# Constrained Decoding for Technical Term Retention in English-Hindi MT

**Niyati Bafna***, **Martin Vastl***, **Ondřej Bojar**
Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics
64780815@o365.cuni.cz, martin.vastl2@gmail.com, bojar@ufal.mff.cuni.cz

## Abstract

Technical terms may require special handling when the target audience is bilingual, depending on the cultural and educational norms of the society in question. In particular, certain translation scenarios may require "term retention" i.e. preserving of the source language technical terms in the target language output to produce a fluent and comprehensible code-switched sentence. We show that a standard transformer-based machine translation model can be adapted easily to perform this task with little or no damage to the general quality of its output. We present an English-to-Hindi model that is trained to obey a "retain" signal, i.e. it can perform the required code-switching on a list of terms, possibly unseen, provided at runtime. We perform automatic evaluation using BLEU as well as F1 metrics on the list of retained terms; we also collect manual judgments on the quality of the output sentences.

## 1 Introduction and Motivation

It is common for bilingual or multilingual speakers to borrow technical terms from other, usually high resource, languages into their native language. This may be for several reasons, e.g. the technical term in the high resource language may be much more popular and therefore better understood, or the required term may simply not exist in the language in question. This is very common, for example, in Indian languages, where the language of education is frequently different from the regional native language.

We can imagine, therefore, a scenario which requires the automatic translation of text or speech, with the constraint that a given list of English domain words appear untranslated in the Hindi output. Essentially, this can be seen as a special case of constrained decoding with a given source-target terminology. We make the assumption that the user knows the terms to be retained at run time, and can provide this information to the system before translating the sentence.[2]

## 2 Previous Work

The idea of constrained decoding has been recognized as useful in several works (Hokamp and Liu, 2017; Chatterjee et al., 2017; Hasler et al., 2018; Dinu et al., 2019; Jon et al., 2021). Usually, the constraints are in the form of a terminology list, as in the above works. To our knowledge, this is the first study on combining this concept with introducing code-switching[3] (CS) into the output for a multilingual educational or technical setting.

## 3 Approach

We set up an end-to-end supervised learning scenario aimed at teaching the model to perform term retention. The basic idea is to train a machine translation model to obey a "signal", that we can then provide at run time on selected words. It is easy to see that such a model (the "tagged" model) would be independent of domain and could in theory perform term retention on any term for which the signal was provided. We also train a simple baseline for comparison; the baseline model sees the same training data as the tagged model, but does not receive any signal that would be highlighting the terms to retain. Therefore, given input at run time, it must rely on past exposure on the specific terms and their (non-)translation to perform term retention.

We provide the mentioned signal in the form of tags i.e. *<REW>* and *</REW>* tags (standing

*Equal contribution by these authors.

[2] We do not, however, assume that we have this information while training, since it would be expensive and unviable to retrain such a model every time for a new setting and/or new domain vocabulary. In this study, we work with English-Hindi MT.

[3] Linguists sometimes make a difference between the terms code-switching and code-mixing; in this paper, they are used interchangeably.

**Source sentence**: *You need to install these Python libraries.*
**Term list**: Python, libraries
**Input to the system**: *You need to install these <REW> Python </REW> <REW> libraries </REW>*
**Desired output**: आपको इन **Python libraries** को स्थापित करना चाहिय

Figure 1: Example input to and desired output from the system

| Dataset | Total sentences | Sentences with CS |
|---|---|---|
| Train | 250 700 | 123 274 |
| Development | 10 247 | 5 064 |
| Test seen | 5 000 | 5 000 |
| Test unseen | 768 | 768 |
| Test w/o CS | 500 | 0 |

Table 1: Types of datasets. CS Sentences: sentences with introduced code-switching. "Test seen": sentences with terminology that were all seen during the training, "Test unseen": sentences with terminology that were never seen during the training as retained words. Test w/o CS: sentences with no terminology constraints.

for "retained English word") to indicate that the enclosed term shall be retained during the translation, see Figure 1. This approach can be used in any type of transformer-based translation system and therefore can be implemented with little to no effort in current systems.

