and observed the outcomes. We also tested the technique on structured data by creating a chatbot to run structured queries. The technique was successful in both the cases.

This paper makes the following contributions:

- An efficient technique has been proposed based on dynamic programming to solve the problem of NER in a closed domain.
- An optimized implementation of this technique, which is hosted in github. Moreover Intensive experimental analysis was done to perceive the correctness and effectiveness of the proposed approach.
- The technique can be used for purposes other than NER. Since this is a generalized partial string matching algorithm, we can apply this technique in solving many other problems.

The rest of the paper has been organized as follows. Section II discusses significant works on the field of NER. Section III presents the proposed approach that recognizes named entities based on dynamic programming. Section IV does an intensive analysis of the correctness of the algorithm and compares it with other alternatives. Section V describes an experimental analysis of the proposed approach. Section VI points out some other possible applications. Section VII concludes the whole work.

II. Related Work

The problem of Named Entity Recognition has been studied for more than two decades. Several techniques have been proposed during this long period of time. However, most of the works in this field require structured data to work with. Some techniques originate from simple handcrafted rules while the recent techniques employ the power of machine learning. The most significant works on this domain have been described below.

One of the very first works of this field was conducted by Rau et al. [6]. This paper gave a complete description of an algorithm that extracts company names from financial news. This algorithm also succeeded in recognizing subsequent references to a company. The algorithm was a combination of clever heuristics, exception lists and extensive corpus analysis. The algorithm generates the most likely variations that those names may go by, for use in subsequent retrieval. The system had extracted thousands of company names with over 95% accuracy and succeeded in extracting 25% more companies that were indexed by a human.

Earlier research rate was very low. It accelerated in 1996, with the first major event dedicated to the task: MUC-6 [1]. From then this problem became more and more popular among researchers and paved the way to numerous scientific events: HUB-4, MUC-7 and MET-2, IREX [7], CONLL [2], ACE [9] and HAREM [10].

A number of researches were devoted to diverse domains. D. Maynard et al. [11] designed a system for emails, scientific texts and religious texts. E. Minkov et al. [12] created a system that was specifically designed for email documents. These experiments demonstrated that NER problem can easily be solved in any domain. However, transferring such a system to a new domain remains a major challenge. T. Poibeau and Kosseim [13] experimented with some systems on both the MUC-6 collection composed of newswire texts, and on a corpus made of manual translations of phone conversations and technical emails. A drop in performance was observed for every system (about 20% to 40% of precision and recall).

The currently dominant technique for solving the NER problem is supervised learning. This includes use of hidden markov models (HMM), support vector machines (SVM), conditional random fields (CRF), decision trees, artificial neural networks (ANN) etc. Some good works are done by M. Asahara et al. [13], A. McCallum et al. [15] etc. For supervised learning, the precision of recognition is about 76% and the recall is about 48%.

Zhou et. al. [16] proposed a Hidden Markov Model (HMM) based chunk tagger, from which a named entity recognition system was constructed to identify and classify names, times and numerical quantities. Through the HMM, the system was built upon four types of features - simple deterministic internal feature of the words, internal semantic feature of important triggers, internal gazetteer feature and external macro context feature. In this way, the NER problem could be resolved effectively. Evaluation of the system on MUC-6 and MUC-7 English NE tasks achieved F-measures of 96.6% and 94.1% respectively. Surprisingly this was much better than machine-learning systems.

Semi Supervised Learning (SSL) is a recent but fairly new significant idea. The technique is mainly bootstrapping and involves a small degree of supervision, i.e. a set of seeds, for the learning process. M. Pasca et al. [17] used techniques inspired by mutual bootstrapping. However, they innovated through the use of D. Lin's (1998) distributional similarity to generate words from the same semantic class to allow pattern generalization. One of the contributions of this paper is to apply the technique to very large corpora (100 million web documents). The authors demonstrated that starting from a seed of 10 examples facts it is possible to generate one million facts with a precision of about 88%. Another problem related to Named Entity Recognition can be name ambiguity; named entities in the web are highly ambiguous. For example, query containing an ambiguous name will return web pages that belong to all of the persons sharing that name. Selvaperumal et al. [19] presented a semi-supervised approach for solving the problem.

Unsupervised learning generally involve clustering, for example, Y. Shinyama and Sekine [18] observed and used the idea that named entities often appear coherently in several articles, whereas common nouns do not. They found a strong correlation between a word, being a named-entity and appearing punctually and simultaneously in multiple news sources. This technique allows identifying rare named entities in an unsupervised manner.

Chaudhuri et al. [5] is one of the few papers that experimented with named-entity detection in Bengali. This paper proposed a three-stage approach of named-entity detection. The stages are based on the use of Named-Entity (NE) dictionary, rules for named-entity and left-right co-occurrence statistics. Experimental results obtained on Anandabazar Patrika (Most popular Indian Bangla newspaper) corpus were quite encouraging.

Some recent groundbreaking works related to Named Entity Recognition are - Santos et al. [20] used neural character embedding to improve NER performance, Chiu et al. [21] presented a neural network architecture that automatically extracts the features using a hybrid bidirectional LSTM and CNN architecture and got 91.62 F1 score, Lample et al. [22] introduced two new neural network architectures - one consisting BLSTM and CRF and the other follows a transition-based approach inspired by shift-reduce parsers. Yang et al. [23] used a deep RNN to extract both morphology and context information on both character and word levels. Also [24] used a combination of BLSTM, CNN and CRF.

