Enhancing E-commerce Spelling Correction with Fine-Tuned Transformer Models

Arnab Dutta
ardutta@ebay.com
eBay GmbH
Dreilinden, Germany

Gleb Polushin
gpolushin@ebay.com
eBay GmbH
Dreilinden, Germany

Xiaoshuang Zhang
xiaoszhang@ebay.com
eBay Inc.
Shanghai, China

Daniel Stein
danstein@ebay.com
eBay GmbH
Aachen, Germany

ABSTRACT

In the realm of e-commerce, the process of search stands as the primary point of interaction for users, wielding a profound influence on the platform's revenue generation. Notably, spelling correction assumes a pivotal role in shaping the user's search experience by rectifying erroneous query inputs, thus facilitating more accurate retrieval outcomes. Within the scope of this research paper, our aim is to enhance the existing state-of-the-art discriminative model performance with generative modelling strategies while concurrently addressing the engineering concerns associated with real-time online latency, inherent to models of this category. We endeavor to refine LSTM-based classification models for spelling correction through a generative fine-tuning approach hinged upon pre-trained language models. Our comprehensive offline assessments have yielded compelling results, showcasing that transformer-based architectures, such as BART (developed by Facebook) and T5 (a product of Google), have achieved a 4% enhancement in F1 score compared to baseline models for the English language sites. Furthermore, to mitigate the challenges posed by latency, we have incorporated model pruning techniques like no-teacher distillation. We have undertaken the deployment of our model (English only) as an A/B test candidate for real-time e-commerce traffic, encompassing customers from the US and the UK. The model attest to a 100% successful request service rate within real-time scenarios, with median, 90th percentile, and 99th percentile (p90/p99) latencies comfortably falling below production service level agreements. Notably, these achievements are further reinforced by positive customer engagement, transactional and search page metrics, including a significant reduction in instances of search results page with low or almost zero recall. Moreover, we have also extended our efforts into fine-tuning a multilingual model, which, notably, exhibits substantial accuracy enhancements, amounting to a minimum of 16%, across four distinct European languages and English.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671625

CCS CONCEPTS

• **Information systems** → **Query reformulation**; **Query suggestion**; *Language models*; Novelty in information retrieval; Search interfaces;

KEYWORDS

Transformers; Encoder-Decoder Architecture; Fine-Tuning; Spelling Correction; Knowledge Distillation; BART; T5; Multilingual; Online Inference

ACM Reference Format:
Arnab Dutta, Gleb Polushin, Xiaoshuang Zhang, and Daniel Stein. 2024. Enhancing E-commerce Spelling Correction with Fine-Tuned Transformer Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3637528.3671625

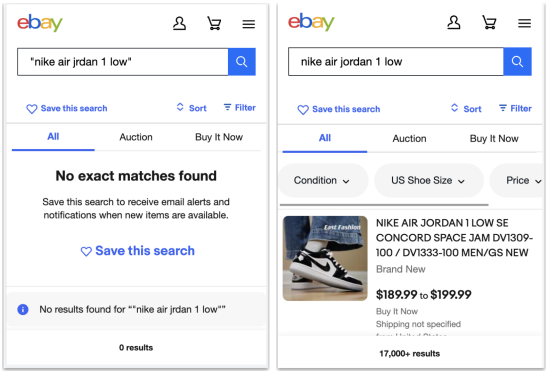
1 INTRODUCTION

The primary goal of an e-commerce platform is to ensure a seamless marketplace experience for buyers and sellers. A key element in achieving this is a robust search functionality within the platform's item inventory, enabling efficient retrieval of relevant items. This task is notably sensitive to the accuracy of user input, which, unfortunately, frequently contains spelling errors. Statistically, spelling errors by users are very common (estimates of 10 % of all queries, as reported by Google,[1] are also in line with our own observations). These errors significantly impact retrieval, leading to zero-result pages, as illustrated in Figure 1. Therefore, a resilient spell correction mechanism is crucial to decipher user intent, correct errors, and deliver desired search results. Implementing such a mechanism is essential for optimizing the overall marketplace experience.

In the development of an integrated, production-ready spell correction module, several key considerations come into play. These considerations encompass: (1) *placement:* its position in the overall life-time of the query, including downstream applications (2) *feedback mechanism*: different forms of human feedback loops, e.g., as (revertible) suggestion or overwrite (3) *quality*: the actual quality of the corrections in terms of accuracy and robustness, taking into account the special usage requirements of the e-commerce, where brands, persons or entities in general are often intentionally "misspelled" to gain attraction, (4) *site speeds*: Evaluating potential

[1] https://blog.google/products/search/abcs-spelling-google-search/, retrieved July 2023

Figure 1: (a) Search results with a misspelling results in a search page with 0 results (b) Desired result with the correction produces over 17K results



trade-offs in terms of quality in adherence to service level agreements (SLAs), encompassing factors like response time, memory usage, hardware resources, and system reliability.

While the concept of spelling errors is as old as the human ability to write, recent advancements in computational power, algorithmic paradigms, and available baseline models create more opportunities to understand the user's intent regardless of any input ambiguity or actual mistakes. However, the key problem from an industry perspective remains to find a careful balance of the accuracy and generative power of any spelling correction model versus its practicability in (online) production systems. In this paper, we offer insight into potential gains and pains of generative modelling versus state-of-the-art discriminative models.

