Improving English-Bharti Braille Machine Translation Through Proper Name Entity Translation

Nisheeth Joshi[1,2] and Pragya Katyayan[1,2]

[1] Department of Computer Science, Banasthali Vidyapith, Rajasthan, India
[2] Centre for Artificial Intelligence, Banasthali Vidyapith, Rajasthan, India
pragya.katyayan@outlook.com, nisheeth.joshi@rediffmail.in

Abstract. Machine Translation has been one of the main areas of research in the dawn of Artificial Intelligence. Although getting a human-level accuracy through a machine was considered as a task that is next to impossible, this cliché is on the verge of being broken. The paper discusses the development of an MT system for English-Bharti Braille. The system translates the English test into Braille which can be read by people who are visually impaired. For this a separate module for handling name entities is incorporated into the MT pipeline. It is an attempt to disseminate knowledge to people who have a craving for learning. The results of the system are compared using BLEU metric and have shown improvement over the baseline system.

Keywords: Machine Translation, Name Entity Recognition, Source Text Rewriting, Syntax Transfer, Bharti Braille.

1 Introduction

Bharti Braille is a type of braille script that is used for writing the Hindi language in India. It was developed by the National Institute of Visually Handicapped (NIVH) in Dehradun, India, to address the needs of blind people who speak Hindi as their primary language. It is important because:

- Accessibility: Bharti Braille makes the Hindi language accessible to blind people, allowing them to read and write in their own language.
- Independence: With Bharti Braille, blind people can communicate independently without relying on sighted people to read or write for them.
- Inclusion: Bharti Braille promotes inclusion by allowing blind people to participate fully in society, including in education, employment, and other activities.
- Cultural preservation: Bharti Braille helps preserve the Hindi language and its cultural heritage by making it accessible to all, regardless of visual ability.

Bharti Braille is an important tool for promoting equality, accessibility, and inclusion for blind people who speak Hindi. It is based on the standard braille system, but with some modifications to accommodate the unique features of the Hindi language. Here are some key features of Bharti Braille:

ICIDSSD 2022, May 24-25, New Delhi, India

DOI 10.4108/eai.24-5-2022.2319020

- Six-dot system: Bharti Braille uses a six-dot system like other braille scripts, with each dot representing a different letter or symbol.
- Consonants: The 33 consonants of the Hindi language are represented by the first 33 letters of the Bharti Braille alphabet.
- Vowels: Bharti Braille uses a diacritic system to represent the 13 vowel sounds of the Hindi language. A dot placed above or below the consonant dot indicates the corresponding vowel sound.
- Additional symbols: Bharti Braille includes additional symbols for punctuation marks, numbers, and special characters.
- Ligatures: To save space, Bharti Braille uses ligatures, which combine two or more letters into a single braille cell.
- Capital letters: Capital letters are indicated by a special capitalization sign at the beginning of a word.

Thus, Bharti Braille is a comprehensive writing system that allows blind people who speak Hindi to read and write their language independently. It is an important tool for promoting accessibility, inclusion, and cultural preservation for this community.

2 Literature Survey

Lewis et al. (2012) have developed acceptable performance items for demonstrating competence I literary Braille. Hong et al. (2012) devised an alternative method for Braille notetakers for visually impaired people in the form of a dedicated notetaking device for Braille. Herzberg and Rosenblum (2012) provided an analysis of 107 mathematical worksheets prepared for visually impaired students on the basis of accuracy. Al-Salman et al. (2012) proposed new method for Braille image segmentation by utilizing between-class variance method with gamma distribution given by Otsu. Their method had two main segments: first, finding optimally estimated thresh-old using the variance technique with gamma distribution mixture and second, using the optimal thresholds for braille image segmentation.

Authman and Jebr (2012) have described a new technique for identifying Braille cells in a single-sided Braille document in Arabic. Their optical Braille Recognition system for Arabic performs two tasks: first is to identify printed Braille cells and second is converting them as normal text. Padmavathi et al. (2013) have proposed a method to convert a scanned document of Braille into completely readable text for-mat. Braille documents were pre-processed to reduce noise and enhance the dots. Dots were then extracted after Braille cells were segmented and was changed in a sequence of numbers. It is then mapped to correct alphabets of the language (English, Hindi or Tamil) and is read aloud with the help of a speech synthesizer. They also gave a method of typing Braille via the numeric pad on keyboard.

