INMT: Interactive Neural Machine Translation Prediction

Sebastin Santy[1] **Sandipan Dandapat**[2] **Monojit Choudhury**[1] **Kalika Bali**[1]
[1] Microsoft Research, Bangalore, India
[2] Microsoft R&D, Hyderabad, India
{t-sesan, sadandap, monojitc, kalikab}@microsoft.com

Abstract

In this paper, we demonstrate an Interactive Machine Translation interface, that assists human translators with on-the-fly hints and suggestions. This makes the end-to-end translation process faster, more efficient, and creates high-quality translations. We augment the OpenNMT backend with a mechanism to accept the user input and generate conditioned translations.[1][2]

1 Introduction

Machine Translation (MT) has witnessed several leaps of advancements and is now capable of producing human level translations (Läubli et al., 2018). However, a single source sentence can be translated into multiple forms, often varying in expression (Halliday, 1978), formality (Heylighen and Dewaele, 1999) or context. Several real-world documents such as books and news require such translations. Even with performance of state-of-the-art MT systems being at par with humans, current MT systems are useful only for information assimilation (Hutchins, 2009). Wider dissemination of translated content requires substantial manual post processing, and saves only a little time compared to a fully manual human translation. Combining the pros of both human and machine translation can potentially lead to streamlining of the translation process by assisting humans with producing high quality translations quicker.

This Machine Assisted Translation can be either done through Post Editing (PE) or Interactive Translation Prediction (ITP). In PE tools such as MateCat (Federico et al., 2014), the translator is provided with a complete translation and is asked to edit tokens which they feel are not appropriate. There are no suggestions provided by the machine beyond the initial gist. Interactive Translation started off with TransType (Langlais et al., 2000), which uses a rule-based translation system. With the introduction of Statistical Machine Translation, it became easier to provide richer phrase based suggestions, which led to creation of tools such as CASMACAT (Alabau et al., 2014)[3] and LILT[4]. Green et al. (2014) extensively researched the user experience of such systems and compared between manual and assisted translation using various metrics. The current method of constrained decoding, in particular coupled with the advent of Neural Machine Translation (NMT), was put forward by Wuebker et al. (2016) and Knowles and Koehn (2016). All these studies have shown that ITP provides improved translation quality compared to PE, and also suggest that human translators prefer ITP over PE. However, due to heavy resource (parallel data) requirements, the available ITP systems work only for a handful of resource-rich languages such as Spanish, Chinese, French and German.

In this paper, we present a proof-of-concept interactive translation system between English and five Indic languages (Bengali, Hindi, Malayalam, Tamil and Telugu) using state-of-the-art NMT models. As Indian languages are resource poor, we could use only 100-1500 thousand parallel sentences for training. There are prominent syntactic differences between Indic languages and English such as the basic word order, which requires substantial replanning of the translation suggestions in real time. These characteristics make ITP a challenging problem for English to/from Indic languages. To the best of our knowledge, this is the first ITP for Indic languages, and the only publicly available Neural ITP system.

[1] Screencast: https://youtu.be/DHan93R8d84
[2] Demo: inmt.southeastasia.cloudapp.azure.com/simple
[3] http://www.casmacat.eu/
[4] http://lilt.com

Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations), pages 103–108
Hong Kong, China, November 3 – 7, 2019.

2 Advantages of ITP

Interactive translation is beneficial to human translators and translation seekers alike:

• **Faster turnaround of document translations**
The gisting and suggestion (refer Interface Overview 5) helps the translator breeze through the translation task with minimal typing. Our preliminary study suggests that regular users use relatively very less number of keystrokes in ITP as compared to both manual typing and the PE process.

• **High quality translation due to human-machine interaction**
Language is inherently divergent and human translators cannot quickly enumerate all acceptable variants of a translation. On the other hand, in general, machine translation has not yet reached human quality though it can provide a number of variants. Combining the strengths of both of human and machine through interaction helps in getting higher quality translations compared to the individual processes. Another interesting usecase of interactive MT is in low resource settings where NMT is not able to churn out the most fluent translation. However, along with human help, this can lead to producing better translations, as compared to other mixed-initiative processes like post-editing often used in computer assisted translation tools.

• **Opens translation tasks to non-expert translators**
Expert human translators are often scarce for a large number of language pairs and are expensive to hire. In some countries (such as in the Indian subcontinent) it is easy to find multilingual speakers with native and near-native proficiency in multiple languages, and interactive MT systems enable non-expert translators to perform translation tasks through gisting and suggestions. The requirements of less proficiency also makes such a system more amenable to a non-expert large crowdsourced setting where the objective can be to gather more data for low resource languages.

