Study of Encoder-Decoder Architectures for Code-Mix Search Query Translation

Mandar Kulkarni, Soumya Chennabasavaraj, Nikesh Garera

Flipkart Data Science

India

(mandar.kulkarni,soumya.cb,nikesh.garera)@flipkart.com

ABSTRACT

With the broad reach of the internet and smartphones, e-commerce platforms have an increasingly diversified user base. Since native language users are not conversant in English, their preferred browsing mode is their regional language or a combination of their regional language and English. From our recent study on the query data, we noticed that many of the queries we receive are code-mix, specifically Hinglish i.e. queries with one or more Hindi words written in English (Latin) script. We propose a transformer-based approach for code-mix query translation to enable users to search with these queries. We demonstrate the effectiveness of pre-trained encoder-decoder models trained on a large corpus of the unlabeled English text for this task. Using generic domain translation models, we created a pseudo-labelled dataset for training the model on the search queries and verified the effectiveness of various data augmentation techniques. Further, to reduce the latency of the model, we use knowledge distillation and weight quantization. Effectiveness of the proposed method has been validated through experimental evaluations and A/B testing. The model is currently live on Flipkart app and website, serving millions of queries.

KEYWORDS

transformers, query translation

1 INTRODUCTION

With the wide reach of the internet and smartphones, e-commerce platforms have an increasingly diversified user base. Due to the language barrier, native language speakers feel comfortable interacting with the platform in the language of their choice. Specifically in India, due to the relatively low percentage of the English-speaking population, the vernacular platform capability is essential to provide a reliable and convenient shopping experience to such users.

The analysis of the in-house query data showed that, apart from English, the major portion of the queries (around 21%) seen on the platform are code-mixed Hinglish i.e. queries where one or

more Hindi words are written in Latin script. This could be because of the large Hindi-speaking population in India. To utilize traditional search workflows for English with Hinglish queries, it was important to translate Hinglish queries to English.

Code-mix translation has its unique challenges. One of the major challenges is the unavailability of a parallel training corpus. A well-known approach for improving the translation model with large target-side monolingual data is back-translation [12]. However, such an approach is not feasible with code-mix data since there is no publicly available model which can reliably translate from English to Hinglish. On top of it, for search query translation, factors such as spell errors in the input queries, ambiguity in language identification, non-standard query format, and free-flowing text pose additional challenges. For some queries, we also encounter articulation gap issues i.e. the user is not able to correctly articulate the query to search the product e.g. user inputting a query 'juta bina dori wala' where the intended product is 'shoe without lace'. In such cases, literal translation may not provide good search results. Table 1 shows examples and issues associated with Hinglish queries.

Hinglish Query	Issue
semsung phone 8 hajar vala	Brand mispelling
redami 5a ki battery	Brand mispelling
laptop ke barrte	Spell errors
makhamal wala seutar	Spell errors
juta bina dori wala	Articulation Gap
baal hawa machine	Articulation Gap
teen char sau wali saree sadi	Free flowing text
kam keemat mein four g mobile	Free flowing text

Table 1: Various issues associated with Hinglish queries.

In this paper, we propose a transformer-based approach for code-mix query translation. We propose to use the pre-trained encoder-decoder models such as T5, and BART as the base model and further fine-tune it on the in-domain query corpus. As opposed to other works on code-mix data such as [6], which proposes to use multilingual pre-trained models, we experiment with models trained only on English text. A parallel corpus for the training is created with monolingual Hindi and Hinglish queries. To detect Hinglish queries, we developed a ML-based approach. We use translation and transliteration APIs for quick and in-expensive labeling with unlabeled queries. We observe that translations on the search queries are noisy, possibly due to domain mismatch. To make the translation robust against noise and issues such as spell errors, we experiment

with different data augmentation schemes, and the model is trained using a combination of supervised and data augmentation objectives. To mitigate issues with the articulation gap, we use a small set of manually tagged Hinglish queries. To reduce the latency of the model in production, we use Knowledge Distillation (KD) and Weight Quantization (WQ). Experimental results under various settings demonstrate that our approach can provide reliable translations for the queries. Post successful results with AB testing, the model is currently live in production, serving millions of queries. The main contributions of the paper are as follows.