Abualkishik and Omar (2013) have introduced a Quran Braille Translator which could translate verses from Quran to Braille. They used extended finite state machine (EFSM) for finding reciting rules of Quran and Markov algorithm to translate reciting rules along with Quran text to appropriate Braille code. Oda et al. (2013) adapted ML

for NLP to improve word segmentation for web-based automatic translation program for braille. They created statistical models using a SVM-based general purpose chunker. Hossain et al. (2013) have identified rules and conventions for Bangla Braille translation based on rules. They proposed a DFA based computational model for MT, which gave acceptable translations. The results were tested by members of visually impaired community.

Al-Salman et al. (2014) have built a Braille copier machine which produced Braille documents in their respective languages. The machine worked as both copier as well as printing system using optical recognition and techniques from image processing. Yamaguchi et al. (2014) have highlighted the problem of accuracy while translating technical notations to Braille. To solve this problem, they have developed a assistive solution for people from STEM background who are not capable of printing. Damit et al. (2014) have mediated a new way of interlinking keyboard inputs from trans-lates to commonly used Braille characters. This enabled visually blessed people to interact with visually impaired people.

Rupanagudi et al. (2014) introduced a new technique of translating Kannada braille to Kannada language. They devised a new algorithm to segment Braille dots and identified characters. Choudhary et al. (2015) have suggested a new approach for supporting communication amongst deaf-blind people. The technique included the use of smart gloves capable of translating Braille alphabets and can communicate the message via SMS. Due to this the user can convey simple messages using touch sensors. Guerra et al. (2015) have developed a prototype using Java which can convert Braille text to digital text. Jariwala and Patel (2015) have developed tool for translation of Gujarati, English and Hindi text to Braille and save it as a data file which can be directly printed via embosser.

Saxena et al. (2016) have provided a real-time solution (hardware and software) for helping blind people. They developed a Braille hand glove which helped in communication for sending and receiving messages in real time. Nam and Min (2016) have developed a music braille converted capable of converting the musical notations such as octaves, key signature, tie repeat, slur, time signature etc. successfully to Braille. Park et al. (2016) have suggested a method of automatic translation of scanned images of books to digital Braille books. They implemented character identification and recognized images in Books while automatically translating them to text. This method reduced time and cost required for producing books in Braille.

Alufaisan et al. (2021) designed an application that identifies Braille numerals in Arabic and converts it to plain text using CNN-based Residual Networks. The system also gave speech assistance to the generated text. Apu et al. (2021) proposed user and budget friendly braille device that can translate text and voice to braille for blind students. It works for different languages and converts based on ‘text’ or ‘voice’ command given by the user.

Yoo and Baek (2022) have proposed a device that can print braille documents for blind. They implemented a raspberry Pi camera to save documents as images stored in device. Characters were extracted from the images and converted to Braille which is then processed to output braille. Their proposed device was portable and could be created using 3D printing. Zhang et al. (2022) have used n-gram language model to

implement Chinese-braille inter translation system. This system integrates Chinese and Braille word segmentation with concatenation rules. They have also proposed an experimental plan to improve Braille word segmentation and concatenation rules with a word corpus of Chinese-Braille.

3 English to Bharti Braille Machine Translation System

3.1 Experimental Setup

For developing our machine translation system, we first collected the corpus for training our MT model. We gathered a large collection of text data (6 lac sentences) that were used for linguistic analysis and language modeling. This corpus has a structured set of texts collected and stored for manual vetting by human annotators. The entire process was accomplished using the following steps:

- Defining the corpus: first, we defined the scope and size of the corpus. This involved identifying the genre, and other characteristics of the texts viz categories of sentences like declarative sentences, interrogative sentences, Wh-questions etc. we collected sentences in tourism, health and administration domains.
- Collecting the data: Once the corpus was defined, Collection of data was started. This involved scraping texts from websites as well as obtaining data from existing sources such as books and newspapers.
- Cleaning and pre-processing the data: After the data was collected, it was cleaned and pre-processed to ensure that it has a usable format for linguistic analysis. This involved removing irrelevant data, standardizing formatting, and tokenizing the text into words or phrases.
- Annotating the corpus: Annotation involved adding metadata to the corpus, such as part-of-speech tags, named entities and multi-word expressions. This made the corpus more useful for our purpose.

