BWN - A Software Platform for Developing Bengali WordNet

Farhana Faruqe Mumit Khan
Center for Research on Bengali Language Processing,
Department of Computer Science and Engineering,
BRAC University, 66 Mohakhali, Dhaka, Bengalidesh
E-mail: farhanadiba@gmail.com, mumit@bracu.ac.bd

Abstract-Advanced Natural Language Processing (NLP) applications are increasingly dependent on the availability of linguistic resources, ranging from digital lexica to rich tagged and annotated corpora. While these resources are readily available for digitally advanced languages such as English, these have yet to be developed for widely spoken but digitally immature languages such as Bengali. WordNet is a linguistic resource that can be used in, and for, a variety of applications from a digital dictionary to an automatic machine translator. To create a WordNet for a new language however is a significant challenge, not the least of which is the availability of the lexical data, followed by the software framework to build and manage the data. In this paper, we present BWN, a software framework to build and maintain a Bengali WordNet. We discuss in detail the design and implementation of BWN, concluding with a discussion of how it may be used in future to develop WordNets for other languages as well.

I. INTRODUCTION

One measure of the "digital maturity" of a natural language is the richness and diversity of linguistic resources available for that language – from the simple digital dictionaries to the complex annotated corpora – needed for advanced natural language processing applications such as automatic machine translation. Bengali, despite being very widely spoken [8], is only just beginning to see the development of these linguistic resources. One such important resource is WordNet, a lexical semantic database for a language [1]. The basic building block of WordNet is a synonym set or *Synset*, a word sense with which one or more synonymous words or phrases are associated. Each synset in WordNet is linked with other synsets through the lexical relations synonymy and anotnymy, and the semantic relations hypernymy, hyponymy, meronymy, troponymy, etc. The applications of WordNet range from creating digital lexica to performing word-sense disambiguation in automatic machine translation. The synonym set {পাখি, গগনগতি, খেচর, চিড়িয়া, নভৌকা, পংখি, পখী, পক্ষধর, পক্ষালু, পক্ষী, পতগ, পত্রী, বিহগ, বিহঙ্গ, বিহঙ্গ} and {পাখি, জমির একক বিশেষ, ৩০ কানি ভূমি, ২৬/৩৩/৩৫ শতাংশ, অঞ্চল একক} for example can serve as an unambiguous differentiator of the two meanings of "পাখি". Such synsets are the basic entities in WordNet; each sense of a word is mapped to a separate synset in the WordNet, and synset nodes are linked by semantic relationships, such as hypernymy. Building a WordNet for a language faces two primary challenges – creating the lexical data, and the software framework to store and manage that data. The primary focus of this paper is on the design and implementation of BWN, which is a framework to enable building and using Bengali WordNet.

The design of Bengali WordNet closely follows that of the English WordNet [2]. The software design that we detail in this paper allows the linguists to import lexical data in a "batch" mode, and then allows for visual querying and editing of all the relations in the data. The basic design to support data import and then subsequent queries is relatively simple; however, support for incrementally building the WordNet and for editing the data using a visual interface are two key features of BWN, and these complicate the design in a significant way.

We start by looking at the current approaches for building a new WordNet and discuss our methodology, and then discuss the design and implementation of BWN. We then conclude with a look at what the future holds for BWN.

II. RELATED WORK AND METHODOLOGY

There are two common approaches for building a WordNet for a target language: (i) a top-down approach, using an existing WordNet in a source language to seed the linguistic data for the target language WordNet, and (ii) a bottom-up approach, where the linguists create the WordNet synsets from scratch without depending on an existing one. The first approach has been tried for a number of WordNets [3][4][5][6][7]. Using a source WordNet as the base, Barbu et al., Chakrabartu et al., and Farreres et al. generate target WordNet by mapping the synsets of the two languages. For the sysnsets to be mappable however, concepts in the source WordNet must exist in the target WordNet being built. Additionally, a significant amount of language resource is required for building a WordNet in this method. For example, a set of synsets strictly aligned with the source WordNet must exist before the new WordNet can be built. This is a significant drawback of building a WordNet from an existing one.