In essence, this research paper contributes to the field by offering insights and solutions related to quality improvement and SLAs in the context of fine-tuned models for e-commerce applications. The key contributions of this paper can be broadly summarized as follows:

- Empirical Validation of Pre-trained Models; we have conducted investigations that demonstrate the superior performance of fine-tuned pre-trained models, specifically tailored for spell correction tasks on e-commerce data sets. Our findings reveal their outperformance compared to discriminative models employing the Long Short-Term Memory (LSTM) architecture.
- Addressing Real-time Latency; to tackle real-time latency concerns, we have incorporated distillation techniques into our methodology. This approach ensures that the system remains responsive and efficient.
- Multilingual Model Development for Enhanced Cross-Site Consistency; we have developed a multilingual fine-tuned model designed to provide a single, unified solution across various sites. This model exhibits notably higher accuracy when compared to its monolingual counterparts.

Thus our work aims to solve a fundamental concern in applied machine learning which is a trade-off between model performance and model latency. We empirically show how to to fine tune a encoder-decoder Transformer model, apply distillation techniques and deploy it for A/B testing for English sites. Furthermore, we extend the same principles to fine tune multilingual models and report our offline performance gains for non-English sites. These models are in preparation for A/B testing phase.

2 RELATED WORK

The task of spelling correction in e-commerce search system is not a recent one, and various approaches have been proposed to address the problem for multilingual markets. One typical class of model is called the noisy channel, which decouples the system into 2 phases: candidate generation and ranking. Edit or phonetic noises are introduced to generate potential correction, then an error model is applied to select the best correction path. [9] designed Tree and Graph-based approaches to create typographic errors in candidate generation, and introduced synthetic spelling errors to represent noisy channels, which can be easily generalized to languages with small effort. [28] proposed a hybrid candidate generation method to integrate phonetics mappings into the system and leveraged the phonetic signals to improve ranking performance. In this regard, Neuspell [11] should also be mentioned, since the authors tried to correct in-context errors by designing the problem as a Sequence labelling task. This toolkit also allows us to perform comparative spelling correction models analysis on. However, this tool was developed for English only and lacks any multilingual aspects. In contrast, our proposed methodology is a sequence learner and has multilingual capabilities. However, we do compare our methodology against the evaluation numbers reported by the authors. The results are reported in evaluation section 6.4.

Another class of model learns to correct spelling errors on top of language models. The state-of-the-art spell correction systems often benefit from language models for performance boosting. [24] formulates the problem as a sequence-to-sequence task. The model is trained with language model corpus with deliberately introduced noises, including changing, removing or adding characters. From multilingual perspective, it still remains to be a challenging task to expand a well-trained speller model to multiple languages. Speller100 [12] proposed a zero-shot learning stack for over 100 languages with a training corpus from a few cultivated markets. The model is pre-trained as a character-level seq2seq denoising autoencoder, so that the model can benefit from the general language comprehension capabilities.

In this context, the work of Rothe et al. [23] requires a special mention due to its strong overlap with our efforts. The authors solve grammatical errors utilising mT5 model. Their work involves fine tuning a large (11B parameters) multilingual pretrained model on a synthetic dataset. This work is essentially not solving the exact same problem as ours but we can adapt some fundamental approaches to further improve our methodology.

In contrast, our work introduces a modelling paradigm shift by leveraging the power of pre-trained language models in the context of e-commerce spell correction. Our fine-tuned sequence-to-sequence architecture can unblock the label space limitations in classification task. Our work extends well to other sites (languages) with limited amount of training samples, thereby enhancing the multilingual prediction capabilities with this approach.

3 BASELINE MODELS

Noisy Channel Speller The statistical spell correction model of Noisy Channel was initially implemented by [13]. It generates 12 most frequent correct tokens for each query token and forms up a transition graph from the token sequence. For each possible path in the graph, correction score is calculated as the sum of a statistical language model score and an error score, where the language model provides the transition probability and the error model considers the phonetic, edit distance, cosine similarity and entropy between the misspelled and corrected tokens.

Char-Tagger Speller is a bidirectional long short-term memory (BiLSTM)-based spell correction model. It is comparable to the architecture as described in [8]. This approach reformulates the task of spell correction as a character level classification problem and for each input character, predictions are defined as an edit operation: *keep*, *delete*, *append* and *replace*. To enhance the model with domain knowledge, Char-Tagger first encodes the query text with a pre-trained character-level Flair[1] language model, which is then fed into a classifier for edit operation predictions.

We would like to highlight, that the set of baselines are not just limited to the ones listed above but can be more. For instance, Transformer-based encoder-only model and decoder-only models fine-tuned on downstream tasks can also serve as a possible baselines, and aim to explore these paradigms in our future work.

4 APPROACH

Our aim is to enhance and operationalize the current spelling correction model while creating a language-agnostic model for deployment across major sites. This involves exploring alternative modeling approaches and the efficiency of pre-trained models for e-commerce tasks. Notably, existing noisy channel models lack generative capabilities, while BiLSTM-based character models are inherently discriminative and constrained by edit operations. We sought a generative approach capable of generalizing across languages and overcoming these limitations. Our strategy addresses the identified model limitations and focuses on optimizing for latency, supported by controlled experiments to deploy these models effectively in real traffic scenarios.