Thus, through this process we collected 5,67,000 sentences with their translations. Out of this, 4,25,250 sentences were used for training the model and 85,050 sentences were used for validation and the remaining 56,700 sentences were used for testing the system.

3.2 Working of the System

The system was developed by constructing a pipeline for preprocessing. The first step towards this was to clean the data as in raw text, several abbreviations for the same word are used. For example, for kilometer, we found different variations like k.m., km, kms etc. these were processed and were all transformed into “kilometer”. Then, POS tagging was done. Next, named entities were identified in the text and then syntax analysis (analyze English sentences) was done. This entire process was done using Stanford CoreNLP library (Manning et al., 2014). The parser generated the English syntactic tree for an input sentence. The phrase structure syntactic tree sent to the transfer grammar engine which transformed the English structure into the Hindi equivalent structure.

The transfer grammar engine was developed as a rule-based system which had rules for conversion of English parse tree into Hindi parse tree. A snapshot of the rules of transfer grammar engine is shown in figure 1. A total of 843 rules were constructed which were used by the engine to transfer syntax using the various English constructs which were incorporated in corpus.

LINK	SOURCE_TREE	TARGET_TREE
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-ReCl-nx0e-Vpadjn-Vnx1	⊟nx-ReCl-nx0e-Vpadjn-Vnx1
1.S_r=2.S_r.~1.S=2.S.~1.NP_0=2.NP_0.~1.VPadjn=2.VP...	⊟nx0e-VPadjn-Vinf-s	⊟snx0e-Vinf
1.PP_r=2.PP_r.~1.Adv=2.Adv.~1.PP_f=2.PP_f.~	⊟ARBpx	⊟ARBpx
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-INF-nx0e-Vpadjn-Vnx1	⊟nx-INF-nx0e-Vpadjn-Vnx1
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.VPadjn=2.VPadjn.~1.ep...	nx0e-VPadjn-Vpx1	nx0e-px1-V
1.S_r=2.S_r.~1.S=2.S.~1.NP_0=2.NP_0.~1.eps=2.eps.~1...	⊟nx0e-VPadjn-Vinf-nx1nx2-s	⊟snx0e-nx1nx2-Vinf
1.S_r=2.S_r.~1.S=2.S.~1.S_f=2.S_f.~1.NP_0=2.NP_0.~1....	⊟nx0e-VPadjn-Vinf-nx1-s	⊟snx0e-nx1-Vinf
1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP.~1.NP...	nx0-VPadjn-Vnx1	nx0ne-nx1ko-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-nx0-VPadjnnx1	⊟nx-nx0ne-nx1ko-ADJ-vz1
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.NP_1=2.NP.~1.VPadj...	nx0e-VPadjn-Vnx1	nx0e-nx1ko-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-nx0-VPadjn-V	⊟nx-nx0ne-ADJ-vz1
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.VPadjn=2.VPadjn.~1.V...	nx0e-VPadjn-V	nx0e-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nxsnx0e-VPadjn-V	⊟nxsnx0e-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.N...	⊟nxsnx0e-VPadjn-Vnx1	⊟nxsnx0e-nx1-ADJ-vz1
1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP.~1.VP...	nx0-VPadjn-V	nx0ne-ADJ-vz1
1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP_0.~1.V...	nx0-VPadjn-Vnx1nx2	nx0ne-nx1ko-nx2-ADJ-vz1
1.VPadjn_r=2.VPadjn_r.~1.PP=2.PP.~1.VPadjn_f=2.VPadj...	⊟PxVPadjn	⊟nxPvxadjn
1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP_0.~1.V...	nx0-VPadjn-VP-AdjP	nx0-AdjP-V
1.S=2.S_r.~1.S_1=2.S_1.~1.Cond=2.Cond.~1.S_2=2.S_2.~	⊟S1CondS2	⊟S1CondS2
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.VPadjn=2.VPadjn.~1.ep...	nx0e-VPadjn-Vnx1nx2	nx0e-nx1ko-nx2-V
1.S_r=2.S_r.~1.NPadjn=2.NPadjn.~1.NP_0=2.NP.~1.NP...	nx0-VPadjn-Vnx1	nx0ne-nx1ko-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-nx0-VPadjnnx1	⊟nx-nx0ne-nx1ko-ADJ-vz1
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.NP_1=2.NP.~1.VPadj...	nx0e-VPadjn-Vnx1	nx0e-nx1ko-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nx-nx0-VPadjn-V	⊟nx-nx0ne-ADJ-vz1
1.S_r=2.S_r.~1.NP_0=2.NP_0.~1.VPadjn=2.VPadjn.~1.V...	nx0e-VPadjn-V	nx0e-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.NP...	⊟nxsnx0e-VPadjn-V	⊟nxsnx0e-ADJ-vz1
1.NP_r=2.NP_r.~1.NP_f=2.NP_f.~1.S=2.S.~1.NP_0=2.N...	⊟nxsnx0e-VPadjn-Vnx1	⊟nxsnx0e-nx1-ADJ-vz1