- Validating the effectiveness of English pre-trained models and data augmentations for code-mix search query translation which achieves 25 BLEU points improvement over generic in-house models and third-party APIs
- Developing a Knowledge Distillation (KD) approach for faster inference which achieves more than 4x reduction in latency with a small drop in the accuracy
- Developing Hinglish language detection approach to identify code-mix queries

2 RELATED WORK

Transformers [14] are widely used for seq2seq applications such as translation. Warm starting the transformer training with pre-trained checkpoints proved to be an effective method for improving the translation model. Rothe et.al [11] extensively studied the effectiveness of such a method for a wide variety of Natural Language Processing (NLP) tasks. It was observed that, for encoder-decoder models, good results are observed when the encoder and the decoder share the vocabulary.

In a recent work on code-mix translation, Jawahar et al. [6] explored the use of multilingual pre-trained encoder-decoder models such as mBART and mT5 for translation from English to Hinglish. On the contrary, in our work, we explore the use of pre-trained models trained only on English text.

Due to the lack of parallel corpus for training, few of the earlier works attempted to use existing MT systems for code-mix translation. Dhar et al. [3] explored augmenting code-mix data for use with existing MT systems such as Google Translate. Srivastava et al. [13] also explored the use of existing MT systems such as Google Translate and Bing for translating code-mix text. They also propose a partial translation approach based on token language labels along with MT systems.

3 EXPLORING PRE-TRAINED CHECKPOINTS FOR QUERY TRANSLATION

Pre-trained transformer models are proved to be effective in a wide variety of Natural Language Processing (NLP) applications. For translating code-mix Hinglish queries to English, we experiment with different pre-trained encoder-decoder models and further fine-tune these models on the in-domain query data. The model is fine-tuned with the combination of two loss components as shown in Eq. 1. $Loss_s$ corresponds to the cross-entropy loss for the labeled set of queries while the $Loss_d$ corresponds to the cross-entropy loss for the data augmentation.

$$L = (1 - \lambda)\ Loss_s + \lambda\ Loss_d \qquad (1)$$

where λ indicates the weighting factor for the supervised and the data augmentation loss.

In the following sections, we describe the details of the training data, models, and data augmentations techniques experimented with for the training.

3.1 Data preparation

As noted earlier, one of the major challenges with code-mix translation is the unavailability of parallel corpus for training. To identify the pre-trained model best suited for the code-mix translation task, for the initial set of experiments, we created a parallel training set of 200k queries. We used monolingual Hindi query data and transliterated it to English (to get Hinglish) and translated it to English (to get the English target label). For translation and transliteration of the monolingual data, we use generic domain third-party APIs and also observe comparable results of the same with generic in-house models in 8.2.

3.2 Test data preparation

For evaluation, we created a parallel test set of 15k manually tagged Hinglish queries. The test set is formed based on the query impressions (repetitions) i.e. the test set consists of approximately equal proportions of Head, Torso, and Tail queries. The same test dataset is used for all the experimental evaluations in the paper.

3.3 Pre-trained models

For query translation use-case, we explored three pre-trained encoder-decoder models.

3.3.1 T5. Text-To-Text Transfer Transformer (T5) [10] is the transformer encoder-decoder model trained in a self-supervised manner on a large corpus of unlabeled English text. We use the T5-base model which has 222M trainable parameters and 12 layers in the encoder and decoder. For our application, since T5 is fine-tuned on a single task, we do not add any pre-text to the input queries during training/inference. The model is fine-tuned for 3 epochs using AdamW optimizer with a learning rate of 1e-5 and batch size of 32.