Also, there have been some recent works related to Named Entity Recognition for Bengali and other Asian languages. Maha et al. [25] presented an ontology based-approach for semantic annotation for Arabic language. Kale et al. [26] discussed various approaches for Named Entity Recognition for Indian languages and performed a comparison among them. Syeful et al. [27] explored some methods to identify proper nouns from Bengali language, which can be treated as a rudimentary version of Named Entity Recognition. Some works on trie based approximate matching can be found in [28] , [29] and [30].

III. Proposed Approach

In this paper, a partial string matching technique is proposed which augments a Breadth First Search (BFS) on a Trie data structure with dynamic programming. This process comprises four steps which are - Dictionary Construction, Linear Sweep, Dynamic Programming and Credibility Estimation. All these steps are described below.

A. Dictionary Construction

In this paper, since we are working in a closed domain, we can exploit the flexibility of restricting our dictionary to contain only the named entities.

So our dictionary will be limited only to store the named entities. Moreover our dictionary should support three operations.

i) Insert
ii) Search
iii) Delete

To conduct these three operations efficiently, this technique uses a Trie or Radix Tree data structure which allows all these three operations in O(input length) time.

So, a Trie data structure is constructed and the named entities are inserted into it. Every node of the Trie has some attributes like node id, node value (i.e. the character it represents), end-mark (i.e. does it completes a word or not), array of pointers to children etc.

B. Linear Sweep

For the next stage, we need to consider a substring from the text corpus and then identify the presence of a named entity. To do this in lieu of trying out all the N^2 substrings where, N = length of text, this technique does something clever. This technique simply takes all the suffixes, where a suffix S_i is a substring starting at index i and ending in the last index N. A linear sweep of I is performed and all the possible suffixes of the text are taken to continue the computation.

C. Dynamic Programming

After obtaining a suffix string, we need to check for the occurrence of a named-entity in the substring. This technique does this by performing a BFS search on the Trie structure, augmented with Dynamic Programming.

This follows just the usual BFS search strategy, we start from the root, check all the possible paths and then extend our position to the next level. BFS algorithm usually uses a queue data structure to maintain the flow and this ensures a shortest path. But in our case we would see in the following paragraphs that the distances from one node to the next may be either 0 or 1. For this we will use a technique of 0-1 BFS and use a Deque (doubly ended queue) data structure.

But, to control the transitions, this technique uses a dynamic programming approach to ensure that same path is not visited multiple times. This dynamic programming model has 2 states:

i) Current node: a node of the Trie, which has attributes like value, children and end-mark.

ii) Current index: the index of the substring that is being matched.

This technique also uses the idea of edit distance to restrict the propagation of the dynamic programming algorithm. In each step, we are on a particular node of the trie and can do one of the followings.

i) Match: If the currently indexed character of the substring matches with the character of the Trie node then we have found a match. So, we will go down that path, the index will be incremented but the edit distance will remain the same. For example, let the substring be “কলম” “কলম” and let index be 1 i.e. it points to ‘ল’ (for 0-indexed string). Now if we are in a Trie node which represents the character ‘ল’, we will find a match.

ii) Insert: If the currently indexed character of the substring does not match with the character of the trie node, we may need to insert the correct character in the substring, this will increment the edit distance by 1 and the index will be also increased, since we have inserted the correct character, so we will point to the next one. For example, let the substring be “কলম” and let the index be 1 i.e. it points to ‘ল’. Now if we are in a Trie node which represents the character ‘ম’, we will find a mismatch. So, we can insert a ‘ল’ in this position and move on to the next index.

iii) Delete: Again, if the currently indexed character of the substring does not match with the character of the Trie node, we can also delete that character in the substring, this will increment the edit distance by 1 but the index will remain unchanged, since we have not found the correct character.

iv) Skip: Also if we find a mismatch, we can skip the index and move on, this will increment both the index and the edit distance. This case may seem unnecessary and redundant, but for cases when the first character is missing this case will be required otherwise we will fail to find a match.

The transitions can be illustrated in Fig. 1:

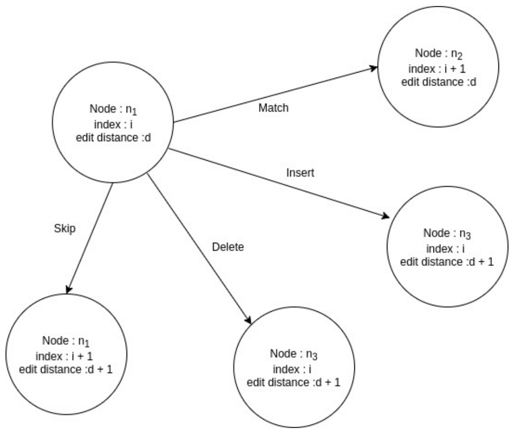


Fig.1. Dynamic Programming Transitions

Now if we find an end-mark on the current node we need to keep a record of that. Because, two scenarios may arise:

i) It is indeed the end of the word.

ii) It is merely a prefix of the word.

So, we will need a data structure where these records will be stored. This technique stores the edit distance and the node id of the Trie, because the string can be obtained by backtracking from that node.

Now it is possible that the correct string is already present in the data structure but with a higher edit distance, in such cases this technique will update that string with the smaller edit distance.