3 Methods

3.1 Neural Machine Translation

Neural MT paradigms, starting from Seq2Seq (Bahdanau et al., 2014) to the current state-of-the-art Transformer-based architectures (Vaswani et al., 2017), use an encoder-decoder combination along with attention which helps in correct alignment of the tokens.

At time step t, the conditional probability of generating output token y_t given the full input sequence $\mathbf{x}$ and the previously output tokens $y_1, ..., y_{t-1}$ is:

$$p(y_t|y_1, ..., y_{t-1}, \mathbf{x}) = g(y_{t-1}, s_t, c_t) \quad (1)$$

where g is a non-linearity function and s_t and c_t are the hidden state and context vector, respectively. The vector c_t is a weighted average of all encoder hidden states with weights generated by the attention mechanism. We use sparsemax (Martins and Astudillo, 2016) attention instead of the usual softmax attention as it helps in focusing on specific source tokens similar to hard attention but still being differentiable. This is done in order to help with word coverage visualization.

3.2 Interactive Neural Machine Translation

In INMT, instead of conditioning the prediction of each token on the previous model predictions $\{y_1, ..., y_{t-1}\}$ (as is done above), we condition based on the partial input from the human translator $\{y'_1, ..., y'_{t-1}\}$. This results in a new conditional probability equation:

$$p(y_t|y'_1, ..., y'_{t-1}, \mathbf{x}) = g(y'_{t-1}, s_t, c_t) \quad (2)$$

For producing multiple suggestions based on the partial input, we rely on beam search decoding which features in all current NMT architectures. It selects the most probable full translation for a given input sentence. If and when the translator diverges from this full translation, a new beam search is conducted from the partial input prefix till end of sentence is encountered. A full beam search (till end of sentence) is done only for gisting. In case of suggestion, we do beam search of maximum length of 2. Beam search has been criticized for its lack of diversity (Gimpel et al., 2013), which is why, showing full sentence suggestions from beam search will present the translator with very similar suggestions. Beam search with length of 2 will produce bigrams with reasonable diversity.

3.3 Character-Level Search

We use a word level sequence modelling for NMT in this task, which does not allow partial words

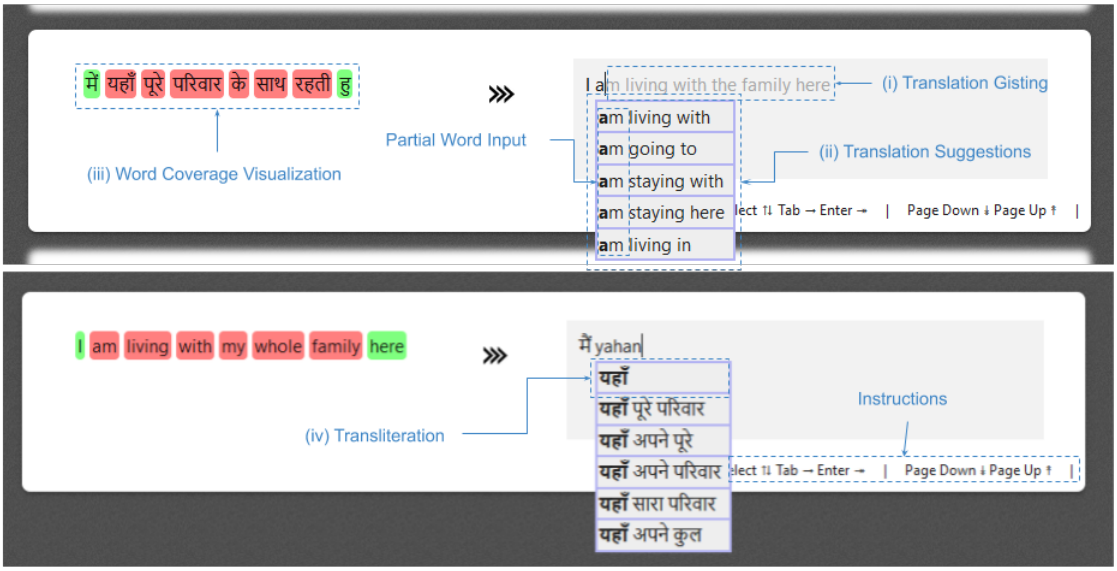

Figure 1: Example sentence being translated. (a) Hindi to English and (b) English to Hindi.

to be input inherently. We introduce two different techniques to help with character level suggestions:

• Start with Partial Word Prefix
During beam search, at each timestep each word in the vocabulary is assigned a score for k times, where k is the beam width. We mask the vocabulary set for words which only star with the partial word prefix input by the translator. This helps in getting top-k words which start with the intended prefix.