To achieve this objective, we leveraged Transformer-based architectures and fine-tuned our spelling correction model using pre-trained language models ([3] [4]), including, but not limited to, T5 and BART. In the context of multilingual support, we adopted mT5 and mBART models. We employed no-teacher distillation approach to fine-tune smaller models to optimise them for both inference speed and performance. Our approach is structured into several sequential components, each of which will be comprehensively discussed in subsequent subsections:

- Training Paradigms:
  - Fine Tuning (Section 4.1): We assess different classes of fine-tuned transformer models to identify the most suitable for our specific use case, with a focus on improving offline performance metrics during training.
  - Multilingual Tuning (Section 4.2): We investigate the performance of pre-trained multilingual models in the context of multilingual queries across non-English websites.
- Model Inference Paradigms:

Table 1: Snippet of training samples consisting of natural, synthetic and correct pairs.

incorrect query	correct query	source
rnyodome nike	*rhyodomo nike*	natural
audity 2000	*audi tt 2000*	natural
checks tie	*checks tie*	correct
1/4-28 rod end	*1/4-28 rod end*	correct
mabetix toy	*magnetix toy*	synthetic
nike track clluv	*nike track club*	synthetic

  - Model Distillation (Section 4.3): We employ knowledge distillation techniques to optimize smaller layered networks, thereby enhancing latency during model inference

4.1 Fine-tuning

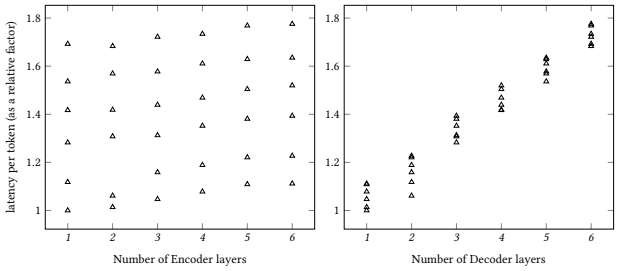
The purpose to design a fine-tuning based approach was two-fold: (1) reduce computational overhead of building a language model from scratch, and (2) add domain specific knowledge to the model. In general, the above class of models have been trained with open domain corpora which are precious for accurate language understanding. Generally, an e-commerce site deals with a lot of domain-specific input data and vocabulary which are hard to interpret without proper domain knowledge. We mined our search logs to fine-tune these models, thereby adjusting the network weights of the base model to better align on the task. We emphasize that, when fine-tuning models like T5, for a good model performance it is crucial to set the task prefix accordingly. This is an artifact of how T5-like models were originally trained.

4.2 Multilingual Training

The competing modelling paradigm lacks a multi-language support. There are several limitations in doing that; in principle, one can train multiple site specific models but that accounts for a higher maintenance overheads and results in model refresh inconsistencies. Further, as elaborated on in the Section 6.2, local site queries are hardly limited to words from the specific language. Especially when it comes to entities and brand names, a multilingual model should benefit from comparable mixed vocabulary usage in other sites.

In order to tackle this, we trained a generic multilingual model supporting English (EN), German (DE), French (FR), Italian (IT) and Spanish (ES). The idea is to fine-tune the model with *enough* samples from each of the aforementioned sites. The only caveat is that this approach is not a zero shot learning methodology and hence is not expected by design to perform well for unseen languages in the future, but it already covers the most active markets. We selected mT5-small, mT5-base and mBART-large as the available pre-trained models with inherent multilingual generative capability. Since mT5-small had even lesser number of model parameters (300 million) compared to both mBART-large (610 million) and mT5-base (580 million), we evaluated with mT5-small.

Figure 2: Effect of Average latency per token with varying number of encoder and decoder layers. The y-axis denotes the factor at which the time taken to infer per token changes. For instance, with varying decoder layers, a model with 6 layers decoder is approximately 1.8 times slower than the one with 1 layer decoder.



4.3 Model Inference Optimising

It was empirically observed that BART model suffered from high online latency. Since the inference generation is performed in an autoregressive fashion, it was accounting for a sequential decoding process which significantly impacted our inference speeds. In order to address this concern we focused on inference optimisation strategies involving knowledge distillations and quantization. These approaches are targeted towards squeezing model sizes without significant drop in model accuracies. However, in our case we observed significant model performance degradation using quantisation techniques. This gap was hard to mitigate even with continued fine-tuning. Thus our approach primarily focused on reducing the model network layers specially the decoder layers in order to come up with a smaller model with lesser number of parameters. We support our hypothesis with the empirical evaluation presented in Figure 2. We varied the number of encoder layers and decoder layers for BART-base model and observed the average latency for each token (in milliseconds). As seen, encoder layer size change has hardly any directional pattern but with decoder layer change we do observe one. To this end, we adopted no-teacher distillation techniques as proposed for models like DistillBART [2]. The key idea behind this approach was to create a cloned student model with reduced layers without any active teacher model *guiding* the student model to mimic the teacher's predictions. Our hypothesis being, a smaller network with reduced decoder layers and fine tuned further can achieve at par results. We also have empirical results that shows that reducing encoder layers had minimal latency impact, only decoder layer reduction had proportional latency improvements.

5 SETUP

Training Data: For the spell correction task, we need to sample high-quality misspelling/correction pairs as training data. [8] proposed a fused training set generated from both natural and synthetic sources. This work adopted a very similar data collection techniques for training and evaluation. Specifically, in natural data collection, we picked up in-session user refined query pairs from our user search logs, applied numerous post-filters to select character-level edits and removed those with user intent drift. To enhance the performance of the popular queries, we also generated synthetic misspellings from frequent query seeds [25]. The overall training data is a combination of the two aforementioned sources, which captures the most typical misspelling patterns. We estimate that at least 10 % of all queries contain spelling errors, thereby leaving the majority of requests as correct. Hence, we also enhance the training data with no-op pairs. Table 1 presents some sample data points from each of the three data sources we had collected. Approximately, our training data set contains 45 % natural data pairs, 16 % synthetic pairs and the rest 39 % correct pairs. We prepared a training data sets consisting of roughly 91 million query pairs distributed over natural, synthetic and correct for site US alone. Likewise, for other sites we have the following data sizes: DE (24 million), IT (11 million), ES (4.6 million), FR (12 million).

Table 2: Training and Inference Details of the final fine tuned EN BART-base model, which was A/B tested on US/UK sites.