## 4 Synthetic Data Creation

We used HindEnCorp 0.5 (Bojar et al., 2014) data set and we split it into multiple parts as seen in Table 1. We adapt pre-existing English-Hindi parallel data so that it manifests term retention on the target while remaining coherent and grammatical. We leverage the fact that our parallel corpus already contains many instances of simple transliteration equivalents, such as names of people, places, organizations, etc. We thus interpret the target sentence as "retaining" the transliterated word, while being perfectly grammatical.[4]

### 4.1 Identifying Transliterations

Given the parallel corpus, we need to identify pairs of transliterated words in each English-Hindi sentence pair. We first find the word level alignments[5] in source-target pairs, using GIZA++ (Och and Ney, 2003). Then for each aligned word pair, we check for transliteration using a normalized edit distance threshold.[6] We define our normalized edit distance as:

$$NED(s,t) = \frac{edit\_distance(s,t)}{max(length(s), length(t))}$$

calculated between the English word and the Hindi word transliterated into Latin script.[7] Eyeballing the resulting pairs, we see that the alignment step along with this threshold results in near perfect accuracy. This method gives us a total of 269095 transliteration pairs in the whole corpus.

Once a transliteration pair is identified in the training corpus, we simply replace the target side Devanagari word with the Latin-script source word, resulting in an instance of term retention. The original sentence pair is no longer used in the training of the tagged model.

## 5 Model

We used a transformer-based model (Vaswani et al., 2017) with vocabulary size of 32000 tokens and with hyperparameters as described in The University of Edinburgh's Neural MT Systems for WMT17 (Sennrich et al., 2017) for both of our models. We used MarianMT framework (Junczys-Dowmunt et al., 2018) to train the models; we let the model train until the BLEU score (Papineni et al., 2002) did not improve on the development set for 5 epochs. We then selected the model with the highest BLEU score as the model used for later experiments. The change of BLEU score on the

[4] Although more sophisticated approaches to synthetic code-switched data creation may be better suited for other tasks, we find that this approach is sufficient for our needs. This may be because term retention is in fact required to be performed on similar words i.e. named entities or domain terms that behave similarly to named entities.

[5] The idea is that the target transliterated word must "come from" or be aligned with the source word, assuming a correct word alignment.

[6] We use a Python transliteration tool `https://pypi.org/project/indic-transliteration/`

[7] The threshold was tuned over a small subset of the Xlit-Crowd: Hindi-English Transliteration Corpus (Khapra et al., 2014): using this corpus, we found the edit distance between the English source words and the "true" transliterations which were back-transliterated into Latin script. For the final experiment, we used the threshold of 0.5.

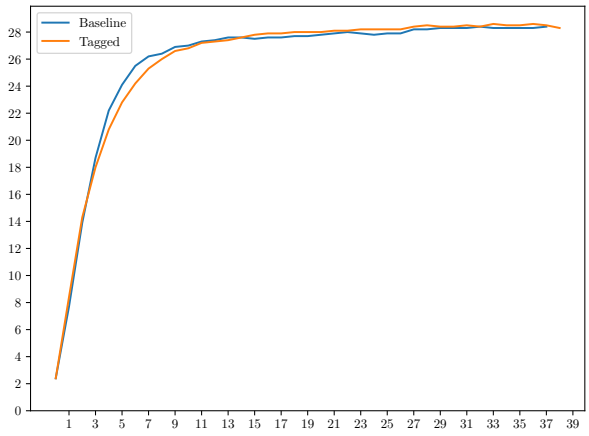


Figure 2: BLEU score on development set per epoch

| Model | Seen | Unseen | Without CS |
|---|---|---|---|
| Baseline | **28.7** | 16.8 | **22.4** |
| Tagged | 27.2 | **17.5** | 21.9 |

Table 2: BLEU score on test set

development set per epoch is in Figure 2. It can be seen that the BLEU scores for both of the models are comparable and they train for a similar number of epochs.

## 6 Automatic Evaluation

There are two components of model performance:

- Retention of marked terminology
- Overall coherence and fluency

For the former, we calculate precision, recall, and F1 over the gold retained set of words and the set of retained terms in the output. Our evaluation script compares the system output with the list of terms that should be untranslated in the given sentence. Precision is the ratio of term occurrences in the system output that were anticipated in the reference, out of all produced Latin terms. Recall is the ratio of term occurrences produced by the system out of all term occurrences anticipated by the reference. For the latter, we use BLEU score.

