Pre-annotation Based Approach for Development of a Sanskrit Named Entity Recognition Dataset

Sujoy Sarkar
IIT Kharagpur
sujoys@iitkgp.ac.in

Amrith Krishna
IT University of Copenhagen
amrk@itu.dk

Pawan Goyal
IIT Kharagpur
pawang@cse.iitkgp.ac.in

Abstract

Recent NLP algorithms for Named Entity Recognition require a large amount of annotated resources. Low resourced languages such as Sanskrit lack adequate annotated data and manual annotation is costly as well as time-consuming. A sequence labeling tool that supports manual annotation with automatic suggestions to reduce time and increase annotators' productivity is appealing, especially for low-resourced languages. Most of the tools developed with such capabilities are for resource-rich western European languages and are not tuned to the need of morphologically rich language like Sanskrit. In our work, we deploy an annotation tool for crowd-sourcing a Sanskrit NER dataset annotation. The selected corpus is *Śrīmad-Bhāgavatam*. The annotation environment is augmented to help the annotators with automatic suggestions. The suggestions are pre-annotated using our proposed heuristic. The heuristic works based on string-matching algorithms. The key idea is to compare transliterated Sanskrit text with an English translation and identify words of similar forms as likely named entities (NE). A gazetteer is applied to further classify the NEs into semantic classes. On a small manually annotated dataset, the heuristic identified the NEs reasonably with an F-Score of 0.80.

1 Introduction

Named entity recognition (NER) is the task of finding spans of text that constitute proper names in unstructured documents, and classify them into predefined semantic categories such as person, location, etc. (Jurafsky and Martin, 2009). It has emerged from the field of Information Retrieval(IR) and has several downstream applications such as open domain question answering (Chen et al., 2017), co-reference resolution (Yang and Mitchell, 2016), improving semantic search (Caputo et al., 2009). One such motivating application is the use of NER to improve the searching and browsing capability of digital libraries (Borin et al., 2007; Caputo et al., 2009). A recent study on The Gallica digital library, a national heritage and encyclopedia library of France shows that 80% of the top 500 queries sent to the portal contained at least one named entity (NE) (Chiron et al., 2017). However, use of keyword-based search mechanisms often limit the effectiveness of search systems for such digital libraries. Words in natural languages tend to express polysemy and synonymy resulting in low precision and recall respectively of such keyword based search systems. An alternative approach is to search documents not only at the lexical level but also at the semantic level. "Named entities (NEs) mentioned in a document constitute an important part of its semantics" (Caputo et al., 2009). Indexing documents with named entities allow users to search the collection in novel ways (Bontcheva et al., 2002). Sanskrit has over 30 million manuscripts produced over a course of four millennia (Goyal et al., 2012). In the last few decades, a prodigious effort is put in to digitize these manuscripts and make them accessible through digital libraries (Krishna et al., 2017). These libraries can be enriched with NE metadata information to improve their functionalities. But to the best of our knowledge, there is no publicly available NE annotated data to facilitate research on Sanskrit NER.

Proceedings of the Computational Sanskrit and Digital Humanities, 18^{th} *World Sanskrit Conference 2023*, pages 59–70

Recent advances in machine learning (ML) have achieved state-of-the-art performance on many sequence labeling tasks, including NER (Liu et al., 2018; Baevski et al., 2019). These ML algorithms rely on the availability of vast volumes of annotated data. However, creation of annotated text data is an expensive and a time-consuming process (Goyal et al., 2012; Yimam et al., 2014; Lin et al., 2019). Further, it puts a heavy demand on human annotators to maintain the quality annotations (Stenetorp et al., 2012). This makes direct application of such difficult for low-resource languages. Machine-assisted human validated semi-automatic annotation is already shown to be effective in reducing the annotation labor for preparing morphologically tagged corpus in Sanskrit (Goyal and Huet, 2016; Goyal et al., 2012). Generic tools are developed for text annotation including the NE annotation task. Most of these tools provide a good interface but does not provide machine-assisted support to human annotators (Bontcheva et al., 2013; Chen and Styler, 2013). More recently, tools are developed to combining automatic pre-annotation and manual annotation to improve the efficiency and quality of annotation (Teng et al., 2019; Yimam et al., 2014). In the pre-annotation approach, the annotation tool automatically suggests sequence labels to human annotators. The automatic suggestion of sequence labels can be static or dynamic. In static suggestion, data is pre-annotated using some rules before the annotation process begins (Lingren et al., 2013). Also, some tools give dynamic suggestions using ML algorithms that gradually learns from previously annotated data as annotation progresses (Teng et al., 2019; Lin et al., 2019). But these tools are developed for resource-rich western European languages and is not tuned to the need of morphologically rich language like Sanskrit. Automatic pre-annotation of NEs in Sanskrit is more challenging due to many challenges presented by the features of the language that varies significantly from typical Western European language features (Goyal et al., 2012). A large number of possible inflections of NEs, difficulty in identifying word boundaries due to phonemic changes and ambiguities between names and other parts-of-speech categories makes the task difficult.