Training		Inference	
Data	91M	Model	BART-base
learning rate	1e-4	Decoding	Greedy (Beam Size = 1)
epochs	6	#Parameters	94M
optimiser	AdamW	GPU	A100
batch	64	Traffic	10%
# GPU	8 V100	Runtime	ONNX-RT

The training iterations were performed for 6 epochs with a linear learning rate of 1e-4 and batch size of 64. We employed HuggingFace Trainer API [3] to design our fine-tuning pipelines. The training was performed on 8 V100 GPUs on a single node cluster.

Golden Set: We manually created the golden sets in five different languages: English, German, French, Italian and Spanish. We focused on those languages only as they are the primary languages from our top revenue generating sites. In particular, we sampled a query set from users' search history, where the user had refined the query within the same search session. The key consideration here was to restrict the reformulations only to the valid ones having shared tokens and no intent drifts. The queries before refinement are considered as potential misspelled query and sent for human correction. The golden set consists of roughly 10 K hand curated pairs of misspellings and its corrected counterparts per language. It also contain correct pairs where the model should not perform a correction. This makes the data set more realistic and practical since every input does not require correction. The numbers we report are against this golden set across all our experimental runs. Furthermore, we also maintained different golden sets across all the sites (English and non-English) we had experimented with (Section 6.2).

Inference Engine: In order to have a systematic latency evaluation, we converted each of the produced PyTorch models and converted them into Open Neural Network Exchange (a.k.a. ONNX) models. This ONNX format solves a very fundamental and a critical issue with model, which is hardware and platform compatibility. ONNX models are more portable than the language specific model

[2]https://huggingface.co/valhalla/distilbart-mnli-12-9

[3]https://huggingface.co/docs/transformers/main_classes/trainer

Table 3: Performance of speller fine-tuned from different pre-trained models. All models were trained on 1 M sample of dataset and evaluated on US (English only) golden data set. Numbers are relative to baseline (Char-Tagger), they are masked with '–'. ΔP(%) = percentage precision change, ΔR(%) = percentage recall change, ΔF1(%) = percentage F1 change, ΔAccuracy(%) = percentage accuracy change, ΔL = offline inference latency change as a factor.

Model	#Params (mill.)	ΔP(%)	ΔR(%)	ΔF1(%)	ΔAccuracy(%)	ΔL
T5-small	60	0.0	+3.2	+2.1	+1.6	9.1x
T5-large	738	+10.9	+31.2	+21.0	+22.0	30.8x
Flan-T5-base	247	+9.3	+26.9	+19.1	+17.9	14.7x
Flan-T5-large	783	+12.6	+46.2	+30.4	+32.1	28.9x
BART-base	139	+8.3	+37.6	+23.3	+24.2	5.0x
BART-large	406	+8.0	+37.4	+22.9	+24.4	10.7x
(Baseline) Char-Tagger	28	-	-	-	-	1x

files across all platforms supporting ONNX Runtime [6], which is a high-performance inference engine for deploying ONNX models to production. The ONNX models were deployed on our internal staging service platforms on A100 GPU card running ONNXRT. On the client side, we designed a load testing environment using K6[4]. This allowed us to easily script and design our request pattern in terms of number of concurrent users, number of requests and time duration. We performed a constant rate concurrent requests (100 qps), i.e., queries with misspellings at our deployed BART model and measure the percentile latency from the model responses. All of the developed models are implemented using the PyTorch library [20] and have both CPU and GPU environment compatibility. Table 2 presents the details in a nutshell for the BART-base model which was eventually A/B tested.

6 EVALUATION

Overall our approach has the following evaluation progression: (1) *Model Architecture Evaluation*: Validate if the transformer based models using the T5- or BART-based solution meet our performance expectations on US only dataset. We compare with baseline models to decide this hypothesis. In this evaluation, we do not consider multilingual modelling approach but focus only on EN. (2) *Multilingual validation*: Extend and evaluate the multilingual performance of fine-tuned models. (3) *Model Latency Evaluation*: Deploy a distilled model to serve real traffic on web, desktop and mobile devices for US and UK sites. This ensures the latency of for our model and also measures the business KPIs impacted by our model. We have deployed only the EN only model BART model for the online serving and the multilingual model serving is still work in progress.

Please note due to our internal publication guidelines, we have consistently reported the evaluation numbers as relative numbers in contrast to reporting absolute numbers. Hence, all the numbers presented as part of our evaluation steps in Table 3, Table 4, Table 5, are represented as percentage improvements over the Char-Tagger baseline method. All the training experiments were performed on a multi-GPU V100 cluster, and the inference of the deployed model was performed using a A100 cluster.

[4]https://k6.io/docs/

Table 4: Relative (Char-Tagger as baseline) performance on US human judged data. The baseline values are masked on purpose, we are reporting only the relative improvements here. Models were trained on full dataset. ΔP(%) = percentage precision change, ΔR(%) = percentage recall change.

Model	ΔP(%)	ΔR(%)	ΔF1(%)
Noisy Channel	-15.4	-27.5	-21.7
BART-base	**+1.6**	**+6.1**	**+3.9**
Char-Tagger	-	-	-

6.1 Model Architecture Evaluation

Given the nature of our problem formulation as a sequence-to-sequence learning task on textual inputs, our selection of models focused on the variants proven to be effective on similar tasks. We randomly sampled 1 M of misspelling/correction pairs for training, and evaluated on the US human judged golden set. All models have been trained for 6 epochs. The purpose of this experimental run was to evaluate the comparative performances quickly against baseline model under similar settings. Hence, we decided to run it for a smaller sample size and on smaller number of epochs. In Table 3, we present the performance study on some of the T5 and BART class of models. Results showed that all transformer-based models outperform Char-Tagger on offline metrics. Inference latency was expected to be larger, as an artifact of having a larger number of model parameters. In later section, we dwelve into inference optimisation strategies as well.