Given that there is a well-defined and well-designed English WordNet, one would be tempted to use that to map the synsets and build a Bengali WordNet of a reasonable quality. However, for that to be successful, there must first be significant level of linguistic similarity between the two languages, which is not

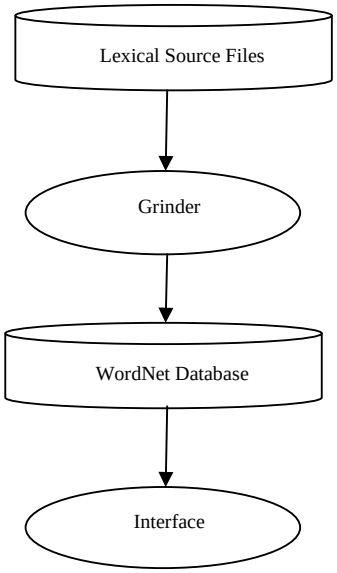

Fig. 1. Block diagram of WordNet system

the case. In addition, Bengali word senses need to be clearly identified in an English-Bengali-English dictionary, which is also not available. Even if there were a rich tagged corpus, a

WordNet can be created semi-automatically. Again, we do not have such a resource available.

There have been many other recent attempts at building a WordNet quickly, such as creating lexical networks by using the web or some well-structured corpora such as Wikipedia, or the BNC corpus. All of these require linguistic resources not yet available for Bengali, leaving us with the bottom-up approach as the most practical one.

Considering the challenges with the first approach, a simpler approach is by using the bottom-up approach, in which we build a WordNet by starting with the words in the target language and not by using an existing WordNet. For BWN, we started by translating the ontology, and chose words using a frequency list from a newspaper corpus. These synsets are compiled in lexical source files, which are then injected into the WordNet database using a "grinder", and the resulting system can then be used through a set of interfaces. We discuss the details of this in the next section.

III. Design and Implementation

Generally, a WordNet software system is comprised of four parts as shown Fig. 1: lexical source files, grinder, WordNet Database and the interface to WNDB to build, use and edit the WordNet. This is the same structure that we follow in BWN as well.

A. Lexical Source Files

These files contain the synsets that are manually compiled by the lexicographers, and are used to eventually populate the WordNet database. In a WordNet, nouns, verbs, adjectives and adverbs are organized into synsets, which are further arranged into a set of lexical source files by syntactic category. This is

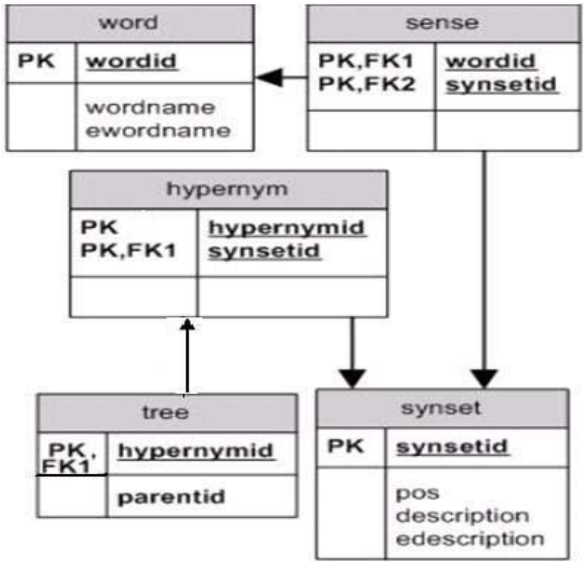

Fig. 2. Block diagram of the WordNet database

where the all the linguistic information is kept, typically hand-crafted by the linguists. The schema used for nouns in the lexical source file is shown below:

Word name / *Word name(english)* / *description* / *Pos* / ||*description(english)*||
Hypernyms:
Synonyms:

And a sample "noun" record is shown below.