3.3.2 BART. BART [8] is a transformer model trained using denoising autoencoder objective. We use the Bart-base model which has 139M trainable parameters and 6 layers in the encoder and decoder. The model is fine-tuned for 3 epochs using AdamW optimizer with a learning rate of 5e-6 and batch size of 64.

3.3.3 Bert2Bert. Rothe et al. [11] demonstrated the effectiveness of pre-trained checkpoints for sequence generation. We use pre-trained bert-base-uncased [2] model as the encoder and the decoder since it showed good performance for translation task [11]. Since the BERT is inherently an encoder model while using it as the decoder, additional cross attention weights are introduced and these are initialized randomly. Therefore, the encoder and the decoder both have 12 layers. The model is fine-tuned for 3 epochs using AdamW optimizer with a learning rate of 5e-6 and batch size of 32.

For all the models, we use label smoothing during the training since it is shown to provide robustness to the label noise [14]. The label smoothing parameter is set to 0.1 for all the experiments. During the inference, we use beam search decoding where the

Setting	BART-base	T5-base	Bert2Bert
Baseline	31.88	32.8	19.88
+AutoEncoder	+1.37	+1.13	+7.13
+DropChar	+1.32	+1.14	+1.8
+Masking	+1.14	+1.43	+5.02
+WordOrderPermute	-3.1	-7.4	+5.6

Table 2: Comparison result for models and data augmentations. Baseline result indicates BLEU scores with pre-trained models without using data augmentation. Rows 2-5 indicate a change in the Baseline BLEU score when a particular data augmentation is used during the training.

beam size is set to 3. All the experiments were carried out on A100 GPU.

3.4 Data Augmentation

We have experimented with the following data augmentation techniques.

3.4.1 **Masking**. A word/token masking has proved be an effective self-supervised training strategy [8] [10]. Since search queries are short in length, we randomly mask a single word from the target (English) and train the model to reconstruct the entire target. For the BART and BERT2BERT models, a randomly selected word is replaced with the '[MASK]' token while for T5, it is replaced with '<extra_id_0>' token.

3.4.2 **AutoEncoder**. We experiment with Autoencoder (AE) as the self-supervised data augmentation objective. For this, we use target data as the input and train the model to reconstruct it. The intuition behind it is that we want to teach the model to reproduce English words in the query as it is. AE techniques have been experimented with NMT [1], however, authors have used an additional target-to-source model to auto-encode the source sentence. In our work, we only use a single source-to-target model for the AE objective.

3.4.3 **DropChar**. Since non-English speakers attempt to spell the words based on the phoneme sound of it, we observe that typically the first and last character of the word is spelled correctly while the majority of the spelling mistakes are present in the middle of the word. To render translation robust to such spell mistakes, we randomly drop a character from 30-50% of the words in the query and train the model with the corresponding translation as the target.

3.4.4 **WordOrderPermute**. As shown in [8], permuting the order of words in the sentence has proved to be an effective noise technique. In our case, we randomly permute the order of the words in the target and train the model to de-noise it.

For all the data augmentations, for each batch of supervised labeled data, we randomly sample a batch of queries from the dataset and apply a data augmentation to it. The model is then updated with a linear combination of supervised and data augmentation loss. For all the experiments, we set λ to 0.5 (Eq. 1).

3.5 Results

To judge the effectiveness of different data augmentation techniques, we performed an experiment where we first train a Baseline model without data augmentation. Then, we add each data augmentation, one at a time, and observe its effect. This experiment is carried out for all the models. Table 2 shows the BLEU score result of the experiment. All the BLEU scores are computed using SacreBLEU [9].

From the results, we draw a few observations. AutoEncoder, CharDrop, and Masking provide an improvement over the baseline for all the models. Surprisingly, a simple data augmentation such as Auto-encoder provides improvement to the accuracy. We further analyze the effect of AE augmentation with additional experiments and details are provided in Appendix section 8.3. WordOrderPermute has a negative effect over the baseline in the case of BART and T5. The reason could be the lack of grammar in the search queries and hence, the model is not able to identify the relative positions of the words. BART and T5 provide comparable accuracies while Bert2Bert gave inferior results. This indicates that the selection of the pre-trained model has an effect on the accuracy and some of the pre-trained embeddings are more favorable than others.