Considering this trade-off, while Flan-T5-large seemed to perform well on the offline metrics, we decided to pick BART-base model for further evaluation because of its smaller parameter space than Flan-T5 models, which would pay off during inference and serving. Also, our latency computation showed BART-base as a faster option than Flan-t5 without any significant loss in accuracy numbers. Hence, we picked this BART-base model and trained on the complete data set (roughly 91 million data points) and compared with the baseline models under similar settings. Results of this comparison are shown in Table 4. Since this was a time consuming run,

Table 5: Multilingual Model performance on 5 languages. Char-Tagger was trained on 1 M sample of dataset, our model on 1 M sample of each site's dataset – 5 M in total. Numbers are relative to site-specific monolingual Char-Tagger baseline and are masked with '–'. ΔP(%) = percentage precision change, ΔR(%) = percentage recall change, ΔF1(%) = percentage F1 change, ΔAccuracy(%) = percentage accuracy change.

Model Name	Site (Language)	ΔP(%)	ΔR(%)	ΔF1(%)	ΔAccuracy(%)
mT5-small	US (English)	+7.7	+29.7	+19.1	+20.1
mT5-small + lang token		**+7.8**	**+31.2**	**+19.8**	**+20.9**
monolingual Char-Tagger		-	-	-	-
mT5-small	DE (German)	+3.6	+21.8	+13.3	+15.3
mT5-small + lang token		**+4.1**	**+23.6**	**+14.3**	**+16.3**
monolingual Char-Tagger		-	-	-	-
mT5-small	ES (Spanish)	**+3.4**	+34.4	+19.5	+21.0
mT5-small + lang token		+3.3	**+35.3**	**+19.7**	**+21.8**
monolingual Char-Tagger		-	-	-	-
mT5-small	FR (French)	+4.8	+35.7	+20.8	+19.5
mT5-small + lang token		**+5.5**	**+36.4**	**+21.5**	**+20.6**
monolingual Char-Tagger		-	-	-	-
mT5-small	IT (Italian)	+2.0	+30.5	+17.1	+22.5
mT5-small + lang token		**+3.1**	**+32.0**	**+18.4**	**+24.1**
monolingual Char-Tagger		-	-	-	-

we did not evaluate all the models on full data set, rather just the winning candidate from Table 3. Char-Tagger was trained for 50 epochs, and BART-base for 6 epochs. Compared with the works of Noisy Channel Speller and BiLSTM Char-Tagger Speller, our BART-base model for Speller performs marginally better than Char-Tagger on precision, and significantly better than Noisy Channel and Char-Tagger on recall. We observed 3.9 % gain in F1 score over the BiLSTM Char-Tagger and over 21 % improvement on F1 scores on the Noisy channel model. However, it is to be noted that the reported numbers for BART-base can potentially improve under even larger epochs. This evaluation helped us prove that the transformer based architecture could surpass LSTM model in terms of model performance but is slower in terms of latency.

6.2 Multilingual Evaluation

To motivate a multilingual setup, we started by conducting human analysis on 1,000 entries from German site. We found that 38.5 % of the queries were solely using words from the German language (with minor product specifics such as *A34* or similar), 25.1 % contained brand names or entities, 4.3 % featured both German and English words (e.g., *"jeanshose stretch"*), and 31.4 %, i.e., roughly one third, do not contain any German word at all. This analytical efforts accurately revealed the mixed nature of colloquial user language, especially in an e-commerce domain. This also justifies that building a model which is indifferent to languages and immune to mixed vocabulary will be ineffective in the long run across different sites.

We fine-tuned mT5-small on a mixed dataset, consisting equal parts of site-specific dataset. Since mT5- small had even lesser number of model parameters (300 million) compared to both mBART-large (610 million) and mT5-base (580 million), we evaluated with mT5-small alone for the multilingual part. Table 5 highlights the comparative numbers across the 5 sites and respectively 5 languages. Each row represents the model and site combination and the respective metrics. Since the baseline model has no multilingual capabilities, we evaluated the baseline model under 5 different language inputs. The line *monolingual Char-Tagger* in each row of the table means it was trained and evaluated on that specific language only. This serves as our site specific baselines. For this evaluation we also took 1 M misspelling-correction pairs from each site for training, resulting in mT5 with 5 M pairs in total.

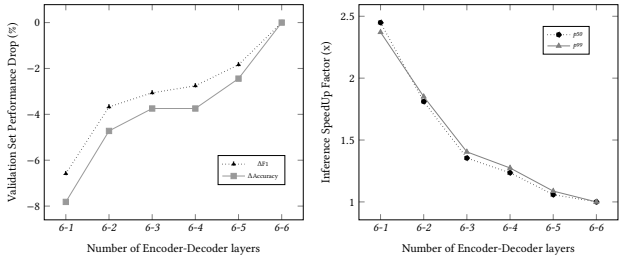
For each of the individual sites, the multilingual model outperforms current site-specific models. The fact that the evaluation sets also contain a mix of languages (cf. Section 5), especially because of brand names and entities, could be the main reason why multilingual model outperforms language-specific models. This table also reports two variants of our multilingual training: one with language tokens and one without. Language token encapsulates the site id and is prepended as an additional token to the input sequence of text. This additional information allows the model to perform marginally better than the scenario when it wasn't added.

6.3 Model Latency Evaluation

Model latency evaluation was conducted in two separate stages, (a) using on premise staging deployment and (b) using live experimentation tests. The first stage allowed us to identify the model-centric optimisations required for model performance enhancements while maintaining site SLAs. The second stage focused primarily on customer engagement and behavioral performance relevant for the overall business.

