Text Normalization system for Bangla

Firoj Alam, S. M. Murtoza Habib, Mumit Khan

Center for research on Bangla Language Processing, Department of Computer Science and Engineering, BRAC University, Bangladesh

{firojalam, habibmurtoza, mumit}@bracu.ac.bd

Abstract

This paper describes a process of text normalization system of Bangla language (exonym: Bengali) by identifying the semiotic classes from Bangla text corpus. After identifying the semiotic classes a set of rules were written for tokenization and verbalization. This study is important for Text-To-Speech (TTS) system and as well as in language model for speech recognition.

Index Terms: speech synthesis, text normalization

1. Introduction

The goal of this paper is to design and implement a text normalization system for Bangla Text-To-Speech (TTS) system. TTS is one of the major areas of research on natural languages. The process of TTS involves the inputting of a text through console, file, OCR, and other means and converting the text into spoken words. Before a text is ready to undergo the process of TTS, it must first be pre-processed to remove the ambiguities and convert some Non Standard Words (NSWs) into their standard word pronunciations. TTS systems work with text in everyday use (real text) [1]. Therefore, one may expect to come across certain symbols such as $ and %, acronyms such as NATO and WHO, orthographic abbreviations such as Mr. and etc, ordinals such as 1st and 3rd, and many more. When converting these texts to speech, one has to use the pronunciation of the words rather than symbols so that "Mr." will sound like "mister", "1st" will sound like "first", and so on. Moreover, certain numbers have to be pronounced as individual digits or as a whole. For example, a phone number such as 88028624921 will be pronounced "eight eight oh two eight six two four nine two one", but it will be pronounced as "eight thousand eight hundred and two crores eighty six lacs twenty four thousand nine hundred and twenty one" if it is refers to as a measurement such as population. This process is called Text Normalization. Moreover, during speech recognition such as Speech-to-Text systems and other document creations from recognized text, inverse text normalization is used [2].

To the best of our knowledge this is the first published account of Bangla text normalization system, designed for TTS. This study develops a method to normalize Bangla text using a rule based system rather than using a decision tree and a decision list [1] to ambiguous tokens. The basic model [1] [4] [7] of a text normalization system is the same for all languages except a language dependent rules and framework. Here our attempt is to find possible changes and enhancements which are required to implement a Bangla text normalization system.

A brief literature review is given in section 2, followed by a description of the methodology in section 3. The analytical results are presented and discussed in section 4. A summary and conclusions of the study are given in section 5.

2. Literature review

Considerable work may has been done in text normalization and disambiguation of other languages such as English, Hindi, Chinese, Japanese and other well resourced languages. Many little works has been done on under resourced languages such as Bangla. The basic similarity among the work done is that each involves repeated sequences of tokenization, token classification, token sense disambiguation and standard word generation to get the normalized text. This effort differs in the ways of tokenization (delimiter, use of Flex, etc), the methods of disambiguation (n-gram comparison, statistical models and POS tagging) and the methods of word representation (tagging, plain text).

There are various techniques used in lexical disambiguation in the English language. Use of decision lists and combining the strengths of decision trees, N-gram taggers and Bayesian classifiers, a text can be processed to resolve disambiguation in TTS synthesis [3]. Most of the work done on text normalization uses tags after an NSW has been identified [1] [2] [3] [4]. After the type of the NSW is resolved, the resulted word representation is tagged. For example, in Chinese and Japanese, NUM, NDAY, NDIG, and NTIME are used for numeric NSWs [4].

3. Methodology

This paper talks of a method to normalize Bangla text. Like other work, the basic processes are same: tokenization, token classification, token sense disambiguation and word representation. Before the processes of Bengali text normalization are discussed, it is necessary to first discuss the different classes of Bangla NSWs and their equivalent pronunciation word representations. Where [1] uses decision tree and decision list for disambiguation, but this work uses rule based system. The following section discusses the semiotic class [7] (as opposed to say NSW) identification, tokenization and standard word generation and disambiguation rule. The system diagram of text normalization procedure is shown in figure 1.

According to semiotic classes a lexical analyzer was designed to tokenize each NSW by regular expression using the tool JFlex [8]. We assigned a tag for each token according

to semiotic classes. The outputs of the tokenization are then used in the next step i.e token expander. According to the assigned tag token verbalization and disambiguation was performed by the token expander.