In our work, we propose a heuristic based on string-matching algorithms to automatically pre-annotate Named Entities (NEs) in a Sanskrit corpus. The key idea is to compare transliterated text and English translation and identify words of similar forms as likely named entity. We use a name dictionary or gazetteer to further classify the NEs into three semantic classes: person, location and miscellaneous. The selected corpus is *Śrīmad-Bhāgavatam* in which words are already segmented and word by word English translation is available digitally. We present a web-based NE annotation platform for annotating *Śrīmad-Bhāgavatam*, based on an open-source text annotation framework. The annotation environment helps annotators by providing automatic NE suggestions based on the pre-annotated data. Currently, the scope is limited to static suggestions. Our platform is publicly available for crowd sourcing the task. Annotators can collaborate, modify the pre-annotations as well as add new annotations. An annotation-approver role also can be assigned to approve the annotations. We make the following specific contributions:

1. Our proposed string-matching based pre-annotation heuristic achieves 0.80 F1 in the task of identifying NEs on a set 60 manually annotated verses from *Śrīmad-Bhāgavatam*.

2. In our current experiment with variations of the heuristic we observe that the performances of string-matching algorithms do not significantly vary if inflected words are used instead of stems obtained by lemmatization. Among the string-matching algorithms that we selected, Jaro-Winkler similarity gives the best performance.

3. We deployed a web-based NE annotation platform for crowd-sourcing NE annotation in *Śrīmad-Bhāgavatam*.[0] The annotation environment is augmented to show annotation suggestions to annotators based on the pre-annotations by the heuristic.

[0] `https://sanskritner.herokuapp.com/`

2 Named Entity Recognition Task

Named Entity Recognition is one of the fundamental tasks in NLP and Information Extraction. NER aims to find spans of text that constitute proper names in unstructured documents, and classify them into pre-defined semantic categories (Jurafsky and Martin, 2009). As natural language is polysemous, ambiguity exists among the name references. Resolving ambiguity involves recognizing the true referent entity of a name reference and then resolving the proper type. NER has achieved state of the art for western European languages using machine learning algorithms. NER task is more complex in Sanskrit compare to western European languages. Several challenges are posed by the rich morphological features of the language.

- Phonemic changes (*Sandhi*) obscure the word boundaries in a sentence by combining multiple words, resulting in long and complex word forms. A named entity may also get combined with other words. For example, *'śrībrahmovāca'* translates to *'Śrī Brahmā said'*. Here Brahmā is a named entity that is easily identifiable in English as it is present as a separate word. Also, the capitalization of the NE is a major feature used by NER systems for European languages. Even multiple NEs may be present in a single word. For example, the word *'sakarṇaduḥśāsanasaubalānā '* contains three NEs: *'karṇa, duḥśāsana, saubalānām'*. So to prepare a dataset one needs to segment the sandhied words in a corpus. However, word segmentation in Sanskrit is not a trivial task as the number of possible segments for a given sentence can be very large. Segmentation tools are available in popular computational Sanskrit platforms like Sanskrit Heritage Reader(SHR), but human intervention is required to decide the final solution (Goyal et al., 2012).

- NEs can also be a compound word, for example, *'kumbhakarṇa'*. Compounds are extensively used in Sanskrit literature but processing compound is a more complex task than *Sandhi* (Kumar et al., 2010). A word segmentation tool may consider the word *'kumbhakarṇa'* as two words making it difficult to identify it as a name.

- Another problem is there are up to 72 possible inflections of NEs due to gender, number and cases. Morphological analysis needs to be done to identify the stems (Krishna, 2019).

- In Indian languages names of persons, places, etc are more diverse and a lot of them can be found in the dictionary as common nouns. Also, ambiguities between names and adjectives, verbs and other parts-of-speech categories are prevalent. For example, *'vidūra'* is the name of a person which also means very far.