This method will be given a constraint on edit distance. Whenever that constraint is violated, the computation will be stopped.

After each computation of a state, the state is marked in a memo type data structure to ensure that the algorithm is not re-computing the same state again (overlapping sub-problems). As previously mentioned, the states are represented by Trie node id and index, this is enough to uniquely represent a state which will be proved in the next section.

This technique uses a Deque, for the cases when edit distance stays the same the new state is pushed to the front of the Deque and when the edit distance increases the new state is pushed to the back of the Deque.

D. *Credibility Estimation:*

After the dynamic programming and linear sweeping steps, a collection of probable candidates is obtained. Now, these candidates satisfy the constraint of edit distance. However, this may not be sufficient. To ensure the robustness further we used another metric. Simply put, this is the percentage accuracy which can be defined as:

$$\begin{matrix}\text{Percentage}\\\text{Accuracy}\end{matrix} = \frac{\text{TotalLength} - \text{EditDistance}}{\text{TotalLength}} \times 100\ \%$$

The reason for this metric is that, often we would find cases where some strings length are smaller than the edit distance of the desired string. So, in such cases, the algorithm may wrongfully output those smaller length strings since their edit distance is still smaller. To avoid this problem, this metric is used.

We can present the pseudo-code for this algorithm as follows:

```
Method constructDictionary ( data ) :

    Global trie = new Trie()
    for each string in data :

        trie.insert(string)

END


Method linearSweep ( input ) :

    for i = 1 : length(input):

        substring = string [ i : length(input) ]

        DP (substring , editDistanceConstraint )

END


Method DP ( substring, editDistanceConstraint ) :

    Global Candidates, trie

    deque = new Deque()
    // deque element = (nodeId, index, edit distance)

    deque.pushBack ( new state(trie.root,0,0) )

    while ( deque.notEmpty() ):

        state = deque.popFront()

        nodeId = state.nodeId
        index = state.inex
        edit = state.edit

        if ( state.endMark == true ):

            Candidates.push( new(nodeId,edit) )

        if ( memo[nodeId][index] == done ):

            continue

        // skip
        if ( editDistanceConstraint not violated):
            deque.pushBack(
new state(child,index+1,edit+1))

        for child in trie[nodeId].children:

            if(trie[child].value ==substring[index]):
                // match
                deque.pushFront(
            new state(child,index+1,edit))

            else if ( editDistanceConstraint not violated):

            // insert
                deque.pushBack(
            new state(child,index+1,edit+1) )

            // delete
                deque.pushBack(
            new state(child,index,edit+1) )

        memo[nodeId][index] = done

END


Method credibilityEstimation () :

    Global Candidates, trie

    for each candidate in Candidates :

        string = trie.backtrack(candidate.nodeId)

        percentageAccuracy =
    ((length(string) – candidate.edit)/length(string)) *100

    return strings in descending order of percentageAccuracy

END
```

IV. Analysis, Correctness and Comparison

A major issue of dynamic programming is determining the state variables. This technique uses trie node id and index of the substring. However, it would be seen that we also need the edit distance as a state variable. This is redundant as we are traversing the trie following a breadth first search (BFS) algorithm. So it is already ensured that, we will be traversing only in shortest path, so it is completely unnecessary to include edit distance as a state variable.

Next we have two types of transitions, when we have found a match we should proceed without penalty thus the edit distance should be the same, on the contrary in other cases we need to penalize the transition. So, we cannot use a queue for this algorithm, rather we must use a double ended queue (deque). When we need to penalize the transition we would push it to back, otherwise we would push to the front. This will ensure the shortest path condition.

Now if we processed the whole text at once then we would not be able to obtain all the possible named entities. For example, let us assume that an entity starts from index 15, surely we can skip the first 15 characters and then start matching. But unfortunately the 15 skipped characters would have been added to edit distance, so even if the entity may have been present correctly it would be highly penalized.

To overcome this limitation this technique takes an elegant approach. It considers all the suffixes of the input text. This will allow us to start our computation from any desired index. Prefix matching of suffixes essentially leads to complete search.

Now, the asymptotic time complexity of the dynamic programming method is for the worst case O(M*E), where M = total number of Trie nodes and E = maximum edit distance considered. For a Trie containing a huge dictionary the number of nodes may seem to be high, but it is not. Since a lot of prefix overlapping takes place in reality. So, ultimately we are left with a reduced number of nodes. Still this number can be very high for a huge dictionary. So, the dynamic programming method should require a great amount of time. But this does not happen because in reality not all the nodes are visited. Actually only a small portion of the Trie is traversed. This became more apparent in experimental analysis. Although we have a huge theoretical upper bound time complexity, it almost never takes place.

Again a complete linear sweep ensures that we will be able to detect an entity with the least edit distance possible. For example, if an entity starts from index 5 and it includes 2 errors, then when we take the suffix starting from index 3 we will have an edit distance of 4 , again suffix starting from index 4 results in an edit distance of 3 . Finally a suffix starting from index 5 will have only edit distance of 2. After this the edit distance will increase gradually, i.e. 3 at suffix starting at index 6, 4 for suffix starting at index 7 and so on. So this ensures that we will always be able to detect the entities at their least possible edit distance. No entity will be missed or be treated unjustly. This algorithm will always produce correct output.