কাজ|work| কিছু করা বা তৈরির লক্ষে সরাসরি কার্যক্রম| বিশেষ্য|
||work -- (activity directed toward making or doing something)||
hypernyms:| কার্যক্রম | কৃতকর্ম | ঘটিত বিষয় | মনস্তাত্ত্বিক-বিষয় | বিমূর্তন | বিমূর্ত-সত্তা | সত্তা |
synonyms: কর্ম, কর্মকাণ্ড, কাজ, কাজকাম, কাম, কার্য

B. Grinder

The grinder is used to convert these lexical source files in a form that can be injected into the WordNet Database (WNDB). Basically, it parses and processes the text from the lexical source files into records, and then stores each record in the WNDB.

C. WordNet Database (WNDB)

WNDB is the heart of WordNet for any language. For BWN, the basic design is similar to "Wordnet SQL Builder" [2], shown in Fig. 2. However, as we shall soon see, there are significant differences under the hood, primarily to support incremental building of the database, and editing of the synsets directly via the user interface. One of the design goals is to ensure that WNDB is extensible to new lexical relations between synsets. In addition, in the word table, we store the English word that can be used to link to other WordNets such as the EuroWordNet in the future. In the sense table, both the word and the synset are mapped together. In the synset table, we generate an ID for a synset but do not create the synset itself. We regenerate the synset at run-time from the sense and

word tables, which serves a very important role in the case of an edit or update operation.

D. WNDB Interface

There are essentially three different interactions with WNDB, the underlying WordNet database. The first is to create the initial database using the lexical source files, and then to incrementally update the database, which is a feature that significantly contributes to the database schema complexity; the second is to use the database to query the data; and, the third is to edit the lexical data, which is the other reason behind the database schema complexity. In this rest of this section, we look at each of these interactions in detail.

TABLE 1
Word table after data entry

wordid	wordname	ewordname
1	কাজ	Work
2	কর্ম	Work
3	কাজকাম	Work

TABLE 2
Synset table after data entry

synsetid	description	edescription	pos
1	কিছু করা বা তৈরির লক্ষে সরাসরি কার্যক্রম	work -- (activity directed toward making or doing something)	বিশেষ্য

TABLE 3
Sense table after data entry

wordid	synsetid
1	1
2	1
3	1

1) Update WNDB: The Grinder takes each record from the lexical source file, then splits the text according to the database field and then stores it into the database. The process starts with reading each record from a lexical source file. To illustrate the process, let us take the following sample record in a lexical source file:

" কাজ। work। কিছু করা বা তৈরির লক্ষে সরাসরি কার্যক্রম। বিশেষ্য।
||work -- (activity directed toward making or doing something)||
hypernyms:| কার্যক্রম | কৃতকর্ম | ঘটিত বিষয় | মনস্তাত্বিক-বিষয় | বিমূর্তন | বিমূর্ত-সত্তা | সত্তা | synonyms:কর্ম, কাজকাম"

After splitting the text, the grinder updates the word table with the value of *wordid* (auto incremented integer), *wordname*, and *ewordname*. Each synonym word is also entered into the word table, (see Table 1).

The Grinder then updates the synset table with *synsetid* (auto incremented integer), *description*, *edescription*, and *pos* (see Table 2).

The Grinder then updates the sense table with those *wordids* and the particular *synsetid* (see Table 3).

To update the hypernym table (Table 4), we need the *synsetid* of that particular record and its corresponding *hypernymid*; because each synset, with the exception of "entity/ সত্তা", may have one or more hypernyms. For that, we have to match each hypernym with the *wordname* field's value in the word table and then take the *wordid*; with this *wordid*, we have to find out the *synsetid* (because the *hypernymid* is nothing but a *synsetid*) from the sense table. Here we assume that all of these hypernym words already exist in the word table.

TABLE 4
Hypernym table after data entry

Synsetid	Hypernymid
1	2
1	3
1	4
1	5
1	6
1	7
1	8

The tree table (Table 5) keeps track of the parent of each hypernym word, because hypernymy relates each child to its parent. Then the Grinder updates the *tree* table with *hypernymid* and *parentid* (which is also a *synsetid*). Since "entity/সত্তা" does not have a parent, its *parentid* is given a value of 0 (zero) to indicate that.