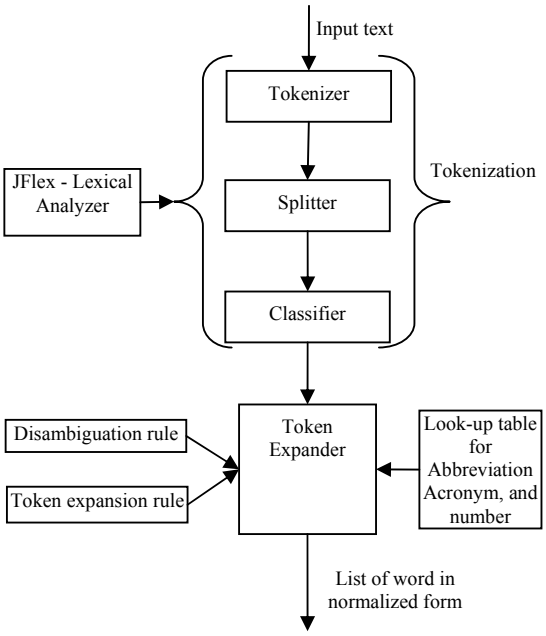

Figure 1: Text normalization system for Bangla

3.1. Semiotic class identification

We identified a set of semiotic classes which belongs to the Bangla language. To do this, we selected a news corpus [9], forum [10] and blog [11], then we proceeded in two steps to identify the semiotic classes: (i) Python [12] script was used to identify the semiotic class from news corpus and we manually checked it in the forum and blog (ii) we defined a set of rules according to context of homographs or ambiguous tokens. The result is a set of semiotic classes in Bangla text as shown in table 1.

Semiotic class/token type	Example
English text	জাভা Platform Independent বলে
Bangla text	এই সময়ের সবচেয়ে
Numbers (cardinal, ordinal, roman, fraction, ratio, range)	১২১,২৩,২৩৪; ১ম, ২য়, ৩য়; I, II, III, ১২.২৩, ২৩,৩৩.৩৩; ১/২, ২৩/২৩; ১২:১২; ১২-২৩
Telephone and mobile number	০২৯৫৬৭৪৪৭; ০১৫২৩০৩৩৯৮ (19 different formats)
Years	২০০৬; ১৯৯৮; ৯৮ সালে
Date	০২২০০৬-০৬- (12 different formats)
Time	৪.২০ মিঃ; ৪.২০ মিনিট;
Percentage	১২%
Money	১০ ৳
E-mail	আমার ই-মেইল ঠিকানা: abc@yahoo.com
URL	সফটওয়্যারটি http://google-gdata.googlecode.com সাইট
Abbreviation	ডঃ ;মোঃ ;সাঃ
Acronym	ঢাবি ;বাউবি, কেবি
Mathematical equation	(১+২=৩)

Table 1: Possible token type in Bangla text

In Bangla text we have English text and even Arabic and Urdu text may also be present. Other than Bangla we worked on English text within Bangla text. The non-natural language [7] token such as number, year, date time etc is also available in this multi-text genre (both scripts). So we are handling two types of natural language [7] tokens such as Bangla and English. Our attempt is to build such a text normalization system that can serve almost every domain of Bangla. Handling Arabic and Urdu text along with any specialist domain i.e. medicine, engineering, chemical equations etc. is beyond the scope of this paper. A simple mathematical equation can handle this system. An example of a mathematical equation is shown in table 1.

3.2. Tokenization

We defined each semiotic class to a specific tag and assigned this tag to each class of token. The tokenization undergoes three levels such as: i. Tokenizer ii. Splitter and ii. Classifier. Like English and other South Asian scripts Bangla also uses whitespace to tokenize a string of characters into a separate token. Punctuation and delimiter were identified and used by the splitter to classify the token. Context sensitive rules written as whitespace is not a valid delimiter for tokenizing phone numbers, year, time and floating point numbers. Finally, the classifier classifies the token by looking at the contextual rule. Different form of delimiters was removed in this step. For each type of token, regular expression were written in .jflex format. Then using JFlex toolkit a Lexer file was generated. If a regular expression is matched then we assign a tag in list[i] and token in list [i+1]. In this way the whole tokenization process is performed. All regular expressions were designed according to our predefined semiotic classes and the rules of the context that were obtained in the previous semiotic class identification phase. This study is different than [1], where decision tree and decision list is used for disambiguation. The generated Lexer file was used in the token expansion phase. The generated Lexer is a java class file which was then invoked by a driver class to get the list of the token. According to the tag in the list, each type of token expander class was then invoked for expanding the token.