Now we have two parameters in this method:

i) Edit distance tolerance
ii) Percentage accuracy tolerance

We can tune these two parameters. If we increase edit distance tolerance then the algorithm will take more time to compute. On the other hand, if we reduce the edit distance tolerance too much some entities that are afflicted with error may be missed. Similarly for percentage accuracy tolerance, when we increase it, the search becomes stricter and we may miss some entities. On the contrary reducing it too much may result in detecting some entities that are not actually present.

Now it may seem that our algorithm takes a lot of time and thus inefficient. However, if we compare this approach to other string processing approaches, we would see the opposite. Indeed our approach is highly efficient.

For the sake of simplicity let us first consider that all the entities are presented correctly. In that case we can simply try out all the substrings and then look for matches in them. If the length of the input string was N and there were M entities it would take O(N^3. M) time if we had used efficient string matching algorithms like KMP algorithm, Z algorithm, Robin Carp algorithm etc. Since there are N^2 substrings in total and they can be of length N at most, we can estimate an upper bound of O(N) for checking each substring and as number of entities are M, so for checking all the entities this time complexity becomes O(N . M). Thus for all substrings the overall time complexity becomes O(N^3. M).

However we can improve this, as we can just employ these string matching algorithms on the entire string at once, we need not consider all the N^2 substrings. This reduces the time complexity to O(N . M).

But, this approach will fail if the entities are not represented properly, there may be some error associated with them. In that case we would have to consider all the substrings and calculate edit distance with each of the entities. Now the number of entities is M , total length of text is N, thus the total number of substrings is N^2. And each calculation of edit distance, using efficient dynamic programming will need at most O(N^2) time, since the substring can be of at most length N. So the overall time complexity for calculating edit distances of the substrings with all the entities will become O(N^2 . M . N^2) thus O(N^4. M) . This is really time consuming.

On the other hand our proposed algorithm needs to consider only N suffixes in lieu of N^2 substrings, again each call to dynamic programming method has a worst case time complexity of O(T . E) , where T = number of Trie nodes and E = the maximum edit distance we will consider. So the total time complexity is O(N . T . E). Now total Trie nodes T can be at most the sum of lengths of all the entities, but in practice this stays much smaller, which was observed from experiments on real data. Also our algorithm hardly ever visited all the Trie nodes, because of the properties of natural languages, they are not just random strings of characters, rather they have well defined grammar and well established vocabulary. So we can be assured that our algorithm will hardly ever require the theoretical O(N . T . E) time complexity. Thus the average case and even the practical worst case time complexity will be much smaller than the upper bound O(N.T.E).

However this algorithm is not perfect like other Natural Language Processing algorithms. We would need to give as input the formats of the named entities. This is the only limitation. Our algorithm stays at a position between hand-crafted algorithms and state of the art machine-learning algorithms.

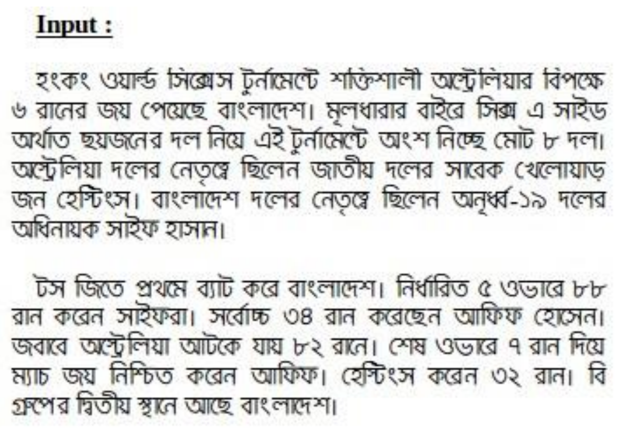
V. Experimental Setup and Result Analysis

In order to evaluate the proposed approach, we underwent a number of experiments and observed the outcomes. We created a very efficient implementation of the algorithm in php language, so that we can use this for web based applications. Hopefully in the future this algorithm will also be implemented in other popular languages e.g. python, java etc. The source code can be found in the following github repository

https://github.com/robin-0/Partial-String-Matching

We experimented on our algorithm using some texts in Bengali. Unfortunately, we failed to find any proper dataset for the task. So we extracted some texts from Bengali newspapers, mostly from the Daily Prothom Alo, most popular newspaper of Bangladesh.

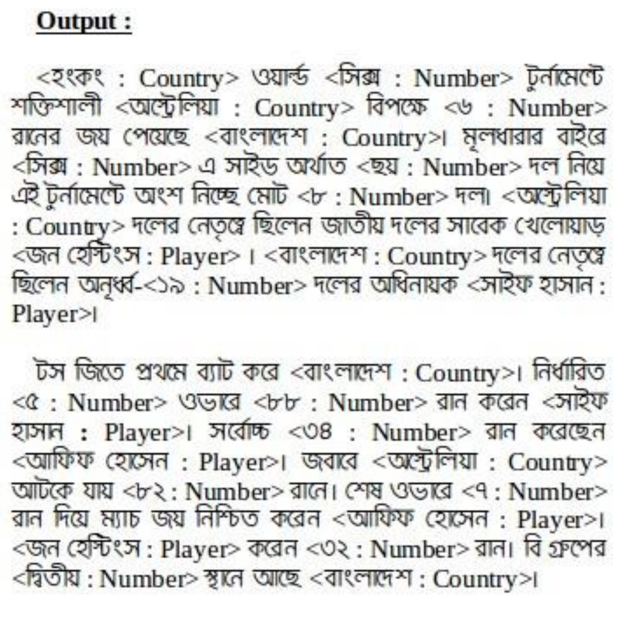
To honor the recent achievements of Bangladesh Cricket Team, we experimented our algorithm on sports related news articles. We trained our model with (i.e. provided named-entity data for) country names, player names, numbers etc. The outcomes of the experiments were very satisfactory as our algorithm managed to identify all the named-entities, present on the text. Outcome of one of the experiments the tests is presented below:



Input :

হংকং ওয়ার্ল্ড সিক্সেস টুর্নামেন্টে শক্তিশালী অস্ট্রেলিয়ার বিপক্ষে ৬ রানের জয় পেয়েছে বাংলাদেশ। মূলধারার বাইরে সিক্স এ সাইড অর্থাত ছয়জনের দল নিয়ে এই টুর্নামেন্টে অংশ নিচ্ছে মোট ৮ দল। অস্ট্রেলিয়া দলের নেতৃত্বে ছিলেন জাতীয় দলের সাবেক খেলোয়াড় জন হেস্টিংস। বাংলাদেশ দলের নেতৃত্বে ছিলেন অনূর্ধ্ব-১৯ দলের অধিনায়ক সাইফ হাসান।

টস জিতে প্রথমে ব্যাট করে বাংলাদেশ। নির্ধারিত ৫ ওভারে ৮৮ রান করেন সাইফরা। সর্বোচ্চ ৩৪ রান করেছেন আফিফ হোসেন। জবাবে অস্ট্রেলিয়া আটকে যায় ৮২ রানে। শেষ ওভারে ৭ রান দিয়ে ম্যাচ জয় নিশ্চিত করেন আফিফ। হেস্টিংস করেন ৩২ রান। বি গ্রুপের দ্বিতীয় স্থানে আছে বাংলাদেশ।



Output :

<হংকং : Country> ওয়ার্ল্ড <সিক্স : Number> টুর্নামেন্টে শক্তিশালী <অস্ট্রেলিয়া : Country> বিপক্ষে <৬ : Number> রানের জয় পেয়েছে <বাংলাদেশ : Country>। মূলধারার বাইরে <সিক্স : Number> এ সাইড অর্থাত <ছয় : Number> দল নিয়ে এই টুর্নামেন্টে অংশ নিচ্ছে মোট <৮ : Number> দল। <অস্ট্রেলিয়া : Country> দলের নেতৃত্বে ছিলেন জাতীয় দলের সাবেক খেলোয়াড় <জন হেস্টিংস : Player> । <বাংলাদেশ : Country> দলের নেতৃত্বে ছিলেন অনূর্ধ্ব-<১৯ : Number> দলের অধিনায়ক <সাইফ হাসান : Player>।

টস জিতে প্রথমে ব্যাট করে <বাংলাদেশ : Country>। নির্ধারিত <৫ : Number> ওভারে <৮৮ : Number> রান করেন <সাইফ হাসান : Player>। সর্বোচ্চ <৩৪ : Number> রান করেছেন <আফিফ হোসেন : Player>। জবাবে <অস্ট্রেলিয়া : Country> আটকে যায় <৮২ : Number> রানে। শেষ ওভারে <৭ : Number> রান দিয়ে ম্যাচ জয় নিশ্চিত করেন <আফিফ হোসেন : Player>। <জন হেস্টিংস : Player> করেন <৩২ : Number> রান। বি গ্রুপের <দ্বিতীয় : Number> স্থানে আছে <বাংলাদেশ : Country>।

As we can see the algorithm manages to identify all the country names, player names and numerical identities. Another strength of this algorithm is since we operate of prefixes we do not need to do any stemming explicitly, the algorithm itself inherently does the stemming. This is why our algorithm manages to identify ‘অস্ট্রেলিয়া’ from the word ‘অস্ট্রেলিয়ার’. Also we can note that, due to our consideration of edit distance and linear sweeping the algorithm is capable of extracting named-entities even if some prefix or suffix or even intermediate characters are missing. As we can see that we were able to identify ‘আফিফ হোসেন’ even when only ‘আফিফ’ was given, and we managed to detect ‘জন হেস্টিংস’ whereas only ‘হেস্টিংস’ was given.

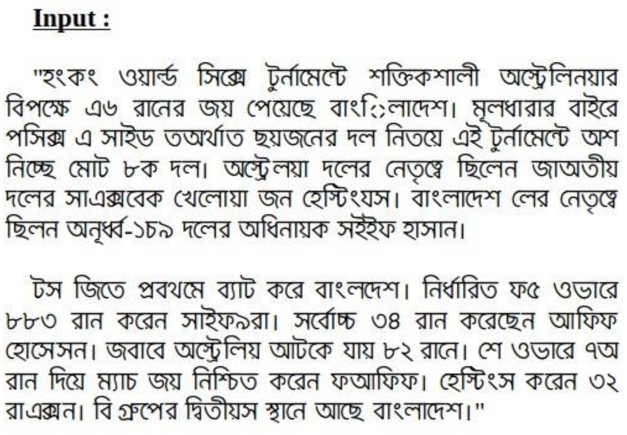
And in order to address our actual practical time complexity rather than theoretical upper bound, let us define the percent average number of visited states as:

$$\text{Average Number ofVisited States}(\%) \\ = \frac{\text{Total Number of Visited States}}{\text{Total Suffixes} \times \text{Total States}} \times 100\%$$

This metric represents how much time our algorithm actually needs compared to the theoretical upper bound of O(NTE). For this particular example, we found this value to be only 3.8336%. So, actually our algorithm is needing only 3.8336% of O(NTE), where N= length of string, T = total Trie nodes, E = edit distance limit. This is surprisingly small. We found that in all our experiments this metric never exceeded 10% and most of the time it is expected to be around 5%. The inherent properties of the natural languages ensure that words are not merely a random permutation of characters, rather they follow certain established rules, this property significantly reduces our actual computation time.