The BLEU scores on test sets can be seen in Table 2. The baseline model is slightly better on the seen test set, while the tagged approach outperforms the baseline model on the unseen test set. On the "Without CS" test, the baseline model still (incorrectly) produces English; however, while the tagged model does not do this,

| Model | Precision | Recall | Micro F1 |
|---|---|---|---|
| Baseline | 0.43 | 0.63 | 0.51 |
| Tagged | **0.88** | **0.88** | **0.88** |

Table 3: Retention results on seen test set

| Model | Precision | Recall | Micro F1 |
|---|---|---|---|
| Baseline | 0.08 | 0.25 | 0.13 |
| Tagged | **0.51** | **0.85** | **0.64** |

Table 4: Retention results on unseen test set

it often produces different and sometimes incorrect Hindi phrasing for these words as compared to the reference, resulting in an overall lower BLEU score. A possible explanation for this observation is that the tagged model has to learn to use the given signal at proper places which can damage its performance. On the other hand on the unseen dataset, the tagged model receives explicit information to retain the term and therefore outperforms the baseline model. Results for the retention metric can be seen in Table 3 and Table 4.

It can be seen that the tagged approach outperforms the baseline model on both the unseen and seen test set, demonstrating that it indeed learns to obey the provided signal, instead of simply relying on previous exposure as the baseline does.[8]

## 7 Manual Evaluation

We also performed a manual evaluation to complement the BLEU score. This evaluation was solely for the purpose of judging the quality of the final output regardless of whether the model managed to retain the required words or not.

### 7.1 Design

We provide the annotators with the *spoken* form of the candidate translation, rather than asking them to read the script-mixed output. There are two reasons for this: (1) we do not want the annotators to be affected by seeing or not seeing Latin script, (2) the spoken form is the more natural setting in which code mixing occurs.

[8]Note that the drop in performance of the tagged model in the unseen test F1 score indicates that it is not wholly independent as yet of the terminology it has been exposed to.

Further, in order to ensure blind evaluation of the Baseline vs. Tagged system, we needed to control for the fact that the Tagged system has a higher tendency to retain words in the Latin script. Since the user may be unfairly biased one way or the other when judging between sentences with different numbers of code-switched words, we decided to select the test sentences in a controlled manner, depending on the number and nature of XXX~~English~~Latin-spelled (i.e. English) words in the output.

The test set partitions are listed as columns in Tables 5 and 6: "Same # of En words" is the group of test sentences where the Baseline and Tagged translated outputs have the same number of English terms, thus controlling for bias for or against a translation simply because it has more English. In total, there were 5 such sentences each scored 3 times, so we collected 15 judgements on this partition. For instance in Table 5, we see that the tagged model was selected as better by 7 judgements and in 4 cases, it tied with the baseline. "Same set of En words" takes this a step further: it is the group of sentences where both model outputs have exactly the same English words in them; of course, they may (and do) differ in the rest of the sentence structure, Hindi wording, etc. Note that selecting sentences with a comparable number of terms English in them as we do results in an inherent advantage for the baseline model: since the baseline model can code-switch when it chooses rather than according to an external signal, it is more likely to choose convenient situations with globally better translations. This is the reason for the "Random" test set (the last column in Tables 5 and 6); i.e. sentences picked randomly, regardless the output of each system, which are intended to judge the average quality of the baseline and tagged against each other, even though these judgments are vulnerable to the biases discussed above.

In the manual evaluation, we gave 3 native Hindi speakers, also fluent in English, the source text and recordings of the translations. The goal of the annotation was a three-way judgment: whether the first translation was better, the second was better, or both were equivalent in quality.

### 7.2 Results and Analysis

Our manual test set covers a total of only 26 sentences, split equally between outputs from the seen and unseen test sets;[9] it is intended more for giving a qualitative sense of the comparison. Broadly, the evaluators considered the tagged outputs roughly comparable to the baseline in terms of coherence and quality, see Tables 5 and 6. Across both test sets, the Baseline model outputs were considered better 33% of the time (26 of 78 judgments), the Tagged model outputs were considered better 32% of the time (25 judgments), and the outputs were considered roughly equivalent in quality in the remaining 35% of the judgments.