As our selected corpus is already in segmented form, we did not need to handle the problem of segmentation and compound word.

2.1 NER Tag-set

The task of NE expression identification in text was first defined in MUC-6 Named Entity task (Sundheim, 1995) where a total of seven NE tags were classified into three classes: ENAMEX (person, location and organization), TIMEX (date, time) and NUMEX (money, percentage). Only four categories of NEs are identified in the shared task 6 of CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003): person, location, organization and miscellaneous. Text Encoding and Interchange guidelines (Burnard and Bauman, 2008) that maintain the XML standard for encoding and documenting have proposed hierarchical NEs. As our annotated data is meant for general-use linguistic resources, we have opted for CoNLL 2003 convention with three NE categories: person, location, and miscellaneous. *Bhāgavatam* is a spiritual text and lacks organizational entities, so we did not explicitly use the organization category.

3 Data

3.1 Corpus

Our approach of pre-annotation is based on a comparison of IAST (Williams et al., 1899) transliteration of Sanskrit text and the corresponding English translation. Many Sanskrit works of literature are translated into English over time. We have selected *Bhāgavatam* to demonstrate our approach and create an NER dataset. It can be extended to other Sanskrit texts with English translation as well. There are two reasons for selecting *Bhāgavatam*. The first reason is that word by word translation is done and made available online. The word by word translation makes it easier to apply our approach. We will discuss this further in Section 4. The second reason is that in the transliterated text words are already present in segmented form which makes it easier to identify the NEs. Also compared to other translated texts, it has a reasonable amount of text. *Bhāgavatam* contains a total of 18K verses with 0.23 million words. CoNLL 2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003), which is one of the standard NER English data set, has 0.3 million words.

We have scrapped the total data of *Bhāgavatam* from online resources. In the online version sometimes, multiple verses are merged into a single verse. Total there are 12 cantos, 335 chapters, 13,004 occurrences of single or multiple verses, where each such instance is taken as one example to annotate at a time. The vocabulary contains a total of 41K unique words and inflections.

3.2 Gazetteer

Name dictionaries or gazetteer are very helpful NER resources. Modern names are collected from different sources like a telephone directory and named dictionaries. Identification of names using Gazetteer is difficult due to inflection, ambiguity etc., but it can be used to boost the performance of NER systems (Saha and Majumder, 2018). *Bhāgavatam* is a well discussed literary work. So, it is relatively easier to collect the list of names of characters or places. We have prepared a name dictionary by automatically scrapping names of characters from online resources. We have mainly collected person names, but the list can be extended further by adding names of other entity types. The gazetteer has 611 unique names.

4 System Description

In this work, we have attempted to develop a system that will provide a platform to crowdsource NER annotation. It will also assist the annotator to improve performance and speed by indicating which words are most likely a named entity and, in some cases, suggesting the possible class. As discussed earlier, suggestions can be static, i.e., prepared before the annotation process begins or these can be dynamically learned and improved as annotators annotate more and more data. The scope of our current work is restricted to static pre-annotation. In this section, we will present the details of the pre-annotation approach and also a description of the annotation framework.

4.1 Pre-annotation Approach

We want to compare Sanskrit transliteration and English translation to identify the named entities. The basic idea is that named entities do not change form in any language. So, the similar strings in transliteration and translation are most likely named entities.

For example, in Figure 1 we can see the English translation of the word *'kṛṣṇe'* has the word *'Lord Śrī Kṛṣṇa'*. Here the word *'kṛṣṇe'* and *'Kṛṣṇa'* have similar forms.

The second thing that we need to handle is the inflected forms of Sanskrit words. A word can appear in many different forms due to the morphological richness of the language. For example, one of the inflected forms of the word *'arjuna'* is *'arjunāya'* meaning *'to Arjuna'*.

https://vedabase.io/en/library/sb/
https://vaniquotes.org/wiki/Category:Personalities_from_Srimad_Bhagavatam
https://vedabase.io/en/library/sb/2/8/3/

kathayasva mahābhāga
yathāham akhilātmani
kṛṣṇe niveśya niḥsaṅgaṁ
manas tyakṣye kalevaram

kathayasva — please continue speaking; *mahābhāga* — O greatly fortunate one; *yathā* — as much as; *aham* — I; *akhila-ātmani* — unto the Supreme Soul; *kṛṣṇe* — unto Lord Śrī Kṛṣṇa; *niveśya* — having placed; *niḥsaṅgam* — being freed from material qualities; *manaḥ* — mind; *tyakṣye* — may relinquish; *kalevaram* — body.