We first empirically tried to identify the performance-latency trade-offs across different student models. Since reduced decoder layers lead to improvements in inference times but at the cost of

Figure 3: Figure showing the effect of model pruning on latency and model accuracy. All the models have been trained on 1 million EN samples data and ran for 6 epochs with a learning rate of 3e-4. The shorthand model name denotes the student network, for instance 6-1 means a pruned student model with 6 encoder layers and 1 decoder layer. (a) Evaluation data set performance for different student models. The y-axis denotes the drop (denoted with negative sign) in performance as a percentage value considering BART-base as the default model. (b) Model latency under K6 load test. Likewise the y-axis denotes the relative speedup factors in latency.



reduced model predictive performance. To justify that, we fine tuned several student models with varying decoder sizes and reported the evaluation data set metrics for them in Figure 3. This training was performed on 1 million training samples for 6 epochs with a learning rate of 3e-4. Thus we performed 6 model fine-tuning steps including the default BART-base model (shorthanded as 6-6). For the latency test, we deployed these models (PyTorch model files) into our model serving platform as ONNX model formats and utilised K6 to request the model REST endpoints concurrently and at a constant rate of requests per second. The model performance against the validation set is depicted in Figure 3(a) and the relative latency improvements are shown in Figure 3(b).

In our investigation of online inference, we specifically examined model beam size adjustments and greedy decoding (i.e., beam size=1). Our evaluation consistently favored greedy decoding over beam decoding. Despite theoretical differences, where greedy decoding prioritizes computational efficiency and beam decoding focuses on exploring next token possibilities, our practical findings indicated that greedy decoding was superior, particularly under strict latency constraints. Notably, we observed no significant accuracy changes between the two methods.

As expected, models with more decoder layers tend to have better performance against the validation data set in comparison to its sibling models under equivalent number of training steps. The figure depicts the comparative changes in the model performances against the BART-base model which has 6 layers of encoder and decoder respectively. In this context, it is important to also highlight that encoder layers had minimal impact in latency improvements, hence the student models we designed for this empirical evaluation were all having a 6 layer encoder sizes but a reduced decoder layer sizes. We observed that, reducing the network size by 1 decoder layer, i.e., a student model with 6 encoder and 5 decoder layers, leads to a roughly 2.5% drop in accuracy but that gives us a meagre

Table 6: Comparison with NeuSpell on BEA-322 and BEA-4660 reporting the Word Level Accuracy/Correction Rate

Model	BEA-4660	BEA-322
GNU ASpell[2]	68.5 / 10.1	61.1 / 18.9
JamSpell[19]	98.5 / 72.9	96.7 / 52.3
CHAR-CNN-LSTM [14]	97.5 / 82.7	94.5 / 57.3
SC-LSTM [18]	97.3 / 86.6	94.9 / 65.9
CHAR-LSTM-LSTM [16]	97.8 / 84.0	95.4 / 63.2
BERT [7]	98.4 / 92.5	96.0 / 72.1
SC-LSTM		
+ELMO [21] (input)	98.2 / 91.9	96.1 / 69.7
+ELMO (output)	97.9 / 88.1	95.2 / 63.2
+BERT (input)	98.4 / 90.2	96.0 / 67.8
+BERT (output)	97.8 / 88.1	95.1 / 67.2
BART-base (6-1) (ours)	94.6 / 49.3	95.5 / 46.7

1.06x times speedup in p99 latency. This trade-off detection was critical for our final model choice since each additional inference time lag is detrimental to our site speeds. Thus, our empirical results show that, reducing aggressively to a 1 layer decoder student BART-base model leads to 7.8% accuracy drop but a handsome˜2.4x time improvements in latency. In our case latency factor is crucial for online serving. Hence, a student model with encoder size 6 and decoder size 1 seemed optimal as an suitable user testing candidate.

Table 7 presents the evaluation metrics of our final model which was served online over an experimentation framework. The 1 layer decoder model was further fine-tuned on the full data set and trained for 10 epochs with a learning rate of 3e-4. This step was necessary to improve the model quality to be at par or better than the production model, without affecting the latency. We observe a +4% gain in F1 score and +4.3% accuracy improvement compared to the production model. This model was deployed for live experimentation.

The experimentation exposed our BART model to a treatment group of random users and our production speller model to the control set of users. After the simulation period, we observed some of the key metrics significantly improved for the treatment group including the basic search page metrics like a drop in pages with near zero results. We also identified positive trends in the users' buying behaviour for the surfaced items indicating that the BART based corrections were at par to the users' buying intent.

6.4 Public Dataset Evaluation

Our approach in particular is fine tuned to excel on short grammar less inputs typical for eCommerce sites. This by definition *cannot* excel in classical in-context natural language spelling errors. In this regard, we put our model to test using the public tool kit and datasets made available by the authors of Neuspell [11]. We ran our model against the inputs from the two datasets BEA-4660 and BEA-322, which are a challenging set of ambiguous mistakes. Since we process only lowercase queries, our BART model was trained likewise. Hence, in this evaluation step, we compared only against

Table 7: Performance of deployed fine-tuned student model optimised for latency and performance. Numbers are relative to baseline (Char-Tagger), they are masked with '–'. ΔP(%) = percentage precision change, ΔR(%) = percentage recall change, ΔF1(%) = percentage F1 change, ΔAccuracy(%) = percentage accuracy change, ΔL = online inference latency change as a factor.