4 HINGLISH QUERY DETECTION

To detect Hinglish code-mix queries, we developed a ML based approach. We chose 80k search queries randomly and got them manually tagged for each word in the sequence. Each word is tagged as English, Hinglish or Other languages. From this, we set aside 20k search queries as a test set.

For query language detection, we explored text classification and sequence labeling approaches. For text classification, the entire query is tagged as English if all the words are English, it is tagged as Hinglish if any of the words is Hinglish and it is tagged as Other if all the words are neither English nor Hinglish. We tried following classification models. 1. fasttext classifier, where the word embeddings are averaged and passed through a softmax classifier. 2. 1-D CNN model with softmax classifier where filter size is set to three. 3. Single layer LSTM model with softmax classifier.

Apart from this, we also tried the sequence labeling approach - where we utilize word-level labels in the query. We tried Conditional Random Field (CRF) with a context window of 3 - current, previous, and next word. For CRF, the features used were character n-grams (uni, bi, tri, quad), length of the word, whether the word contains digits, and whether the word contains special characters. While

Model	Precision	Recall	F1 score
Fasttext	0.90	0.61	0.73
CNN	0.80	0.65	0.72
LSTM	0.82	0.63	0.71
CRF	0.84	0.72	0.76

Table 3: Comparison of different approaches for Hinglish query detection

inference, a query is termed as Hinglish if any of the words is classified as Hinglish.

Table 3 shows the comparison results for the different approaches. CRF model provides the best results compared to other approaches. On the test set, a simple keyword-based approach provides precision and recall of 77% and 28% respectively while the CRF-based approach provides recall of 72%, thus improving the detection recall by more than 2x.

5 TRAINING CORPUS PREPARATION

For capturing more word variations, we prepared a training corpus derived from the incoming traffic of unlabeled Hindi and Hinglish queries. We used generic domain third-party APIs for translation and transliteration and also experimented with training in-house generic translation and transliteration models for data tagging, and the results are given in section 8.2. The parallel query training dataset is created as follows:

- Unlabeled Hindi queries: Translating and transliterating them to English
- Unlabeled Hinglish queries: First transliterating to Hindi followed by Translation to English

Hindi (Devanagari) queries are filtered based on the unicode ranges for the script. We also observe a very small percentage of Marathi queries since it shares a script with Hindi. We did not specifically filter out Marathi queries. For Hinglish language detection, we use the approach mentioned in section 4.

We created a parallel corpus of 5M queries for training the model. We observe that translations with are noisy possibly due to domain mismatch. Therefore, for testing and fine-tuning with cleaner labels, we also created a dataset of 70k Hinglish queries tagged in-house. Note that the in-house tagged data is less than 1.5% of the total labeled set.

For model training, we follow a two-stage process. In the first stage, we use 5M noisy query annotations for fine-tuning the model where we warm start the training with the existing pre-trained checkpoints. We use the loss function described in Eq. 1 for training the model. We use DropChar, AutoEncoder, and Word Masking as the data augmentation techniques. For each batch of supervised data, a data augmentation method is chosen randomly and applied to the randomly sampled query batch to compute $Loss_d$. The model is trained for 5 epochs with the learning rate 5e-6 and batch size of 64. In the second stage, we further fine-tune the model on the in-house labeled set of 55k queries where 10% of queries are used for validation and the remaining 15k queries are used as the test set. During the second stage of fine-tuning, we only use DropChar, and AutoEncoder data augmentations and the model is fine-tuned

until the validation loss seizes to improve. We use label smoothing for both stages where the label smoothing parameter is set to 0.1. During inference, we use beam search decoding with a beam size of 3.