TABLE 5
Tree table after data entry

hypernymid	parentid
2	3
3	4
4	5
5	6
6	7
7	8
8	0

At this point, a specific complication may arise because of BWN's support of incremental update – there may be some hypernym words that do not currently exist in the WordNet Database. As we have noted earlier, this is one of the key features of BWN, and one that contributes significantly to the complexity of the design. We still have to enter these words into the database because the hypernym and tree tables' values are fully dependent upon the *synsetid*. However, the currently entered record is only partially complete, which is why we have to mark it as such. We do that by marking it with a special tag, "hypernym", to be updated later when its corresponding entity record is encountered in the lexical data. When this record eventually comes as an entity, we update the record tagged as a "hypernym" with its complete value. In fact, we have to consider all the synonym words, and not just the entity word, because the previously entered hypernym word may exist as part of a synset. Let us illustrate this with the following example record:

"কালবিন্দু। measure, quantity, amount তা ক্ষণিক সময়। বিশেষ্য।||point, point in time -- (an instant of time)||
hypernyms: |বিমূর্তন | বিমূর্ত-সত্তা | সত্তা | synonyms: কালবিন্দু"

After entering the data word table (see Table 6), synset table (see Table 7), and the sense table (see Table 8) as discussed earlier, the data looks like the following:

TABLE 6
Word table after data entry

wordid	wordname	Ewordname
12	কালবিন্দু	measure, quantity, amount

TABLE 7
Synset table after data entry

synsetid	description	Edescription	Pos
9	তা্ ক্ষণিক সময়	measure, quantity, amount	বিশেষ্য

TABLE 8
Sense table after data entry

wordid	Synsetid
12	9

TABLE 9
Word table after data entry

wordid	wordname	ewordname
13	বিমূর্ত-সত্তা	hypernym

TABLE 10
Synset table after data entry

synsetid	description	Edescription	Pos
10	বিমূর্ত-সত্তা	hypernym	বিশেষ্য

TABLE 11
Sense table after data entry

wordid	Synsetid
13	10

TABLE 12
Hypernym table after data entry

synsetid	hypernymid
9	6
9	10
9	8

TABLE 13
Tree table after data entry

hypernymid	parentid
6	10
10	8
8	0

TABLE 14
Word table after updated data

wordid	wordname	ewordname
13	বিমূর্ত-সত্তা	Abstract Entity

TABLE 15
Synset table after updated data

synsetid	description	edescription	pos
10	শুধুমাত্র বিমূর্ত (দৈহিক রূপহীন) অস্তিত্ব আছে এমন সত্তা	abstract entity -- (an entity that exists only abstractly)	বিশেষ্য

Now suppose that one of the hypernym words "বিমূর্ত-সত্তা" does not yet exist in the database; in this case, we have to enter this word into the word table (see Table 9) and the synset table (see Table 10), generating the *wordid* and the *synsetid*; then, we have to enter it in the sense table (see Table 11) with the generated *wordid* and *synsetid*.

Then we insert the value into the hypernym table (see Table 12) and the tree table (see Table 13) as discussed earlier.

Now, later one, when this "বিমূর্ত-সত্তা" hypernym word shows up as an entity, we have to update the word and the synset tables with new value, while the sense table remains the same.

For example, the following records add the hypernym word as an entity:

অমূর্ত-সত্তা । Abstract Entity । শুধুমাত্র বিমূর্ত (দৈহিক রূপহীন) অস্তিত্ব আছে এমন সত্তা। বিশেষ্য। ||Abstract Entity -- (an entity that exists only abstractly)||hypernyms:সত্তা | synonyms: অরূপ-সত্তা, নিরূপ-সত্তা, বিমূর্ত-সত্তা, সত্তা।

Here "বিমূর্ত-সত্তা" comes as part of a synonym, and not as an entity name. Now we have to update the word table (see Table 14) and the synset table (see Table 15) with the new value.

The rest of the entry – the entity name, the synonym, and the hypernym will be entered in the same manner as discussed earlier.