Model	#Params (mill.)	ΔP(%)	ΔR(%)	ΔF1(%)	ΔAccuracy(%)	ΔL
BART-base student (6-1)	92	+1.2	+7.2	+4.0	+4.3	1.7x
(Baseline) Char-Tagger	28	-	-	-	-	-

Table 8: Example multilingual corrections. (a) "*toll*" is a German word meaning "*great*", but from the context of a swiss knife brand, the English word "*tool*" is correct. (b) "*be quite*" is a brand name of a German company producing cooler devices. (c) "*stabil*" is German for "*stable*", but "*stabilo*" is a brand for board markers, which is not captured by any model yet.

Id	Source	Correct	Char-Tagger	mT5-small
(a)	victorinox toll tasche	victorinox tool tasche	victorinox toll tasche	victorinox **tool** tasche
(b)	be quite lüfter	be quiet lüfter	be quiete lüfter	be **quiet** lüfter
(c)	stabil maker	**stabilo marker**	stabil maker	stabil maker

the lower cased data sets and ignored the exhaustive datasets as was reported in the original paper[5].

As seen in Table 6, our methodology is inferior in terms of the correction rate in both the data sets. But, accuracy on BEA-322 is at par with some of the most accurate systems evaluated by the authors. This is an expected behaviour since, our model is trained on a specific domain of data which are very different from the ones used for evaluation in Neuspell. This comparison places our work in contrast to the other relevant models and also highlights the limitation it possesses. However, having almost at par accuracy on a cross domain dataset does exhibit the overall capabilities of our fine tuning based modeling approach.

7 IMPROVEMENT SCOPES

The BART models both multilingual ones and distilled ones showed overall positive impact. However, this modelling approach has its drawbacks. Table 8 shows some examples of corrections made by monolingual Char-Tagger and mT5 for the German golden set. Our model is seen to be effective against proper in-context corrections and often with wrong entity references. For instance it corrects well the entry *jake wobble* into *jake wobbler*, or queries with entities (baseball player) like *topps 2021 cristian packe* into *topps 2021 christian pack* but it also seems to fail under scenarios where a correction is undesirable. For instance, *the merck index* (the book) to *the merc index* or *easywear o* to *easywear oil*.

The model deployed in our live A/B test framework was initially exposed to a subset of the total traffic, meaning that the model endpoints weren't made available to 100% of users. This decision was made to assess the model's performance under realistic traffic conditions during its first iteration. Although the latency numbers for the online test met our site's service level agreements, the model's capacity to serve at full scale is still constrained. To address this limitation, we're actively working on optimizing the model using quantization techniques. It's worth noting that we've already deployed ONNX models for our online service, but we haven't explored TensorRT paradigms yet, which could offer a promising model format for inference. Currently, we are preparing a multilingual model for our experimentation phase using similar model reduction techniques.

[5]Detailed explanation of each of the presented model is beyond the scope of this paper and it is recommended to refer to the original work.

8 CONCLUSION

The novelty of this work lies in fine-tuning off-the-shelf Transformer models for spelling correction while addressing latency issues. Key results include:

- BART and T5 models generally outperform LSTM models in spelling correction tasks.
- Non-optimized BART shows faster response times than T5, prompting further BART optimizations for latency.
- Reducing decoder layers significantly improves online model latency, enabling real-time traffic serving.
- Our fine-tuned model for US/UK sites enhances user experience, evidenced by reduced session exit rates (>1%) and a drop in zero-result search pages (>8%).
- Extending the modeling approach to non-English sites shows strong performance potential, with A/B testing in progress.

Our work addresses critical aspects of query reformulation in search, leveraging advanced learning paradigms to tackle a traditional problem. We provide a comparative analysis, demonstrating the superiority of our modeling strategy over robust discriminative models. Additionally, we address latency concerns in deploying such advanced models by pruning and distilling them for performance while highlighting necessary trade-offs in system design.

ACKNOWLEDGMENTS

To Tianming Lu, David Chen, Irina Leschuk, Selçuk Kopru and Zhe Wu for the constant support, trust and helpful inputs along the course of the project.

REFERENCES

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

[2] Kevin Atkinson. 2019. *GNU Aspell*.

[3] Osman Büyük. 2020. Context-Dependent Sequence-to-Sequence Turkish Spelling Correction. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 19, 4, Article 56 (apr 2020), 16 pages. https://doi.org/10.1145/3383200

[4] Yu-Chieh Chao and Chia-Hui Chang. 2020. Automatic Spelling Correction for ASR Corpus in Traditional Chinese Language using Seq2Seq Models. In *2020 International Computer Symposium (ICS)*. 553–558. https://doi.org/10.1109/ICS51289.2020.00113

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[6] ONNX Runtime developers. 2021. ONNX Runtime. https://onnxruntime.ai/. Version: x.y.z.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[8] Mengyi Gao, Canran Xu, and Peng Shi. 2021. Hierarchical character tagger for short text spelling error correction. *arXiv preprint arXiv:2109.14259* (2021).

[9] Prabhakar Gupta. 2020. A context-sensitive real-time Spell Checker with language adaptability. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. IEEE, 116–122.

[10] Chengming Hu, Xuan Li, Dan Liu, Xi Chen, Ju Wang, and Xue Liu. 2022. Teacher-Student Architecture for Knowledge Learning: A Survey. arXiv:2210.17332 [cs.LG]

[11] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A Neural Spelling Correction Toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Qun Liu and David Schlangen (Eds.). Association for Computational Linguistics, Online, 158–164. https://doi.org/10.18653/v1/2020.emnlp-demos.21

[12] Lu Jingwen, Long Jidong, and Majumder Rangan. 2021. *Speller100: Zero-shot spelling correction at scale for 100-plus languages*. https://www.microsoft.com/en-us/research/blog/speller100-zero-shot-spelling-correction-at-scale-for-100-plus-languages/

[13] Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A Spelling Correction Program Based on a Noisy Channel Model. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*. https://aclanthology.org/C90-2036

[14] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. arXiv:1508.06615 [cs.CL]

[15] 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 preprint arXiv:1910.13461* (2019).