Model	BLEU
Azure Translate	25.5
Google Translate	35.8
RandomInit BART	49.9
BART w/o DataAug	50.3
BART with DataAug	**51.3**

Table 4: Results on Hinglish test set

In an earlier experiment, since BART and T5 gave comparable results, we experimented with both the models and computed the BLEU scores on the test set. BART-base and T5-base gave BLEU scores of 51.3 and 51.33 respectively. Interestingly, though the BART-base model is less deep, it provides comparable results T5-base which has twice the number of layers. This indicates that for the query translation, deeper models may not necessarily provide an advantage possibly due to the lack of grammar in the queries. We choose BART for further experiments because it gave comparable results to T5.

Further, to validate the effectiveness of warm starting the training with pre-trained weights, we did an additional experiment. We randomly initialize the BART model with weights as $\mathcal{N}(0, 0.02)$ and follow the two-stage training procedure as explained earlier. Table 4 shows the comparison result. The result indicates that starting the training with pre-trained weights provides better BLEU scores. Hence, pre-trained weights as well data augmentations are essential for good performance.

We also compared our result with general domain translations from Azure and Google translate. Hinglish queries are first transliterated to Hindi, and then a Hindi-to-English translation is used. The first and second row in Table 4 indicates the BLEU score result. The generic domain translators have significantly lower accuracies than the proposed model, possibly due to domain mismatch.

6 KNOWLEDGE DISTILLATION (KD)

For a good user experience, a quick turnaround with search queries is essential. Hence, it is important to have a low latency translation model. Due to a large number of parameters, self-attention computation across all layers, and auto-regressive generation, translation models typically have a higher latency during inference. To reduce the latency of the model, we use Knowledge Distillation (KD) where we train a smaller student model with the larger BART model as the teacher. In particular, we follow an approach proposed in [7] for training the student model. We use a single-layer BART model as the student. We use the following loss function to train the student model.

$$L_{student} = (1 - \lambda)\ (Loss_s + Loss_d) + \lambda\ Loss_{KD} \qquad (2)$$

The training loss consists of three loss components: supervised cross-entropy loss, data augmentation cross-entropy loss, and Knowledge Distillation (KD) loss. λ is set to 0.5 for all experiments. For

Model	# layers	KD loss	BLEU	p95
BART teacher	6	-	51.3	~ 200 ms
Student (Q.)	1	CE	49.7	~ 47 ms
Student (Q.)	1	JS	50.8	~ 47 ms

Table 5: Results for Knowledge Distillation. (Q.) indicates the Weight Quantization in the student.

Hinglish Query	English Translation	Issue
semsung phone 8 hajar vala	samsung phone for 8 thousand rupees	Brand mispelling
redami 5a ki baitiry	redmi 5a battery	Brand mispelling
goldan kalar ka aie linar	golden colour eye liner	Spell errors
lenovo k6 ka kabar	lenovo k6 cover	Spell errors
makhamal wala seutar	velvet sweater	Spell errors
resmi 7a kwr	redmi 7a cover	Spell errors
250 tak ki biluthuth	bluetooth upto 250	Spell errors
baal hawa machine	hair dryer machine	Articulation Gap
juta bina dori wala	shoe without lace	Articulation Gap
teen char sau wali saree sadi	saree for 300-400 rupees	Free flowing text
kam keemat mein four g mobile	4g mobile in low price	Free flowing text

Table 6: Example results for Hinglish to English query translations.

the KD loss component, we experimented with Cross-Entropy (CE) and Jensen-Shannon (JS) loss. In contrast to Kullback–Leibler (KL) divergence or Cross-Entropy (CE), JS loss is symmetric, bounded, does not require absolute continuity [4]. The JS loss is defined as follows.

$$JS = D_{KL}(T_p||m) + D_{KL}(S_p||m) \qquad (3)$$

Where T_p and S_p indicate output probability distributions of the teacher and the student respectively. The distribution m is obtained as follows.