Figure 1. Snapshot of Transfer Grammar Rules.

After the syntax transfer of English sentence, we then translated the name entities using a name entity translation sub-module which translated organization names and transliterated the rest of the name entities. Figure 2 shows the complete workflow of the system. The system takes input from the user and then sends it for preprocessing (tokenized, normalized etc.) to pipeline. After this, Name entities from the input sentence was recognized, and lexical analysis was done where POS Tagging and parsing of the is done. The outcome of this step was a phrase structure tree of the input sentence. Here, the system also identifies the name entities in the input text. After source text analysis, the phrase structure tree of English was converted into the phrase structure tree of Hindi equivalent grammar. The transfer engine performed this task, by converting the parse of English, with subject and predicate (SVO) word order into the default Hindi syntax, which has SOV as its default word order.

LISP notation was used to generate the phrase structure tree. Table 1 shows the output of the example sentence, and the generated phrase structure tree is shown in figure 3. After this, the syntax of English was transformed into the syntax of Hindi. This was done using the transfer grammar engine. Phrase structure trees of both source (English) and target (Hindi) are shown in figures 4 and 5 respectively. Next, the given sentence was generated according to target syntax tree. Once this is done, the name entities identified earlier are translated into Hindi and are replaced by English name entities. Phrase structure tree of the same is shown in figure 6. Finally, the output generated at this point was sent to the NMT System for final translation. The output of the NMT system was

then sent to the post processing module to check for any known anomalies and accordingly rectify them. Finally the output text in Hindi is generated which is then sent to the Bharti Braille Translation Engine which generated the text in Bharti Braille format.

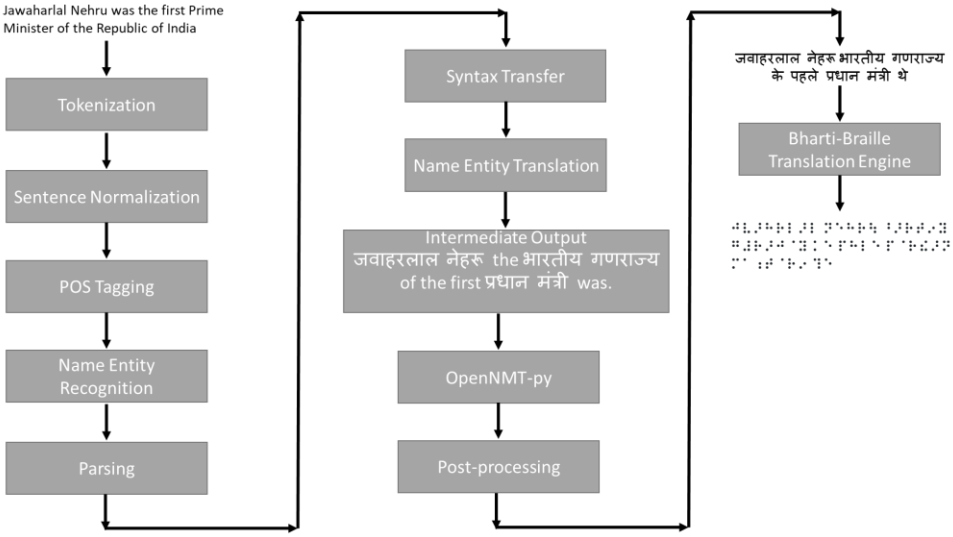

Figure 2. Working of the System.