• Closest to Partial Word Prefix
We also devise a edit distance-based (Yujian and Bo, 2007) algorithm to rerank the beams which not only helps in suggesting sentences with partial word inputs but also helps in cases where the translator makes spelling mistakes for complex words. The latter algorithm already includes tokens which are limited by the former, but the former is faster due to lesser search complexity. We provide an option for the translator to select either one.

4 System Overview

We use OpenNMT (Klein et al., 2017), an open source neural machine translation toolkit to build the MT system.[5] We write a new interactive translation mechanism to accept the user input and do constrained decoding, which is plugged on top of this toolkit. This helps in keeping up with the state-of-the-art models and other updates released through the toolkit and still keeping the interaction functional. The primary advantage of using OpenNMT as our translation backend is that it is highly efficient, modular, extensible and well documented. In order to add new models (e.g. extending to new languages etc.) into this system, the models have to be trained based on OpenNMT instructions. If the target language uses a non-Latin script and users wish to use an Latin keyboard, the developer has to add an API to help with transliteration. In our case, we use the openly available Quillpad Indic Transliteration service for Indic languages.[6] There is also a default support for ISO romanizations (like ITRANS for Indic Languages) which can help with transliterations, though it might not be particularly convenient for non-Latin.

We use the Django framework (Python) for server purposes. The system makes use of the PyTorch (Paszke et al., 2017) version of OpenNMT for its ease of use. The interface, which is described below is made with JQuery (JavaScript) and communicates with the server through sockets - for interactive typing as they are faster and AJAX requests for all other calls.

5 Interface Overview

The interface is designed similar to MateCat (Federico et al., 2014) which is a open-source

[5] http://opennmt.net/

[6] http://www.quillpad.in

Word Coverage and Translation Gisting	Suggestions	Keystrokes
उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है Similarly , knowledge for mental health is necessary .	Similarly , In the The knowledge Thus , So the	↓ ↓ Enter ↩
उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है **In the** same way , knowledge of knowledge is essential for mental health	same way	Tab Tab Tab Tab
उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है **In the same way , knowledge** of knowledge is essential for mental health	of knowledge is essential is necessary for mental	i
उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है **In the same way , knowledge i**s essential for mental health	**i**s essential for **i**s necessary for **i**s required to	Enter ↩
उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है **In the same way , knowledge is essential for mental health**		Page ↓

Table 1: Translation workflow when translating from "उसी प्रकार मानसिक स्वास्थ्य के लिए ज्ञान की प्राप्ति आवश्यक है" to "In the same way , knowledge is essential for mental health". The bold black text shows the current state of user input, the gray text shows gisting and suggestions. 10 keystrokes are required when using interactive typing, whereas for manual typing 60 keystrokes will be required (Refer Keystroke Reduction 6.2)

freely available web-based post-editing translation software. In our interface, the user is able to upload documents in the format they choose, or just input free-flowing text. The user then has to select the source and target language. The next page will show tokenized sentences from the input corpus on the left with a corresponding right column where the user is expected to add the translation. Figure 1 shows how the interface looks. For each translation we use 2 different techniques to assist in the translation flow:

(i) Translation gisting

Gisting the user with a full sentence translation will prime the translator with a quick translation with very less cognitive load. Users have much less cognitive load when it comes to spotting errors in the gisting, than trying to mentally structure the translations. This accelerates the translator's initial time taken.

(ii) Translation suggestion

Gisting is not expected to always be what the translator ideally wants. This would mean that the translator needs to change the default gist which the translation engine provides. Based on the context of the previous sentence, the translation engine also sends a maximum of 5 suggestions from which the user can choose to select to move forward with the translation. If the user can find nothing from either the gist or suggestion, the translator can choose to type. The user still does not need to type the complete word, the suggestions are assigned a score based on the edit-distance between the partial input and the tokens in the decoding lattice. The suggestions are then re-ranked taking into account both the edit-distance score along with the beam score.

Furthermore, we provide mechanisms which allow increased throughput of translations as below:

(iii) Word Coverage Visualization

The standard NMT architectures use attention to correctly align the words from source to target. We use sparsemax attention rather than softmax, as it allows us to attend to a specific word in context while translating. This helps in an appropriate visualization of word coverage which the translator can use to their benefit to validate whether they have correctly translated the sentence (cf. Table 1).

(iv) Transliteration

Languages, except the European ones generally have a non-Latin script. Translators especially non-expert ones usually use English keyboards to type. Usually they are assisted with some browser plugin like Google input tools which helps them in transliterating whatever they type in English to the target language. Here they use English characters as phones rather than as actual English words. However, such plugins deteriorates the experience when it comes to such

an interactive interface, where each character is used for translation and such post-replacement of English to target language cannot work properly. For experimentation, we use an openly available transliteration API (Quillpad) for Indic Languages.