$$m = 0.5 * T_p + 0.5 * S_p \qquad (4)$$

While training the student model, $Loss_s$ and $Loss_d$ are computed on the batch of manually labeled data. For each batch of manually tagged data, we randomly sample a batch of Hinglish queries from the 5M corpus. We obtain the pseudo translation labels from the teacher on this Hinglish query batch. Using input Hinglish queries and corresponding pseudo labels and with teacher forcing strategy, we obtain probability distributions for the teacher (T_p) and the student (S_p). Note that the teacher forcing term here is generic to transformers and not related to the teacher model in KD. $Loss_{KD}$ is then computed with either CE or JS. We use Weight Quantization to further speed up the inference on the CPU devices.

Table 5 shows the accuracy result for different KD loss types. JS loss gives a BLEU score improvement of 1.1 over CE and facilitates much better knowledge transfer from teacher to student. Our findings are in line with [4]. We calculate latency for all the models on the test set. The best student model provides more than **4x** speed up for the inference with a 0.5 drop in BLEU score. We also further optimised the latencies via C libraries for production deployment.

7 PRODUCTION SEARCH RESULTS

The code-mix translation model is currently live on the Flipkart app and website. Fig. 1 shows the search results for different Hinglish queries. Note that we get good search results even for the queries with spell errors (queries: mbile phone ki battry, peele klr ka chasma), for the queries with articulation gap (query: baal hawa machine), and for the queries where attributes such as color and price are specified (query: kale jute 300 tak).

8 ADDITIONAL EXPERIMENTS

8.1 Comparison with multilingual pre-trained models

We compared the results of BART model with multilingual pre-trained models. Specifically, we consider 2 pre-trained models, mT5-small and mBART-large-50. We follow a similar training procedure for multilingual models where we first fine-tune the model with the 5M corpus for 3 epochs using a learning rate of 5e-6 and then further fine-tune with a manually labeled corpus using a learning rate of 1e-5. Table 7 shows the comparison accuracy. The result indicates that models pre-trained with English text provide better initialization than multilingual models for query translation.