Now, to make our task more challenging, we then added some noise to the text, i.e. we randomly deleted some characters and randomly inserted some characters. We had the ability to change the amount of noise. It was seen that as long as the noise was within the limit of our edit distance constraint, our algorithm successfully managed to identify all the named entities. However if we add more noise compared to our edit distance tolerance, the system starts to fail, which is expected. This added noise can be taken into account by relaxing the edit distance tolerance, but in our real-world the texts usually do not contain that much of noise, so our algorithm is practical.

The same example with random noise becomes as follows:

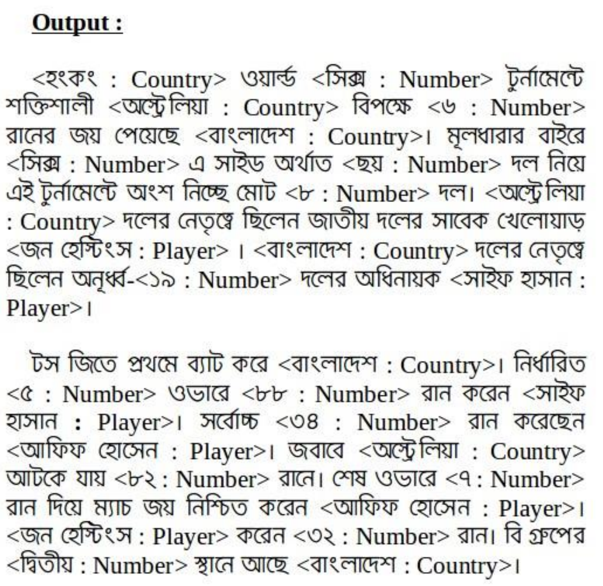


Input :

"হংকং ওয়ার্ল্ড সিক্সে টুর্নামেন্টে শক্তিকশালী অস্ট্রেলিনয়ার বিপক্ষে এ৬ রানের জয় পেয়েছে বাংি্লাদেশ। মূলধারার বাইরে পসিক্স এ সাইড তঅর্থাত ছয়জনের দল নিতয়ে এই টুর্নামেন্টে অশ নিচ্ছে মোট ৮ক দল। অস্ট্রেলয়া দলের নেতৃত্বে ছিলেন জাঅতীয় দলের সাএক্সবেক খেলোয়া জন হেস্টিংযস। বাংলাদেশ লের নেতৃত্বে ছিলন অনূর্ধ্ব-১চ৯ দলের অধিনায়ক সইইফ হাসান।

টস জিতে প্রবথমে ব্যাট করে বাংলদেশ। নির্ধারিত ফ৫ ওভারে ৮৮৩ রান করেন সাইফ৯রা। সর্বোচ্চ ৩৪ রান করেছেন আফিফ হোসেসন। জবাবে অস্ট্রেলিয় আটকে যায় ৮২ রানে। শে ওভারে ৭অ রান দিয়ে ম্যাচ জয় নিশ্চিত করেন ফআফিফ। হেস্টিংস করেন ৩২ রাএক্সন। বি গ্রুপের দ্বিতীয়স স্থানে আছে বাংলাদেশ।"

And the output is:



Output :

<হংকং : Country> ওয়ার্ল্ড <সিক্স : Number> টুর্নামেন্টে শক্তিশালী <অস্ট্রেলিয়া : Country> বিপক্ষে <৬ : Number> রানের জয় পেয়েছে <বাংলাদেশ : Country>। মূলধারার বাইরে <সিক্স : Number> এ সাইড অর্থাত <ছয় : Number> দল নিয়ে এই টুর্নামেন্টে অংশ নিচ্ছে মোট <৮ : Number> দল। <অস্ট্রেলিয়া : Country> দলের নেতৃত্বে ছিলেন জাতীয় দলের সাবেক খেলোয়াড় <জন হেস্টিংস : Player> । <বাংলাদেশ : Country> দলের নেতৃত্বে ছিলেন অনূর্ধ্ব-<১৯ : Number> দলের অধিনায়ক <সাইফ হাসান : Player>।

টস জিতে প্রথমে ব্যাট করে <বাংলাদেশ : Country>। নির্ধারিত <৫ : Number> ওভারে <৮৮ : Number> রান করেন <সাইফ হাসান : Player>। সর্বোচ্চ <৩৪ : Number> রান করেছেন <আফিফ হোসেন : Player>। জবাবে <অস্ট্রেলিয়া : Country> আটকে যায় <৮২ : Number> রানে। শেষ ওভারে <৭ : Number> রান দিয়ে ম্যাচ জয় নিশ্চিত করেন <আফিফ হোসেন : Player>। <জন হেস্টিংস : Player> করেন <৩২ : Number> রান। বি গ্রুপের <দ্বিতীয় : Number> স্থানে আছে <বাংলাদেশ : Country>।

We can see that, though we added some noise by randomly inserting and removing characters, our algorithm still manages to identify the named-entities. For examples it succeeded to identify ‘অস্ট্রেলিয়া’ from ‘অস্ট্রেলিনয়া’ , '১৯' from '১৮৯' and so on. Also this time the percent average number of visited states is only 3.676%.