Through multiple design iterations and feedback from our pre-pilot study, it was observed that certain keyboard commands are naturally helpful for the translator to breeze through translations. Options available are:

- Tab : To get the next word from the selection
- Enter ↩ : To get all the words from the selection.
- ↑ ↓ : To alter the selection between multiple suggestions.
- Page ↑ Page ↓ : To traverse from one sentence to another.
- End : To end the translation process and download the translated document.

The system is designed in such a way that translators never need to take their hands off from the keyboard. This helps in faster typing and selection of suggestions.

6 Experiments

We use parallel corpus from OPUS (Tiedemann, 2012) to build our NMT model for 5 different Indic Languages and English (en). These 5 Indic Languages include Bengali (bn), Hindi (hi), Malayalam (ml), Tamil (ta) and Telugu (te). The training data from OPUS contain 100-1500 thousand parallel sentences each for the mentioned Indic languages. All our models have been tested on the Indic Language Dataset (Post et al., 2012)[7] which contains around 1000 parallel sentences with each sentence having 4 English references. This helps us measure multi-reference BLEU score which is often useful for relatively free word order languages.

We take inspiration from Aharoni et al. (2019) to build a multilingual many-to-one and one-to-many indic neural MT system based on the state of the art transformer architecture (Vaswani et al., 2017). We use the recommended set of parameters for transformers from Vaswani et al. (2017) to get comparable results. We conduct multiple simulation experiments to show the efficacy of our system.

6.1 BLEU Score Analysis

BLEU (Papineni et al., 2002) scores are the current standard to evaluate MT performance. We measure the BLEU score of the generated gist after a certain fraction - x% of words of the intended translation has been provided. Table 2 shows the average BLEU score for each language pair at different values of x. As we can see, after 40% of the sentence has been provided by the user, rest of the gist is almost always perfectly correct.

	Data Size	0%	10%	20%	40%
bn-en	1.1M	25.31	27.54	35.68	54.03
hi-en	1.5M	40.64	42.06	47.90	62.18
ml-en	897K	19.76	21.95	29.84	49.88
ta-en	428K	18.71	20.90	27.05	44.55
te-en	104K	11.92	14.57	21.17	41.98

Table 2: Multi-BLEU Score with x% of partial input

The difference in performance for different languages is presumably because of the amount of resources available for each of them.

6.2 Keystroke Reduction

Keystroke Reduction algorithmically compares the minimum number of keystrokes required when typing interactively versus the same when manually typing. Interactive typing accounts for the keystrokes made when navigating through like ↑, ↓, Tab, Enter ↩ each of which is one keystroke, whereas for manual typing, number of keystrokes is determined by the number of characters in the sentence. This allows us to get an approximate idea of how many keystrokes are being saved while using interactive typing. We observe a reduction of around 30% keystrokes for all the above mentioned languages.

7 Conclusion and Future Work

The system currently provides an interface for interactive machine translation with a latency of less than 0.5 seconds. However, a real-time interactive application should focus on keeping latency to the

[7] The widely used WMT testset is only available for Hindi. Thus, to make the testset uniform across language we do not use WMT testset for our experiments.

minimum. Network calls are the primary latency bottleneck for web-based applications. We are currently working on adding support for front-end deep learning frameworks such as TensorflowJS to do heavy lifting for network intensive components like dropdown suggestions on client side.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. *arXiv preprint arXiv:1903.00089*.

Vicent Alabau et al. 2014. Casmacat: A computer-assisted translation workbench. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–28.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marcello Federico et al. 2014. The matecat tool. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 129–132.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111.

Spence Green, Jason Chuang, Jeffrey Heer, and Christopher D Manning. 2014. Predictive translation memory: A mixed-initiative system for human language translation. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 177–187. ACM.

Michael Alexander Kirkwood Halliday. 1978. *Language as social semiotic: The social interpretation of language and meaning*. Hodder Arnold.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center "Leo Apostel", Vrije Universiteit Brüssel*.

John Hutchins. 2009. Multiple uses of machine translation and computerised translation tools. *Machine Translation*, pages 13–20.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Proceedings of the Association for Machine Translation in the Americas*, pages 107–120.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. Transtype: a computer-aided translation typing system. In *ANLP-NAACL 2000 Workshop: Embedded Machine Translation Systems*.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? a case for document-level evaluation. *arXiv preprint arXiv:1808.07048*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke et al. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 66–75.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.