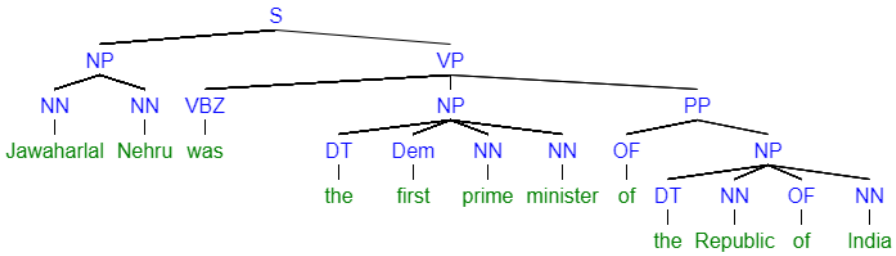

Figure 3. Phrase Structure Tree of English Input Sentence

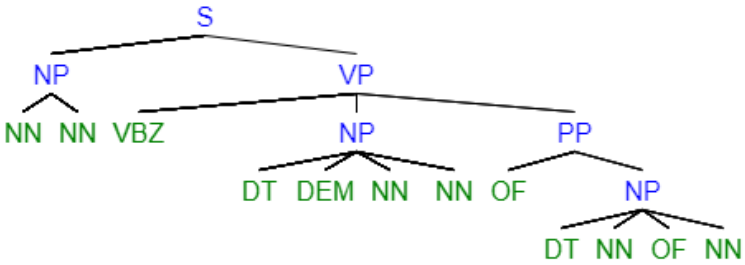

Figure 4. Phrase Structure Tree of English Sentence without Leaf Nodes

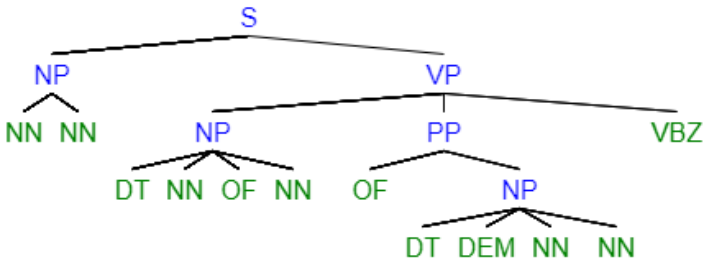

Figure 5. Hindi Equivalent Phrase Structure Tree Generated by Transfer Grammar Engine

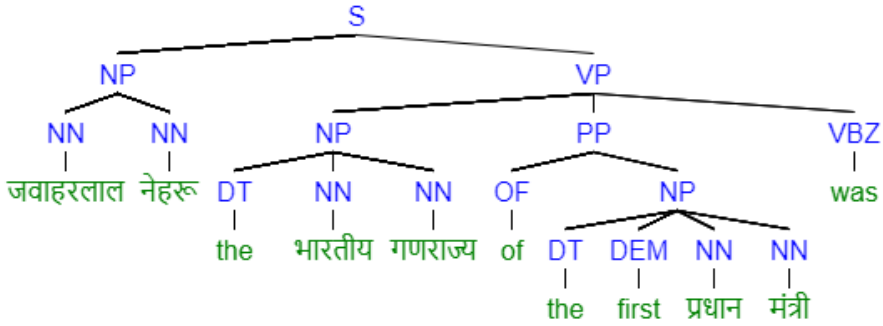

Figure 6. Name Entities Translated Phrase Structure Tree

Table 1. Example of MT Pipeline Workflow

Input Sentence	**Jawaharlal Nehru was the first prime minister of the Republic of India.**
POS Tagging	Jawaharlal/NN Nehru/NN was/VBZ the/DT first/DEM prime/NN minister/NN of/OF the/DT Republic/NN of/OF India/NN ./SYM
English Parse in LISP Notation	[S [NP [NN Jawaharlal] [NN Nehru]] [VP [VBZ was] [NP [DT the] [Dem first] [NN prime] [NN minister]] [PP [OF of] [NP [DT the] [NN Republic] [OF of] [NN India]]]]]
LISP Notation of English Phrase Structure Tree	[S [NP [NN] [NN]] [VP [VBZ] [NP [DT] [DEM] [NN] [NN]] [PP [OF] [NP [DT] [NN] [OF] [NN]]]]]
Transferred Phrase Structure Tree	[S [NP [NN] [NN]] [VP [NP [DT] [NN] [OF] [NN]] [PP [OF] [NP [DT] [DEM] [NN] [NN]]] [VBZ]
NER Translated Final Structure	[S [NP [NN जवाहरलाल] [NN नेहरू]] [VP [NP [DT the] [NN भारतीय] [NN गणराज्य]] [PP [OF of] [NP [DT the] [DEM first] [NN प्रधान] [NN मंत्री]]] [VBZ was]