2) Using WNDB: The second interface to the WNDB is for querying the data in WNDB, as shown in Figures 3 and 4. A typical scenario is the following:

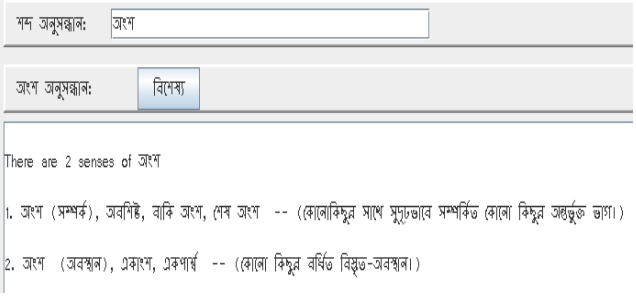

Fig. 3 Result of a search option

TABLE 16
Word table

wordid	wordname	ewordname
20	অংশ (সম্পর্ক)	part, portion
25	অংশ (অবস্থান)	region, part

TABLE 17
Sense table

wordid	synsetid
20	**17**
25	**19**

TABLE 18
Sense table

wordid	synsetid
20	17
21	17
22	17

TABLE 19
Word table

wordid	Wordname	ewordname
20	অংশ (সম্পর্ক)	part, portion
21	অবশিষ্ট	part, portion
22	বাকি অংশ	part, portion

TABLE 20
Synset table

synsetid	description	edescription	pos
17	কোনোকিছুর সাথে সুদৃঢ়ভাবে সম্পর্কিত কোনো কিছুর অন্তর্ভুক্ত ভাগ	……	বিশেষ্য
19	কোনো কিছুর বর্ধিত বিস্তৃত-অবস্থান	……	বিশেষ্য

TABLE 21
Hypernym table

synsetid	hypernymid
17	8
17	9
17	10

TABLE 22
Tree table

hypernymid	parentid
8	9
9	10
10	0

1. User enters the query text into query field as shown in the following figure.

2. The WNDB search engine first finds the sense (or senses) of that given word from word table (see Table 16), then maps the *wordid* to the *synsetid* from the sense table (see Table 17), and then returns those *synsetids.*
   In this example, "অংশ" has two senses (each word represents a single value, as mentioned earlier).
   So, the returned *synsetids* are 17 and 19.
3. For each of the resulting *synsetids*, we have to find all the *wordids* from the sense table. To create a synonym set, we have to find all the *wordnames* from the word table after matching the *wordids* for a particular synsetid. Tables 18 and 19 show these procedures. Here, we consider only one *synsetid*, 17. For *synsetid* 17, the synonyms are {অংশ (সম্পর্ক), অবশিষ্ট, বাকি}.
4. Then, we find the description for each *synsetid* from the synset table (see Table 20) with those *synsetids*. Then the search result is shown in Fig 3.
5. To view a noun's hypernymy relation, as shown in Fig 4. The application execute steps 2-4 for each sense, and then, within each sense, it performs the following steps:
   a. It finds the *hypernymids* from the hypernym table (see Table 21) for the specific *synsetid.*
   b. The application also has to track each of the hypernym's parent from the tree table (see Table 22) to track the child-parent relation.
6. After performing steps 5 (a) and (b), it shows the hypernym from child to parent order.

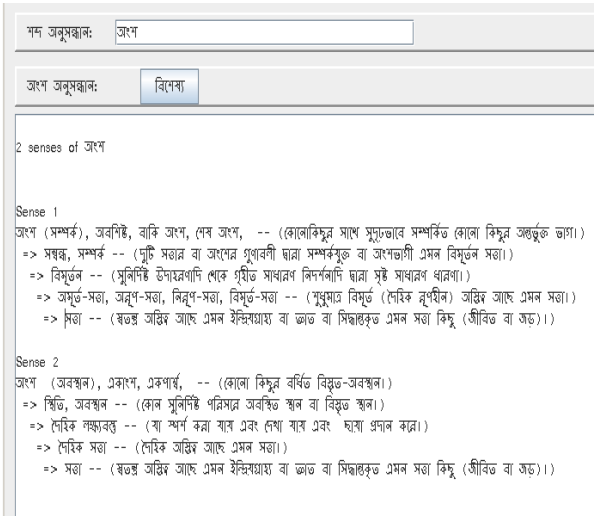

Fig. 4. Hypernym relation of a noun

3) Editing WNDB:

BWN supports editing any existing record through a user interface shown in Fig 5. It also supports a limited version of delete operation, because an unrestricted deleted may destroy the underlying tree. If the user wants to delete a record, there are three cases to consider:

- If the record has synonym, then we can delete it (updates only the word table);
- If the record is used as a hypernym entry then we cannot delete it without risking relational integrity;
- If the record is not used as a hypernym entry, then we can delete that record, which affects all tables except the tree table.

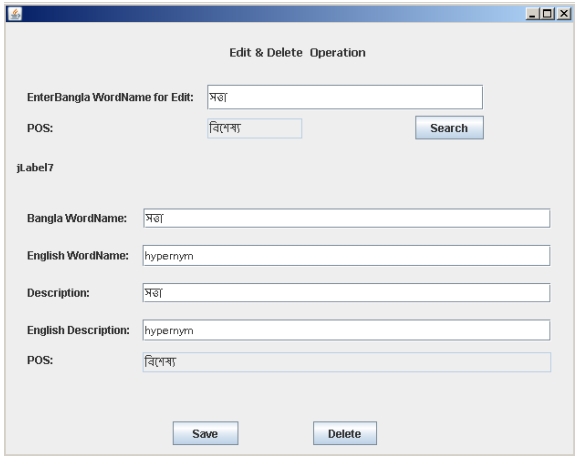

Fig. 5. Edit interface

IV. CONCLUSION AND FUTURE WORK

We present the design and implementation of BWN, a software framework for developing a Bengali WordNet. BWN at the basic level supports building the WordNet database from lexical source files using a grinder, and then supports querying the data using an interface; in addition, it has two key features

not found in other designs support for incremental building of the WordNet database, and for editing the WordNet data using an interface. These two key features significantly contribute to the complexity of the design and implementation of BWN. BWN makes no assumption about the underlying language, so it should be extendable to other languages as well. Future work will focus on two fronts – improving the interface to the underlying WordNet database such as creating Webservice and .NET bindings, and to link to non-Bengali WordNets such as the Hindi and Euro WordNets.

V. ACKNOWLEDGMENT

This work has been supported in part by the PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada. The Center for Research on Bangla Language Processing (CRBLP) is supported in part by IDRC and Microsoft Corporation.

VI. REFERENCES

[1] Fellbaum C. “WordNet: An Electronic Lexical Database”, MIT press. 1998.

[2] “wordnet sql builder”, http://wnsqlbuilder.sourceforge.net/schema.html, last accessed: 16 Oct 2008

[3] Manish Sinha, Mahesh Reddy and Pushpak Bhattacharyya, "An Approach towards Construction and Application of Multilingual Indo-WordNet", 3rd Global Wordnet Conference (GWC 06), Jeju Island, Korea, January, 2006.

[4] Piek Vossen, “EuroWordNet: A Multilingual Database with Lexical Semantic Networks”, Computational Linguistics, Volume 25, Number 4, September 1999.

[5] Farreres, Xavier, German Rigau and Horacio Rodriguez. “Using WordNet for building WordNets.” In: Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems, Montreal, 1998.

[6] Barbu, Eduard and Verginica Barbu Mititelu, “Automatic Building of Wordnets" In: Proceedings of the International Conference Recent Advances in Natural Language Processing, Borovets, Bulgaria, pp. 329-332, 21-23 September 2005.

[7] Debasri Chakrabarti, Gajanan Rane and Pushpak Bhattacharyya, Creation of English and Hindi Verb Hierarchies and their Application to English Hindi MT, International Conference on Global Wordnet (GWC 04), Brno, Czeck Republic, January, 2004.

[8] Wikipedia contributors, Bengali language, Wikipedia, The Free Encyclopedia; 2008 Oct 27, 17:06 UTC [cited 2008 Oct 28]. Available: http://en.wikipedia.org/w/index.php?title=Bengali_language&oldid=248011954