Model	BLEU
BART	51.3
mT5	48.3
mBART	46.6

Table 7: Multilingual models on Hinglish test set

Query: kale jute 300 tak

Query: mbile phone ki battry

Query: baal hawa machine

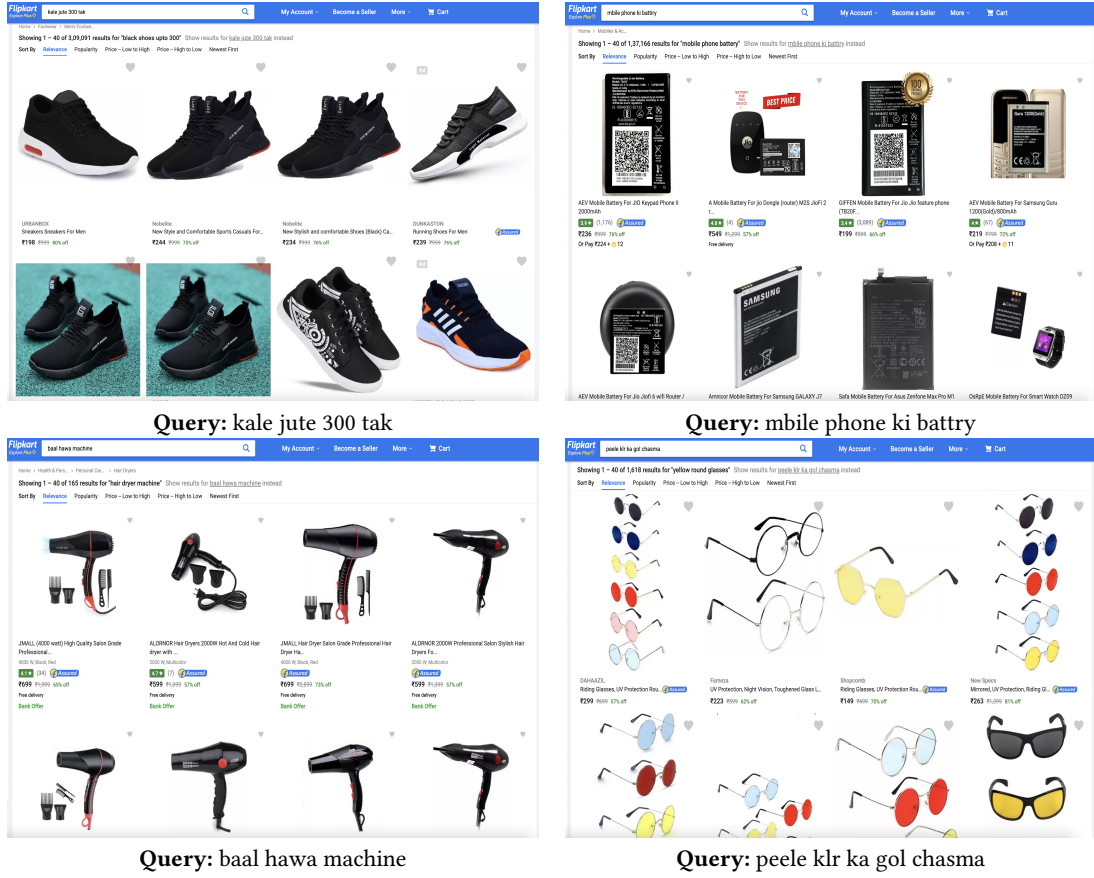**Query:** peele klr ka gol chasma

Figure 1: Hinglish query search results on Flipkart website.

8.2 Comparison with in-house generic models for dataset tagging

We also experimented with building generic translation and transliteration models in-house for dataset tagging and observed comparable results with the generic third-party APIs. Model details below:

8.2.1 Translation model. We train a generic translation model for English to Hindi translation. For training the model, we use the following publicly available translation datasets: Opensubtitiles, Pib, PMI, WikiMatrix, ALT, FLORES-101, indian-parallel-corpora, Man ki bat,WikiMedia,Tatoeba, Global voices, GNOME, KDE, Ubuntu. The total size of the training corpus is ~1.4M. We train a transformer model with 6 layers in encoder and 6 layers in decoder layers. The model has 12 attention heads, 512 hidden dimension, and ~96M trainable parameters. Using this model, we get translations on a large corpus (~157M) of Hindi monolingual text where text is translated from Hindi to English. The monolingual Hindi text is obtained from datasets such as CC-100, OSCAR, news-crawl, Wikipedia. Using this Hindi-English parallel corpus, we then train an English to Hindi translation model. The model architecture is the same as mentioned earlier in this section.

8.2.2 Transliteration model. For building a transliteration model, we use a dataset of size 96.4k of a word-level manual taggings of

Setting	BLEU
Dataset created with generic domain third-party APIs	51.3
Dataset created with in-house generic models	51.1

Table 8: Third-party APIs vs in-house generic models

Hindi-English transliterations. We train a 2 layer transformer model that uses character level tokenization on this dataset. The model has 4 attention heads and a hidden dimension size of 128. For transliterating of an input text, a hybrid approach is followed. If a word from input text is present in the manual mappings, the corresponding transliteration is used. If the word is not present, then a transformer model is used with greedy decoding to get the word transliteration.

To create a large parallel Hinglish-English query corpus, we first translate ~18M unlabeled English search queries to Hindi using the translation model. We then use the transliteration model to transliterate Hindi text to English.

The BART model is then pre-trained on this dataset and fine-tuned on the manually tagged Hinglish queries as before. Table 8 shows the BLEU score comparison result on the test set. With the dataset created with generic in-house models, we get comparable results as with third-party APIs.