In our experiment, the NMT model was trained using the process explained previously. The whole corpus was transformed using the preprocessed pipeline developed and discussed in section 3.2. An NMT model was trained using this preprocessed corpus. For the training, DGX system was used where 9 encoders, 9 decoders and 18 heads per layer were configured for training. We used 1536 feed forward layers and 1024 embedding layers. For testing the system, the same pipeline of section 3.2 was used which generated Hindi target translations for the given English input sentences. These Hindi translations were then sent to transfer engine for Bharti Braille conversion. Figure 2, shows the complete workflow. Once we got the intermediate translation, the same was given to the NMT model which generated the output. With little or no post processing the target sentence was produced which was then converted into Bharti Braille text.

4 Evaluation

The trained MT system was tested on 56700 sentences. BLEU metric was used for the evaluation of the system and compared it result with a base line system which did not have the pipeline discussed in section 3.2. In this baseline system, the English sentence was preprocessed and was directly sent to NMT system and then to the Bharti Braille system.

The name entity induced NMT model performed better than the baseline NMT model. The result of this study is shown in table 2, which shows system level scores where the individual translation scores were added and were divided by the total number of test sentences i.e. 56700. equation 1 shows the computation of system level BLEU metric. Here, BLEU-Scorei is the individual BLEU score for each sentence which are then added for all n sentences and are then divided by total number (n) of sentences.

$$System - level\ BLEU\ Score = \frac{\sum_{i=1}^{n} BLEU-Score_i}{n} \quad (1)$$

Table 2. Evaluation results of NMT models.

Language Pair	Baseline Models	Name Entity Induced NMT Models	Improvement
English-Hindi	0.4892	0.7543	0.2651

5 Conclusion and Future Work

In this paper we have shown the development of an NMT system for English-Bharti Braille Machine Translation. This is the first step towards providing access to the abundant knowledge in the world to the people who are visually impaired. We compared the results of our system with the baseline system and found that it produced better translations. Overall, we got a improved of 0.2651 using BLEU metric.

While doing qualitative evaluations we found that the system still lacks the capabilities of translating complex sentences. Thus, as an extension to this work, we would like to add more corpus for training and add more linguistic knowledge for the same.

Acknowledgements

This work is supported by the funding received from SERB, GoI through grant number CRG/2020/004246 for the project entitled, “Development of English to Bharti Braille Machine Assisted Translation System”.