Figure 1: Example of a verse (Śrīmad-Bhāgavatam 2.8.3) and word by word English translation.[0]

Here either we can apply approximate string matching directly using the inflected word or can apply morphological analysis to get the stem and then match. We have experimented with both the approaches. In second case, before applying string comparison the stems of the Sanskrit word is obtained using Sanskrit Heritage Reader (Goyal et al., 2012; Goyal and Huet, 2016). For string matching, we have opted for edit-distance based string similarity algorithms. String distance metrics are simple, fast and can be applied without prior knowledge of data. Cohen et al. (2003) have used string distance metrics for name-matching tasks. They have explored edit distance-based and token-based methods for matching named entities and found that Jaro and Jaro-Winkler metrics are better for short strings. In our data, we observed that most of the names are single word entities. So, we have tried only the edit distance-based methods such as Levenshtein distance, Monger-Elkan distance, Jaro similarity, and Jaro-Winkler similarity. Overall, the best-performing method is Jaro-Winkler similarity.

4.1.1 Jaro-Winkler Similarity

In record-linkage literature, good results have been obtained using Jaro-Winkler (Cohen et al., 2003). The similarity between two strings is measured in the range $[0, 1]$ based on the number and order of the common characters. Jaro-Winkler similarity uses Jaro similarity. Given two strings s_1 and s_2, Jaro similarity between these is defined as

$$Jaro(s_1, s_2) = \begin{cases} \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & : m > 0 \\ 0 & : otherwise \end{cases} \quad (1)$$

where $|s_1|, |s_2|$ are the lengths of the two strings, m is the number of matching characters, and t is the number of transpositions. Two characters from s_1, s_2 respectively, are considered matching if the maximum distance between the characters is, $w = \frac{max(|s_1|,|s_2|)}{2}$–1. So a character from s_1 is matching to a character in s_2 if it is similar to any character from s_2 within the range of w charters before and after. The Jaro-Winkler similarity increases the Jaro similarity value if both the strings have common prefixes.

$$JaroWinkler(s_1, s_2) = \begin{cases} Jaro(s_1, s_2) + l \cdot p \cdot (1 - Jaro(s_1, s_2)) & : Jaro(s_1, s_2) \geq b_t \\ Jaro(s_1, s_2) & : otherwise \end{cases} \quad (2)$$

l is the length of the common prefix of both strings up to a maximum l_{bound}, b_t is the boost threshold with default value 0.7, p is the prefix scale with default value 0.1, and $l_{bound} \cdot p \leq 1$ must holds true (Keil, 2019). For example, if we compare the two words *'kṛṣṇe'* and *'kṛṣṇa'* the Jaro value is 0.86, and both have common prefix of length 4. So the Jaro-Winkler value is boosted to 0.92.

4.1.2 Word Selection for Comparison

The annotation heuristic is improved further taking advantage of the fact that in English, named entities always start with a capital letter. For example, the word *'abhimanyunā'* is translated in the context as *'by the hero Abhimanyu'*. So only those Sanskrit words are selected which

have at least one word in translation starting with a capital letter. To reduce the number of comparisons, we compare *'abhimanyunā'* only with *'Abhimanyu'* as it starts with a capital, and avoid comparing to remaining words. An English dictionary is also used to avoid comparison to general English words. For example, *'om'* is translated as *'O my Lord'*. Clearly, *'om'* is not NE but, *'O'* and *'om'* have very high similarity. Here, although *'O'* and *'Lord'* start with a capital, both are present in the English dictionary. So, we do not calculate similarity and 'om' is marked as non-NE.

Finally, as transliterated verses and many words in the English translation also in the scrapped data are in IAST transliteration scheme, before applying string matching strings are converted to ASCII and case folded. The procedure of pre-annotation is presented in Algorithm 1.

4.1.3 Gazetteer Based Classification

Gazetteer based NE identification is not so simple for Indian languages as names can be ambiguous. Many common words are used as names in Indian languages. Ambiguity between nouns and NEs is observed in many languages. But in Indian languages ambiguity occurs between names and adjectives, verbs and other parts-of-speech categories also (Saha and Majumder, 2018). The problem is more prevalent in Sanskrit. For example, *'karṇa'* is name of a character and also translates as ear.