3.3. Verbalization & disambiguation

The token expander expands the token by verbalizing and disambiguating the ambiguous token. Verbalization [7] or standard word generation is the process of converting non-natural language text into standard words or natural language text. A template based approach [7] such as the lexicon was used for number cardinal, ordinal, acronym, and abbreviations. For expanding the cardinal number, we have calculated the position of the digit rather than dividing by 10. Expanding the token cardinal number we have chosen the following steps: (i). traverse from right to left. (ii). Map first two digits with

lexicon to get the expanded form (i.e. 10 → ten). (iii). After the expanded form of the third digit insert the token "hundred". (iv). Get expanded form of each pair of digit after third digit from the lexicon. (v). Insert the token "thousand" after the expanded form fourth and fifth digit and "lakh" after expanded form of sixth and seventh digit. These processes continue for each seven digits. Each seven digit is divided as a separate block. After each of the second block (traversing from left to right) we insert the token "koti". So the expanded form of token 10910 is "ten thousand nine hundred ten". Abbreviations are productive and a new one may appear, so an automatic process may require solving unknown abbreviations. In [5] an automatic process is shown for the prediction of unknown abbreviations, our effort deals with this problem in an automatic way and looks for possible changes. In case of Bangla acronyms, most of the time people say the acronym as it is without expanding it. For example, দুদক /ḍuḍɔk/ expands to দুর্নীতি দমন কমিশন /ḍurniṭi ḍɔmon komiʃɔn/ but people say it as দুদক /ḍuḍɔk/. So it is a matter of linguistic decision whether we will expand the acronym or not but for the time being we have expanded to the full form by using the lexicon. Bangla has the same type of non-natural language ambiguity like Hindi [1] in the token year-number and time-floating number. For example: (i). the token ১৯৯৮ (1998) could be considered as a year and at the same time it could be considered as number and (ii). the token ১২.৮০ (12.80) could be considered as a floating point number and it could be considered as a time. Context dependent hand written rules were applied for these ambiguities. In case of Bangla, after time pattern ১২.৮০ (12.80) we have a token মিঃ (minute) so we look at the next token and decide whether it is time or a floating point number. In the worst case scenario where our context dependent rule will fail in year-number ambiguity we are verbalizing the token as a pair of two digits. For example, the token ১৯৯৮ (1998) we expand it as উনিশ শত আটানব্বই (Nineteen hundred ninety eight) rather than এক হাজার নয় শত আটানব্বই (one thousand nine hundred ninety eight). This is not wrong in case of Bangla as both are acceptable by this culture. The natural language text is relatively straightforward, and Bangla does not have upper and lower case. The mispellings problem is not dealt with in this system which however, may be handled in a seperate module before the text normalization system. Certain homograph problems exist in Bangla such as an abbreviation homograph token মিঃ may appear as "minute" or it may appear as "mister" and a part-of-speech (POS) homograph such as কর would be pronounced as কর/kɔr/ and করো/kɔro/. The POS homograph problem exists in Bangla verb cases in terms of grade, such as কর/kɔr/ is pronounced in 2nd person informal and করো/kɔro/ is pronounced in 2nd person formal. We have not solved this problem as we do not have any accessible tagged corpus or syntactical parser which can solve this problem. The natural language English text that exists within the Bangla text are kept as it is, we are not dealing any ambiguities that may be present in the English text.

4. Results

The output of this work is the list of words in a normalized form. The performance of this rule based system is 99% for ambiguous tokens such as float, time and currency. These three types of token appear as a floating number. The test was performed by collecting 797 strings from one year news paper corpus [9] using python script for ambiguous token. Among 797 strings 617 were floating point number, 167 were in currency and 13 strings were in time format. The accuracy of these three types of token is: floating point 100%, currency 100% and time 62% as shown in figure 2. The POS homograph ambiguity in Bangla appears in different grades of the same verb and will be resolved in the next step of TTS development after text normalization. Other than the ambiguous token the rest of the token expansion performs satisfactory results in this system. This system will be available online at location <http://www.bracu.ac.bd/research/crblp/download.php> as open source after completing the system as a generalized form so that it can be expanded for other languages which is derived from Brahmi script.