References

1. Lewis, S., D'Andrea, F. M., & Rosenblum, L. P. (2012). The development of accepted performance items to demonstrate competence in literary braille. *Journal of Visual Impairment & Blindness*, *106*(4), 197-211.
2. Hong, S. (2012). An alternative option to dedicated braille notetakers for people with visual impairments: Universal technology for better access. *Journal of Visual Impairment & Blindness*, *106*(10), 650-655.
3. Herzberg, T. S., & Rosenblum, L. P. (2014). Print to braille: Preparation and accuracy of mathematics materials in K-12 education. *Journal of Visual Impairment & Blindness*, *108*(5), 355-367.
4. AlSalman, A., El-Zaart, A., Al-Salman, S., & Gumaei, A. (2012, May). A novel approach for Braille images segmentation. In *2012 International Conference on Multimedia Computing and Systems* (pp. 190-195). IEEE.
5. Authman, Z. I., & Jebr, Z. F. (2012). Arabic Braille scripts recognition and translation using image processing techniques. *Journal: Journal of College of Education*, *2*(3), 18-26.
6. Padmavathi, S., Reddy, S. S., & Meenakshy, D. (2013). Conversion of braille to text in English, Hindi and Tamil languages. *arXiv preprint arXiv:1307.2997*.
7. Hossain, S. A., Mahbub-ul-Islam, F. M., Azam, S., & Khan, A. I. (2013). Bangla braille adaptation. In *Technical Challenges and Design Issues in Bangla Language Processing* (pp. 16-34). IGI Global.
8. Abualkishik, A., & Omar, K. (2013, November). Framework for translating the Holy Quran and its reciting rules to Braille code. In *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)* (pp. 380-385). IEEE.
9. Oda, T., Sugano, A., Shimbo, M., Miura, K., Ohta, M., Matsuura, M., ... & Takaoka, Y. (2013). Improvement in Accuracy of Word Segmentation of a Web-Based Japanese-to-Braille Translation Program for Medical Information. *Journal of Communication and Computer*, *10*, 82-89.
10. Al-Salman, A. M. S., El-Zaart, A., Al-Suhaibani, Y., Al-Hokail, K., & Gumaei, A. (2014). Designing braille copier based on image processing techniques. *Int. J. Soft Comput. Eng. ISSN*, *4*(5), 62-69.
11. Yamaguchi, K., Suzuki, M., & Kanahori, T. (2014). Braille capability in accessible e-textbooks for math and science. In *Computers Helping People with Special Needs: 14th International Conference, ICCHP 2014, Paris, France, July 9-11, 2014, Proceedings, Part I 14* (pp. 557-563). Springer International Publishing.
12. Damit, D. S. A., Ani, A. I. C., Muhamad, A. I., Abbas, M. H., & Ali, F. Z. (2014, September). Dual braille code translator: Basic education tool for visually impaired children. In *2014*

International Conference on Computer, Communications, and Control Technology (I4CT) (pp. 399-402). IEEE.

13. Rupanagudi, S. R., Huddar, S., Bhat, V. G., Patil, S. S., & Bhaskar, M. K. (2014, January). Novel methodology for Kannada Braille to speech translation using image processing on FPGA. In *2014 International Conference on Advances in Electrical Engineering (ICAEE)* (pp. 1-6). IEEE.
14. Choudhary, T., Kulkarni, S., & Reddy, P. (2015, January). A Braille-based mobile communication and translation glove for deaf-blind people. In *2015 international conference on pervasive computing (ICPC)* (pp. 1-4). IEEE.
15. Guerra, C., Novillo, D., Alulema, D., Ortiz, H., Morocho, D., & Ibarra, A. (2015, October). Electromechanical prototype used for physical deployment of Braille characters for digital documents. In *2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* (pp. 191-198). IEEE.
16. Jariwala, N. B., & Patel, B. (2015, April). Transliteration of digital Gujarati text into printable Braille. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 572-577). IEEE.
17. Saxena, R., Mahajan, T., Sharma, P., & Sood, M. (2016, March). Braille hand glove—A real time translation and communication device. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1714-1718). IEEE.
18. Nam, Y. K., & Min, H. K. (2016). Design and Implementation of Music-To-Braille Translator. *Journal of rehabilitation welfare engineering & assistive technology*, *10*(3), 215-220.
19. Park, T., Jung, J., & Cho, J. (2016). A method for automatically translating print books into electronic Braille books. *Science China Information Sciences*, *59*, 1-14.
20. Alufaisan, S., Albur, W., Alsedrah, S., & Latif, G. (2021). Arabic Braille numeral recognition using convolutional neural networks. In *International Conference on Communication, Computing and Electronics Systems: Proceedings of ICCCES 2020* (pp. 87-101). Springer Singapore.
21. Apu, F. S., Joyti, F. I., Anik, M. A. U., Zobayer, M. W. U., Dey, A. K., & Sakhawat, S. (2021, July). Text and Voice to Braille Translator for Blind People. In *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)* (pp. 1-6). IEEE.
22. Yoo, D. H., & Baek, S. W. (2022). Developing of Text Translation and Display Devices via Braille Module. *Mathematical Statistician and Engineering Applications*, *71*(3), 537-547.
23. Zhang, J. X., Chen, H. F., Chen, B., Chen, B. Q., Zhong, J. H., & Zeng, X. Q. (2022). Design and Implementation of Chinese Common Braille Translation System Integrating Braille Word Segmentation and Concatenation Rules. *Computational Intelligence and Neuroscience*, *2022*.