Table 1. Result Analysis

Dataset	Edit Distance 0	Edit Distance 1	Edit Distance 2
Dataset 1	**100.00 %**	**100.00 %**	**100.00 %**
Noisy Dataset 1 σ = 1.5	66.67 %	83.33 %	83.33 %
Noisy Dataset 1 σ = 2.0	50.00 %	66.67 %	66.67 %
Noisy Dataset 1 σ = 2.5	33.33 %	66.67 %	66.67 %
Dataset 2	**60.00 %**	**60.00 %**	**60.00 %**
Noisy Dataset 2 σ = 1.5	40.00 %	40.00%	60.00 %
Noisy Dataset 2 σ = 2.0	30.00 %	30.00 %	30.00%
Noisy Dataset 2 σ = 2.5	20.00 %	30.00 %	30.00 %
Dataset 3	**81.81 %**	**81.81 %**	**81.81 %**
Noisy Dataset 3 σ = 1.5	63.60 %	72.72 %	81.81 %
Noisy Dataset 3 σ = 2.0	54.54 %	54.54 %	72.72 %
Noisy Dataset 3 σ = 2.5	27.27 %	36.36 %	54.54 %

We performed similar experiments on several other datasets and recorded the outcomes. We also added noises and observed that accuracy declines. However, by tuning the constraints we can further improve the accuracy. Some of the results are shown in Table 1.

These datasets and several others can be found on the github repository **https://github.com/robin-0/Partial-String-Matching** . Actually when we add noise, the accuracy declines because then the text does not remain grammatically correct anymore, as we randomly deleted and randomly inserted random characters. Hopefully in practice we are very unlikely to face such cases. Thus we can conclude that our technique works satisfactory in practice.

These experiments were for unstructured data. We also did some experimenting on structured data by creating a chatbot which will receive some query from the user and look for different named-entities in the text and provide information based on that. We also obtained successful outcomes from these experiments. We trained the chatbot with car brand names, model numbers, etc. and it managed to detect them. For example:

User Query:

```
>> How much will Tesla's model X cost in Canada?
```

Bot perceived:

```
< Tesla : Car Brand>
< Model X : Car Model>
< Canada : Country>
```

Thus the bot was able to extract the named-entities from the user's query.

Thus our algorithm is proved work very well for both structured and unstructured data.

VI. Applications

This algorithm was developed to recognize named-entities from unstructured text, more precisely Bengali text. We developed this algorithm with the sole purpose of named entity recognition. However this same algorithm can be used to extract information from texts, or even can be used as a spell-checker. In the later case the dictionary should contain all the words. And as seen from the experiments this algorithm can also be used to construct IR powered chatbots.

VII. Conclusion

Named Entity Recognition is a very important problem in the fields of NLP and IR. In today’s age of information technology a big challenge is to parse texts to extract information from them. This is the preprocessing step for a lot of digital services today. But unfortunately very little work has been done for Named Entity Recognition for Bengali language.

In this paper, a technique is proposed for named entity recognition (NER) task in Bengali language within a closed domain using partial string matching. This technique is based on dynamic programming. We take all the suffixes of a string. Then we perform a BFS search on a Trie data structure. In order to consider errors in text we use the notion of edit distance, and to prevent the same computation taking place again and again we employ dynamic programming.

This technique was discussed in a very detailed manner using pseudo-codes. We analyzed the algorithm and proved its correctness. Moreover we have created an optimized implementation of the algorithm and experimented with it thoroughly. The experiments were successful and the algorithm performed as expected. Experiments were done for both structured and unstructured data. However we can prune the algorithm by tuning the parameters, it draws a balance between efficiency and accuracy. Since we don't have adequate data for Bengali to train a machine-learning model, this partial string matching approach is the best approach we can follow currently. Our technique stands in between hand-crafted algorithms and state of the art machine-learning algorithms.

References

[1] D. Nadeau, and S. Sekine, “A survey of named entity recognition and classification,” Lingvisticae Investigationes, vol. 30, no. 1, 2007, pp. 3–26.

[2] T. K. Sang, F. Erik, and F. De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,” in Proceedings of the Seventh Conference on Natural Language Learning, Association for Computational Linguistics, 2003.

[3] T. K. Sang, F. Erik, and F. De Meulder, “Introduction to the CoNLL-2002 shared task: language-independent named entity recognition,” in Proceedings of the 6th Conference on Natural Language Learning, Aug 2002, pp. 1–4.

[4] R. Grishman, and B. Sundheim, “Message understanding conference-6: A brief history,” in Proceedings of the 16th International Conference on Computational Linguistics, 1996.

[5] B. B. Chaudhuri, and S. Bhattacharya, “An Experiment on Automatic Detection of Named Entities in Bangla,” *IJCNLP*, pp.75–82, 2008.

[6] L. F. Rau, “Extracting company names from text,” in Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications, IEEE, Feb 1991, pp. 29–32.