So, we do not use the gazetteer for the identification of NEs. Instead we use it to the classification of already identified NEs by the previous step. The advantage of the dataset is that it provides word by word meaning in the context which makes word sense disambiguation easier. For example, the word *'sakarṇaduḥśāsanasaubalānā '* is present in the dataset in split form as four words: *'saḥ — He (the Lord); karṇa — Karṇa; duḥśāsana — Duḥśāsana; saubalānām — Saubala'*. Another word *'hṛtkarṇarasāyanāḥ'* is present as three words: *'hṛt — to the heart; karṇa — to the ear; rasa-ayanāḥ — pleasing;'*. So, we can easily classify *'karṇa'* in the first example as the name of a person.

Algorithm 1: Preannotation algorithm

```
Result: List of annotated verses
begin
    θ ← Jaro Winkler Similarity Threshold
    for verse in list_of_verses do
        for sanskrit_word in verse do
            english_translation ← GetTranslation(sanskrit_word)
            for word in english_translation do
                if StartsWithCapital(word) = TRUE then
                    if EnglishDictionary(word) = FALSE then
                        d ← JaroWinklerSimilarity(sanskrit_word, word)
                        if d > θ then
                            NE [sanskrit_word] ← TRUE
                            if sanskrit_word in Gazetteer then
                                Label [sanskrit_word] ← GetLabel(sanskrit_word)
```

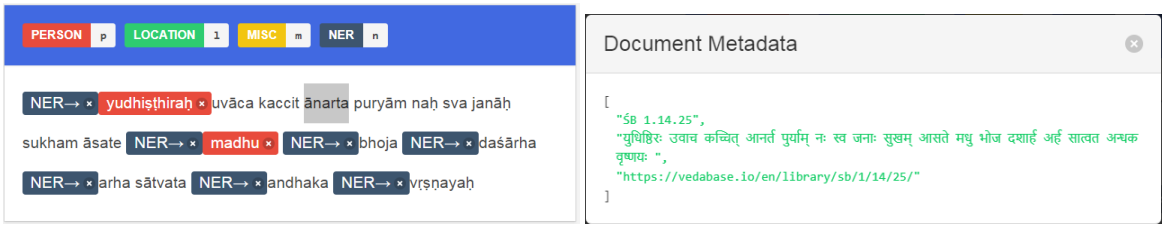
4.2 Annotation Framework

We have deployed a web-based crowd-sourcing annotation tool.[0] The UI design is based on an open-source Django framework named Doccano (Nakayama et al., 2018). The open-source framework provides annotation features for sequence labeling tasks. It can collect annotations from multiple (non-expert) contributors with lower cost and higher speed. It has Unicode support to display Sanskrit text in different encoding.

Figure 2 illustrates the annotation environment and how the tool presents the pre-annotated

suggestions for the verse. The **NER→** tag suggests to annotators that the next word is most likely a named entity. In the figure, a total of seven suggestions are given. In that, the two words *'yudhiṣṭhiraḥ'* and *'madhu'* are further suggested as name of PERSON. If annotators want to remove the suggested tag or to correct the tag, they can simply click on the cross symbol. A new tag can be assigned by selecting the word and then clicking the right type appearing on the top. They can also press the shown shortcut key for quick tagging. For example, *'ānarta'* is the name of a place but it is not pre-annotated. So, an annotator can select that word and press 'l' on the keyboard or click on the LOCATION tag to assign the new tag. Additionally, there is a metadata button. Annotators can use it to read the verse in Devanagari or can use the URL to read about it in detail.

The tool is hosted on the public domain and anyone can register and contribute, once admin approves the registration. Admin can assign three types of roles for crowd-sourcing, annotator, annotation approver and project admin.



(a) A sample verse and tagging suggestions (b) Metadata

Figure 2: Annotation Environment

5 Experiment

To investigate the performance of the heuristics in the task of identification of NEs, we have experimented on a set of 60 manually annotated verses containing 997 words. 5 verses are selected randomly from each of the 12 cantos.To investigate the performance of the heuristics in the task of identification of NEs, we have experimented on a set of 60 manually annotated verses containing 997 words. 5 verses are selected randomly from each of the 12 cantos. We took the help of two annotators from Sanskrit background. The first annotator was given the raw verses and he adds 81 NE annotations. The second annotator was given the raw verses along with the annotations done by the previous annotator. The second annotator verified and accepted all those 81 NEs and added some additional NEs. However, as those additional NEs were not verified by a third person, we did not include it in our experiment. Both the annotators have used our deployed annotation platform for the task. So, it is a binary classification task where the classless are highly imbalanced, 81 NE and 916 non-NE words. We have reported Precision, Recall, and F-score only for NE class identification in Table 1. The F-score calculated is the harmonic mean between precision and recall.