We investigated the following questions:

- Do the models perform better on seen words than on unseen words?

| | Same # of En words | Same set of En words | Random | ∑ |
|---|---|---|---|---|
| Baseline | 4 | 5 | 3 | 12 |
| Tagged | 7 | 4 | 1 | 12 |
| Equal | 4 | 6 | 5 | 15 |
| ∑ | 15 | 15 | 9 | 39 |

Table 5: Manual test judgments for seen test set. Overall, the set contains 13 sentences from the seen test set, leading to the total of 39 judgments over 3 annotators. For example, we had 3 sentences (and therefore 9 total judgments) in the randomly selected group of sentences ("Random"); of these 9 judgments, 4 preferred the baseline model, 1 the tagged model, and 5 judgments saw the baseline and tagged outputs as equal in terms of overall quality.

| | Same # of En words | Same set of En words | Random | ∑ |
|---|---|---|---|---|
| Baseline | 3 | 8 | 3 | 14 |
| Tagged | 7 | 4 | 2 | 13 |
| Equal | 5 | 3 | 4 | 12 |
| ∑ | 15 | 15 | 9 | 39 |

Table 6: Manual test judgments for unseen test set. This test set again contains 13 sentences from the unseen test set, so a total of 39 judgments over 3 annotators is collected. The columns have the same meaning as in Table 5.

[9] This is because of the demanding procedure involving sentence recordings.

In the manual evaluation, we observed that the models dip in fluency around the segments with introduced English words. For example, there is a lack of syntactic agreement, or the model loses the thread of the sentence.

Tagged: *आवश्यक packages हटाया जाएगा।
(*Essential packages will be$_{sg}$ removed$_{sg}$)

In this example, we need the plural inflection of the verb phrase "हटाया जाएगा।" (will be removed). We see these instances both in the seen and unseen test sets; however, on the whole, the models are able to keep track of the source sentences a little better with the seen test set.

- Why does tagged do better than baseline in sentences where the same number of English words was produced in the output?

  The baseline model is worse at retaining fluency around code-switched words, especially in the unseen test set. While the tagged model also shows this tendency, it manages to translate the shorter instances correctly. With longer sentences, it is performing equally bad, especially in the unseen test set.

The "random" test set is intended to take a look at the average outputs of the models, not controlled for the number of English words in them. Here, the models perform similarly, but users differ in their preferences regarding the presence of English words.[10] Overall, the qualitative assessment yields that the tagged model performs on par with the baseline with respect to fluency, and of course much better at the retention task.

## 8 Conclusion

The task of applying terminology constraints while dealing with code-switched text seems especially important in current multilingual educational and other settings. We present a simple technique that can adapt a vanilla transformer-based MT tool for performing this task, by synthesizing training data that exhibits term retention. We demonstrate that our model performs well on unseen terminology, and that its general translation quality is not damaged. Future research should consider using code-switched parallel corpora, either for training or fine-tuning, in order to teach the models the various nuances of natural human code-mixing.

[10] For example, in a sentence that only differs in the fact that a word is in English in the first sentence and in the Hindi form in the second sentence, annotators apply their preferences.


## Acknowledgements

This project has received funding from the grants H2020-ICT-2018-2-825303 (Bergamot) of the European Union and 19-26934X (NEUREM3) of the Czech Science Foundation. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and datasets come from the Lindat Repository supported by the Ministry of Education, Youth and Sports of the Czech Republic, Project No. LM2018101 LINDAT/CLARIAH-CZ.


## References


Ondřej Bojar, Vojtěch Diatka, Pavel Straňák, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp 0.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. *arXiv preprint arXiv:1906.01105*.

Eva Hasler, Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. *arXiv preprint arXiv:1805.03750*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *CoRR*, abs/1704.07138.

Josef Jon, João Paulo Aires, Dusan Varis, and Ondrej Bojar. 2021. End-to-end lexically constrained machine translation for morphologically rich languages. *CoRR*, abs/2106.12398.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast

neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Mitesh M Khapra, Ananthakrishnan Ramanathan, Anoop Kunchukuttan, Karthik Visweswariah, and Pushpak Bhattacharyya. 2014. When transliteration met crowdsourcing: An empirical study of transliteration via crowdsourcing using efficient, non-redundant and fair quality control. In *LREC*, pages 196–202. Citeseer.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.