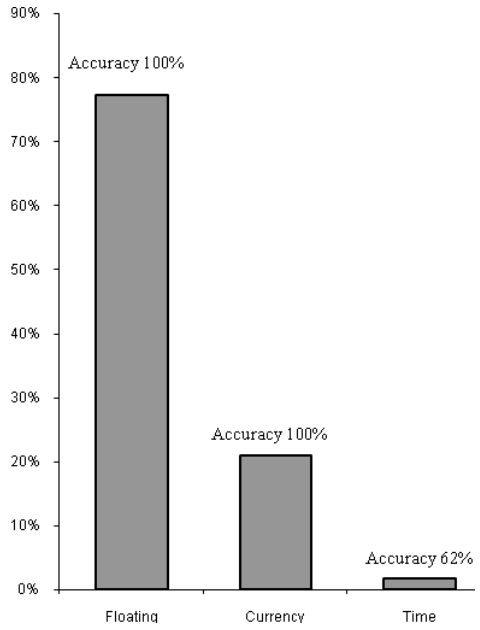

Figure 2: Performance of the system

5. Conclusions

Text normalization is a very important issue in TTS systems. TTS systems will be highly beneficial in various fields and its use will make life easier in unimaginably numerous aspects. In this paper, the preprocessing of TTS, text normalization has been discussed. This is the first working system that has been implemented based on rule without comparing the performance of decision tree and decision list for ambiguous token, because of unavailability of linguistic resources such as a tagged corpus. Hence this work will be helpful in future by integrating with TTS and Speech Recognition combined with comparison of the ambiguity techniques of rule based systems and other classification systems. More work is needed to solve POS homograph and other problematic areas.

6. Acknowledgements

This work has been supported in part by the PAN Localization Project (www.panl10n.net), grant from the International Development Research Center (IDRC), Ottawa, Canada, administrated through Center for Research in Urdu Language Processing (CRULP), National University of Computer and

Emerging Sciences, Pakistan. We would also like to thank Naira Khan (Dhaka University) who helped by providing her comments.

7. References

[1] K. Panchapagesan, Partha Pratim Talukdar, N. Sridhar Krishna, Kalika Bali, A.G. Ramakrishnan, "Hindi Text Normalization" *Fifth International Conference on Knowledge Based Computer Systems (KBCS)*, Hyderabad, India, 19-22 December 2004. http://www.cis.upenn.edu/~partha/papers/KBCS04_HPL-1.pdf

[2] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, *Spoken Language Processing: A Guide To Theory Algorithm And System Development.* Prentice Hall, pp. 706-720, 2001.

[3] Yarowsky D., "Homograph Disambiguation in Text-to-Speech Synthesis" *2nd ESCA/IEEE Workshop on Speech Synthesis,* New Paltz, NY, pp. 244-247,

[4] Olinsky, C. and Black, A., "Non-Standard Word and Homograph Resolution for Asian Language Text Analysis" *ICSLP2000,* Beijing, China, http://www.cs.cmu.edu/~awb/papers/ICSLP2000_usi.pdf

[5] Sproat R., Black A., Chen S., Kumar S., Ostendorf M., & Richards C, "Normalization of Non-Standard Words: WS'99 Final Report" *CLSP Summer Workshop,* Johns Hopkins University, 1999. www.clsp.jhu.edu/ws99/projects/normal

[6] A Raj A., Sarkar T., Pammi S. C., Yuvaraj S., Bansal M., Prahallad K., and Black, A., "Text Processing for Text-to-Speech Systems in Indian Languages" *ISCA SSW6,* Bonn, Germany, pp. 188-193, http://www.cs.cmu.edu/~awb/papers/ssw6/ssw6_188.pdf

[7] Paul Taylor, *Text to Speech Synthesis.* University of Cambridge, 2007. pp.71-111, DRAFT available in: http://mi.eng.cam.ac.uk/~pat40/ttsbook_draft_2.pdf

[8] Elliot Berk, JFlex - The Fast Scanner Generator for Java, 2004, version 1.4.1, http://jflex.de/

[9] "News corpus", Prothom-Alo, http://www.prothom-alo.com/

[10] "Forum", http://forum.amaderprojukti.com/

[11] "Blog", http://www.somewhereinblog.net/

[12] "Python", www.python.org/, version 2.5.2