$$P = \frac{True\ Positives}{(True\ Positives + False\ Positives)} \tag{3}$$

$$R = \frac{True\ Positives}{(True\ Positives + False\ Negatives)} \tag{4}$$

$$F1 = \frac{2 * P * R}{(P + R)} \tag{5}$$

As discussed earlier, the basic idea is to match the transliteration of a Sanskrit word with the words in English translation. The results in Table 1 shows the performance of two string matching algorithms, Levenstein distance, and Jaro-Winkler similarity. We also check the impact of

One annotator has Master and another has Doctorate degree

selecting only those words in English translation which starts with a capital and applying a dictionary to exclude general English words before performing string matching. Also, an experiment is done to check whether doing lemmatization of Sanskrit words before string matching improves the performance over using the actual word. The stems are obtained using SRH (Goyal et al., 2012).

If string similarity/distance between the Sanskrit transliteration and a word in the translation is above/below a certain threshold then the word in the translation is marked as NE. Changing the threshold value changes the performance of the heuristic. For Levenstein distance we tried thresholds in the range 0 to 8 and for Jaro-Winkler similarity in the range 0.5 to 1. Table 1 reports the maximum F1 value achieved using grid search over the range of thresholds.

	Using word			Using stem		
Heuristic	P	R	F1	P	R	F1
Levenstein	0.76	0.47	0.58	0.79	0.58	0.67
Capital + Levenstein	0.69	0.63	0.66	0.80	0.60	0.69
Capital + Dictionary + Levenstein	0.78	0.69	0.74	0.92	0.59	0.72
Jaro-Winkler	0.81	0.69	0.75	0.83	0.70	0.76
Capital + Jaro-Winkler	0.85	0.72	0.78	0.85	0.72	0.78
Capital + Dictionary + Jaro-Winkler	0.93	0.70	0.80	0.93	0.70	0.80

Table 1: Performance of different heuristics

Here we can observe, Jaro-Winkler is performing better than simple Levenstein distance. Also, checking whether the word starts with a capital and applying a dictionary to exclude general English words improves performance. It is observed that lemmatization improves performance if we use a tighter threshold value. For instance, if we apply direct string matching(Jaro-Winkler similarity threshold = 1), using stems 39 true positives(TPs) are identified while using the word 15 TPs are identified. But lemmatization is a costly operation and with tighter threshold though precision is high, recall drops reducing the overall F-score. Figure 3 plots the F-score of the best performing heuristic for different string matching threshold limits. The optimal threshold value for Jaro-Winkler similarity in both cases when matching is performed using the word and using the stems is 0.8. At this threshold, Jaro-Winkler similarity gives a similar performance without lemmatization also. So, the final pre-annotation of the data is done without performing lemmatization.