8.3 Analysis of Auto-Encoder (AE) objective

Language Model (LM) regularization is known to help in the translation tasks [5]. We hypothesize that Auto-Encoder (AE) augmentation may have a similar effect as adding a LM regularization. To validate this, we design the following experiment. During the training, we analyze the cross attention matrices in different layers in the decoder. This is because the decoder in the conditional generative seq2seq models differs from the decoder in language models (such as GPT) in terms of additional cross attention layers. While generating the output tokens, a conditional decoder also pays attention to encoder states through cross attention layers. Intuitively, the conditional decoder with identity cross attention matrices would be equivalent to a language model. To verify if 'any' of the cross attention matrices in the decoder are becoming similar to the identity matrix with AE data augmentation loss, we calculate error using the following equation.

$$e = \min_{h} ||C - I||_F \qquad (5)$$

where C indicates the cross attention matrix and I indicates the identity matrix. We calculate the Frobenius norm of the difference between the cross attention matrix and the identity matrix and the error (e) is calculated by taking the minimum across the attention heads. Note that, since the same tokenizer is used for the encoder and the decoder, cross attention matrices are square matrices. The error (e) is calculated across all layers in the decoder. We use the same 200k query dataset as mentioned in section 3.1 for the experiment. Out of 200k samples, 190k are used for training and 10k are used for validation. The model is trained with the loss as specified in Eq. 1 where $loss_d$ is only calculated with AE augmentation. The model is trained for 5 epochs and at the end of each epoch, we calculate the average error e on the validation set where target-side data is used as the input and the output as mentioned in section 3.4.2.

Figure 2 shows the error values for different layers across 5 epochs as the training progresses. Interestingly, for the initial layers (layer: 2,3 4), the error goes on decreasing indicating that at least one of the cross attention matrices is gradually becoming similar to an identity matrix. For the last (6th) layer though, we do not observe a similar trend indicating that the last layer features are more task-specific.
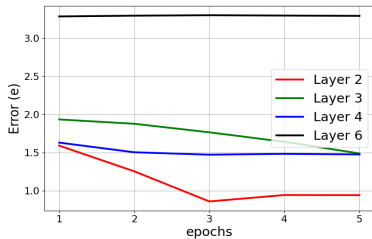

Figure 2: Analysis of cross attention matrices for Auto-Encoder data augmentation.

9 CONCLUSION

In the paper, we describe a transfer learning-based approach for code-mix query translation. We found the pre-trained encoder-decoder model and data augmentation technique best suited for the task through experimentation. A larger training corpus is created with translation of the unlabeled query data using generic domain models. To reduce the inference time latency of the model, we used Knowledge Distillation and Weight Quantization. The effectiveness of the proposed method has been validated through experimental evaluations and AB testing. The model is currently live in production, enabling search with Hinglish language for millions of queries.

10 ACKNOWLEDGEMENTS

REFERENCES

[1] Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-Supervised Learning for Neural Machine Translation. arXiv:1606.04596 [cs.CL]

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling Code-Mixed Translation: Parallel Corpus Creation and MT Augmentation Approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 131–140. https://aclanthology.org/W18-3817

[4] Erik Englesson and Hossein Azizpour. 2021. Generalized Jensen-Shannon Divergence Loss for Learning with Noisy Labels. arXiv:2105.04522 [cs.LG]

[5] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. arXiv:1503.03535 [cs.CL]

[6] Ganesh Jawahar, El Moatez Billah Nagoudi, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. 2021. Exploring Text-to-Text Transformers for English to Hinglish Machine Translation with Synthetic Code-Mixing. arXiv:2105.08807 [cs.CL]

[7] Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. arXiv:1606.07947 [cs.CL]

[8] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461 [cs.CL]

[9] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. arXiv:1804.08771 [cs.CL]

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG]

[11] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics* 8 (2020), 264–280.

[12] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. arXiv:1511.06709 [cs.CL]

[13] Vivek Srivastava and Mayank Singh. 2020. PHINC: A Parallel Hinglish Social Media Code-Mixed Corpus for Machine Translation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*. Association for Computational Linguistics, Online, 41–49. https://doi.org/10.18653/v1/2020.wnut-1.7

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).