[7] S. Sekine, and H. Isahara, “IREX: IR & IE Evaluation Project in Japanese,” in LREC, 2000, pp. 1977–1980.

[8] T. K. Sang, F. Erik, and F. De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,” in Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, 2003.

[9] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, “The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation,” In LREC, vol. 2, pp. 837–840, 2004.

[10] D. Santos, N. Seco, N. Cardoso, and R. Vilela, “Harem: An advanced ner evaluation contest for portuguese.” quot; In Nicoletta Calzolari; Khalid Choukri; Aldo Gangemi; Bente Maegaard; Joseph Mariani; Jan Odjik; Daniel Tapias (ed) in Proceedings of the 5th International Conference on Language Resources and Evaluation, May 2006.

[11] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks, “Named entity recognition from diverse text types,” in Recent Advances in Natural Language Processing 2001 Conference, 2001, pp. 257–274.

[12] E. Minkov, R. C. Wang, and W. W. Cohen, “Extracting personal names from email: Applying named entity recognition to informal text,” in Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Oct 2005, pp. 443–450.

[13] T. Poibeau, and L. Kosseim, “Proper name extraction from non-journalistic texts," *Language and computers, vol.* 37, no.1, pp. 144–157, 2001.

[14] M. Asahara, and Y. Matsumoto, “Japanese named entity extraction with redundant morphological analysis,” in Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Association for Computational Linguistics, May 2003, pp. 8–15.

[15] A. McCallum, and W. Li, “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons,” in Proceedings of the seventh conference on Natural language learning, Association for Computational Linguistics, May 2003, vol. 4, pp. 188–191.

[16] G. Zhou, and J. Su, “Named entity recognition using an HMM-based chunk tagger,” in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, Jul 2002, pp. 473–480.

[17] R. C. Bunescu, and M. Pasca, “Using Encyclopedic Knowledge for Named entity Disambiguation,” Eacl, vol. 6, pp. 9–16, 2006.

[18] Y. Shinyama, and S. Sekine, “Named entity discovery using comparable news articles,” in Proceedings of the 20th International Conference on Computational Linguistics. Association for Computational Linguistics, Aug 2004.

[19] P. Selvaperumal, and A. Suruliandi, “Semi-Supervised Personal Name Disambiguation Technique for the Web,” *International Journal of Modern Education and Computer Science(IJMECS*), vol. 8, no. 3, pp. 28–36, Mar 2016.

[20] C. N. Santos, and V. Guimaraes, “Boosting named entity recognition with neural character embeddings,” arXiv preprint arXiv:1505.05008 (2015).

[21] J. P. Chiu, and E. Nichols, “Named entity recognition with bidirectional LSTM-CNNs,” arXiv preprint arXiv:1511.08308 (2015).

[22] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural architectures for named entity recognition,” arXiv preprint arXiv:1603.01360 (2016).

[23] Z. Yang, R. Salakhutdinov, and W. Cohen, “Multi-task cross-lingual sequence tagging from scratch,” arXiv preprint arXiv:1603.06270 (2016).

[24] X. Ma, and E. Hovy, “End-to-end sequence labeling via bi-directional lstm-cnns-crf,” arXiv preprint arXiv:1603.01354 (2016).

[25] M. Al-Yahya, M. Al-Shaman, N. Al-Otaiby, W. Al-Sultan, A. Al-Zahrani, M. Al-Dalbahie, “Ontology-Based Semantic Annotation of Arabic Language Text,” *IJMECS*, vol. 7, no. 7, pp. 53–59, 2015.

[26] S. Kale, and S. Govilkar, “Survey of Named Entity Recognition Techniques for Various Indian Regional

Languages," International Journal of Computer Applications, vol. 164, no. 4, 2017.
[27] M. S. Islam, and J. K. Das, "Design Analysis Rules to Identify Proper Noun from Bengali Sentence for Universal Networking language", *IJMECS*, vol. 6, no. 8, pp. 1–9, 2014.
[28] Risvik KM "Search system and method for retrieval of data, and the use thereof in a search engine." ,United States Patent 6377945 B1, April 23 2002
[29] Shang H, Merrettal T "Tries for approximate string matching." , IEEE Trans Knowl Data Eng 8(4):540–547
[30] Oommen, B.J. & Badr, G. Pattern Anal Applic (2007) 10: 1. https://doi.org/10.1007/s10044-006-0032-z

Authors' Profiles

Nabil Ibtehaz is a graduate student at the Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET), Bangladesh. Currently, he is pursuing his Master of Science in Computer Science and Engineering (MSc CSE). He earned his Bachelor of Science in Electrical and Electronics Engineering (BSc EEE) from the same institution. His core areas of interest are machine learning, natural language processing, computer vision , signal processing , evolutionary algorithms . He has participated in various national and international competitions.

Abdus Satter is a graduate student at the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. Currently, he is pursuing his Master of Science in Software Engineering (MSSE). He earned his Bachelor of Science in Software Engineering (BSSE) from the same institution with the top score in his class. His core areas of interest are data mining, machine learning, software engineering, web technologies, systems and security. He has numerous awards in various national and international software and programming competitions, hackathons & project showcasings.

How to cite this paper: Nabil Ibtehaz, Abdus Satter, "A Partial String Matching Approach for Named Entity Recognition in Unstructured Bengali Data", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.1, pp. 36-45, 2018.DOI: 10.5815/ijmecs.2018.01.04