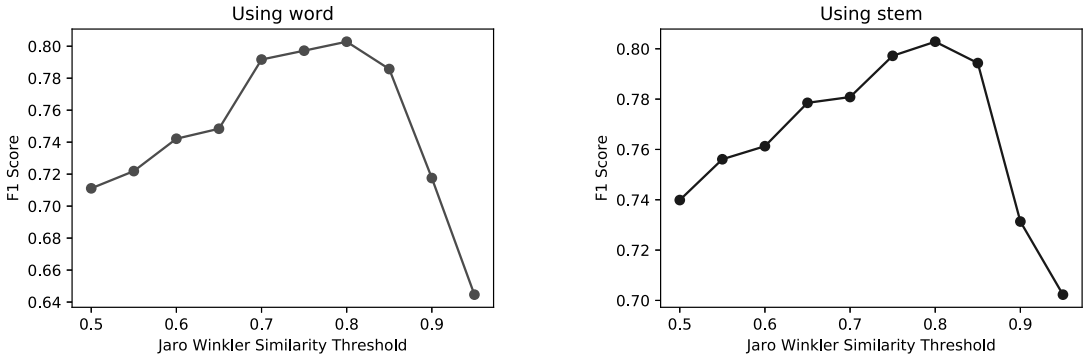


Figure 3: Plot of F1 at different string metric thresholds

6 Error Analysis

The heuristics work based on the assumption that NEs have a similar form in different languages. It is not always true. In *Bhāgavatam*, many times a single character has different names but in English translation uses only single name for all variation. For example, *'bhagavān rudraḥ'* is translated as *'Lord Śiva'*. So here the forms are not similar. Sometimes the translation instead of the name gives a description. For example, *'indra'* is translated as 'the King of heaven'. In another case, if the Sanskrit word is part of English vocabulary then also it is not marked. For example, *'buddhāya'* – *'to Lord Buddha'* , here *Budhha* is present in the dictionary as an English word, so the comparison is not done. The tool can be improved to handle these cases through dynamic suggestions. We have discussed it in Section 7.

7 Related Work

Our work aims to improve annotators' efficiency in preparing a Sanskrit NER dataset using pre-annotation. Pre-annotation has been wildly applied for the task of NER in different domains such as biomedical (Lingren et al., 2013), astrophysical (Hachey et al., 2005), etc. Lingren et al. (2013) studied the impact of pre-annotation on the speed of manual annotation of clinical trial announcements. They evaluated dictionary/gazetteer based approaches for pre-annotating entities related to symptoms and diseases. The study concluded that time savings were statistically significant and present in all of the experiments when the annotator used pre-annotated text. Savary et al. (2010) in their work of creating a largescale NE annotated dataset in Polish used a rule-based method to automatic pre-annotation of named entities, then the annotations are manually corrected and completed by linguistic experts. They customized a graphical tree editor TrEd to provide an annotation environment for manual correction of annotations. Similar works have been done for Indian regional languages also. Saha and Majumder (2018) developed a Hindi NER system without using any manually annotated training corpus. They prepared the initial training data using gazetteer and context patterns matching. They first extracted names from an English corpus using a pre-trained English NER system. Then the English gazetteer lists were used to transliterate into Hindi using a two-phase transliteration approach.

Annotation frameworks and tools are also developed which include some kind of pre-annotation module and use active learning to reduce the efforts of annotators. Yimam et al. (2014) applied their text annotation tool WebAnno that includes a machine learning component for automatic annotation suggestions of span annotations. They achieved an increase in annotation speed of about 21% with automatic suggestions in a German NE annotation project. Stenetorp et al. (2012) achieved 15% decrease in total annotation time using their rapid annotation tool BRAT on a multi-category entity mention annotation task by augmenting the annotation process with input from statistical and machine learning methods. Yang et al. (2018) from their experiment with open-source text span annotation tool YEDDA noted that annotation time can be compressed by 16.47% through intelligent recommendation. Experiments with more recent tools like EPAD showed that it shortened almost 60.0% of the total annotation time, and improved 12.7% of F-measure for annotation quality on a medical record named entity recognition task (Teng et al., 2019). Another tool, AlpacaTag which is implemented based on Doccano provides suggestions using active learning as well as a dictionary of frequent noun phrases and already annotated spans (Lin et al., 2019).

In computational Sanskrit, efforts are directed towards reducing annotation work by building software to allow semi-automated annotation. The SRH platform help linguists create a morphologically tagged corpus in a semi-automatic manner to reduce annotation labor (Goyal et al., 2012; Goyal and Huet, 2016). The interface allows a Sanskrit scholar to do segmentation (sandhi analysis), tagging, and parsing of a corpus by selecting from automatically presented words, stems, and morphological tags.

From these studies we hypothesize that one can take advantage of these approaches to easily build annotated resources for resource-poor languages like Sanskrit.

8 Conclusion and Future Direction

The approach to build Sanskrit NER dataset discussed in this paper is based on static pre-annotation. As there is no previous work in this direction, our work provides an overview and challenges of NER for Sanskrit. We explore the possibility of using external resources such as parallel corpora for the development of annotated datasets. An annotated dataset can help progress further research in this direction. Currently, in the annotation environment the annotators are given fixed suggestions based on the pre-annotation. The tool can be modified to improve the suggestions by taking into consideration the already annotated and verified annotation by annotators. To do so, we intend to integrate active learning strategy (Teng et al., 2019) in our system. Static suggestions also can be improved. The pre-annotation heuristic currently works based on language-independent string-matching algorithms by identifying strings of similar form in transliteration and translation of words. On a small manually annotated dataset it is able to perform reasonably with an F-measure of 0.80, precision 0.95 and recall 0.70. Language dependent analysis and rule-based approaches can be applied to handle difficult linguistic cases and to improve recall. Also, other approaches to compare the names, such as n-gram based approach (Jahangir et al., 2012), can be tested. Another way to improve the suggestions is by building a better gazetteer. One way the gazetteer can be improved is by identifying NEs from the translated text with the help of available NER systems of English (Saha and Majumder, 2018).

Acknowledgement

The authors are thankful to the anonymous reviewers for their valuable feedback. The authors would like to thank Malay Maity, University of Hyderabad, and Dr. Pavankumar Satuluri, Assitant Professor, Chinmaya Vishwavidyapeeth for their help in data annotation and helpful feedback.

References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5360–5369, Hong Kong, China, November. Association for Computational Linguistics.

Kalina Bontcheva, Diana Maynard, Hamish Cunningham, and Horacio Saggion. 2002. Using human language technology for automatic annotation and indexing of digital library content. In Maristella Agosti and Costantino Thanos, editors, *Research and Advanced Technology for Digital Libraries*, pages 613–625, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.

Lars Borin, Dimitrios Kokkinakis, and Leif-Jöran Olsson. 2007. Naming the past: Named entity and Animacy recognition in 19th century Swedish literature. In *Proceedings of the Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2007).*, pages 1–8, Prague, Czech Republic, June. Association for Computational Linguistics.

Lou Burnard and Syd Bauman. 2008. Tei p5: Guidelines for electronic text encoding and interchange.

Annalina Caputo, Pierpaolo Basile, and Giovanni Semeraro. 2009. Boosting a semantic search engine by named entities. In Jan Rauch, Zbigniew W. Raś, Petr Berka, and Tapio Elomaa, editors, *Foundations of Intelligent Systems*, pages 241–250, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia, June. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July. Association for Computational Linguistics.

Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, Muriel Visani, and Jean-Philippe Moreux. 2017. Impact of ocr errors on the use of digital libraries: Towards a better access to information. In *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*, JCDL '17, page 249–252. IEEE Press.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, page 73–78. AAAI Press.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4:145, 10.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India, December. The COLING 2012 Organizing Committee.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 144–151, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Faryal Jahangir, Waqas Anwar, Usama Ijaz Bajwa, and Xuan Wang. 2012. N-gram and gazetteer list based named entity recognition for Urdu: A scarce resourced language. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 95–104, Mumbai, India, December. The COLING 2012 Organizing Committee.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Jan Martin Keil. 2019. Efficient bounded jaro-winkler similarity based search. *BTW 2019*.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for Sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada, August. Association for Computational Linguistics.

Amrith Krishna. 2019. *Addressing Language Specific Characteristics for Data-Driven Modelling of Lexical, Syntactic and Prosodic Tasks in Sanskrit*. Ph.D. thesis, IIT, Kharagpur.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. 2010. Sanskrit compound processor. In Girish Nath Jha, editor, *Sanskrit Computational Linguistics*, pages 57–69, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. 2019. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy, July. Association for Computational Linguistics.

Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. 2013. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3):406–413, 09.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human. Software available from https://github.com/doccano/doccano.

Sujan Kumar Saha and Mukta Majumder. 2018. Development of a hindi named entity recognition system without using manually annotated training corpus. *International Arab Journal of Information Technology*, 15:1088–1098, 11.

Agata Savary, Jakub Waszczuk, and Adam Przepiórkowski. 2010. Towards the annotation of named entities in the national corpus of polish. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.

Beth M. Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

Fei Teng, Minbo Ma, Zheng Ma, Lufei Huang, Ming Xiao, and Xuan Li. 2019. A text annotation tool with pre-annotation based on deep learning. In Christos Douligeris, Dimitris Karagiannis, and Dimitris Apostolou, editors, *Knowledge Science, Engineering and Management*, pages 440–451, Cham. Springer International Publishing.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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*, pages 142–147.

Monier Monier Williams, Ernst Leumann, and Carl Cappeller. 1899. *A Sanskrit-English dictionary: Etymologically and philologically arranged with special reference to Cognate Indo-European languages*. At the Clarendon Press.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California, June. Association for Computational Linguistics.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. 2018. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia, July. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Richard Eckart de Castilho, and Iryna Gurevych. 2014. Automatic annotation suggestions and custom annotation layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, Baltimore, Maryland, June. Association for Computational Linguistics.