[16] Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. 2018. Spelling Error Correction Using a Nested RNN Model and Pseudo Training Data. arXiv:1811.00238 [cs.CL]

[17] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual Denoising Pre-training for Neural Machine Translation. *Transactions of the Association for Computational Linguistics* 8 (2020), 726–742. https://doi.org/10.1162/tacl_a_00343

[18] Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Mirella Lapata, Phil Blunsom, and Alexander Koller (Eds.). Association for Computational Linguistics, Valencia, Spain, 229–234. https://aclanthology.org/E17-2037

[19] Filipp Ozinov. 2019. Jamspell.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG]

[21] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[22] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[23] Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2022. A Simple Recipe for Multilingual Grammatical Error Correction. arXiv:2106.03830 [cs.CL]

[24] Shuvendu Roy. 2019. Denoising sequence-to-sequence modeling for removing spelling mistakes. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, 1–5.

[25] Felix Stahlberg and Shankar Kumar. 2021. Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Online, 37–47. https://aclanthology.org/2021.bea-1.4

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903* (2022).

[27] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. arXiv:2010.11934 [cs.CL]

[28] Fan Yang, Ali Bagheri Garakani, Yifei Teng, Yan Gao, Jia Liu, Jingyuan Deng, and Yi Sun. 2022. Spelling correction using phonetics in e-commerce search. (2022).

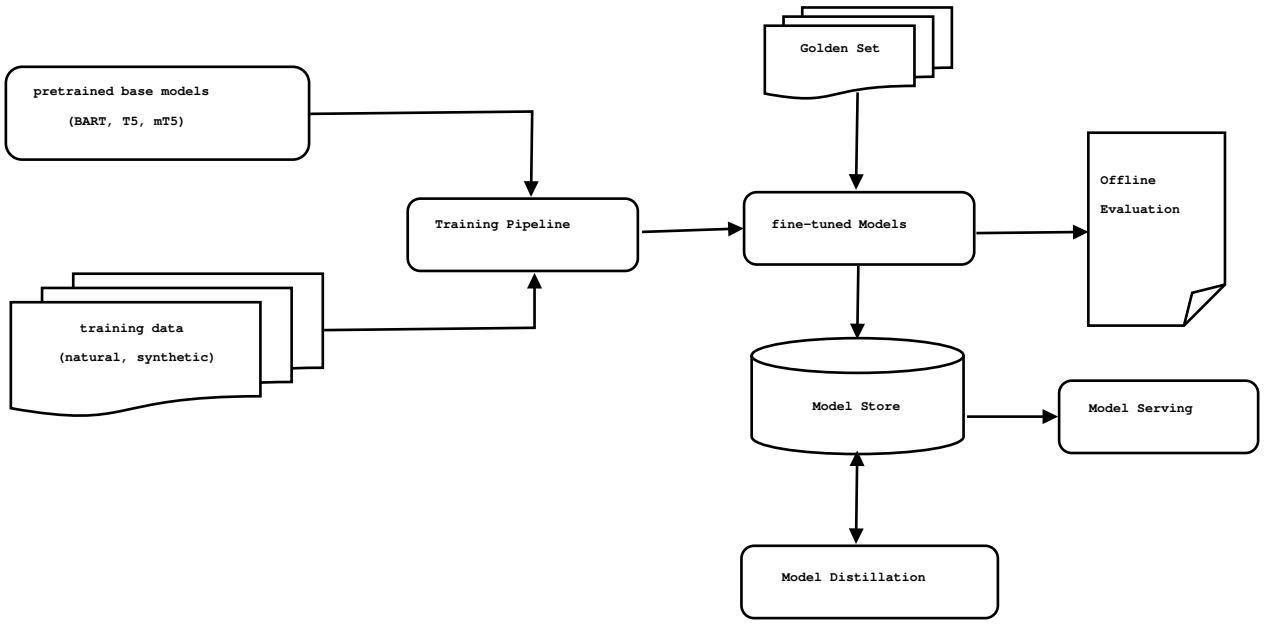


Figure 4: Modular flow of the pipeline

A WORKFLOW OVERVIEW

Figure 4 illustrates the comprehensive pipeline of our proposed solution, which can be broadly categorized into a training part, optimisation and serving part. This is a schematic overview of the interconnected components and not a detailed technical process diagram.

During the training phase, pre-trained models are employed, with each training run utilizing a specific model variant and custom data points from our collected dataset. This process refines the model, evaluated against our golden datasets, with offline metrics detailed in Table 3. Training runtime configurations were iteratively adjusted for relative improvements, and the best empirically observed numbers are reported in the table. Optimal models are checkpointed and stored internally in the Model Store.

Subsequently, models undergo optimization for inference time speed-ups. Model distillation accesses the model store, persisting the distilled models. The distillation process performs a model pruning by dropping decoder layers and subsequent fine tuning them. As mentioned, we have adopted a no-teacher distillation scheme and it was observed to produce better model performance compared to classical teacher-student [10] distillation process.

The model serving platform, a CPU/GPU-based Java framework for deep learning model deployment, uses candidate models on load-balanced endpoints for internal and production use. In our experimentation evaluation, the 2-layer decoder model was deployed through this process, integrating the exposed endpoint with the platform.

B PEFT EVALUATION

We applied LoRA-based parameter-efficient fine-tuning on pre-trained language models. To understand the effect of LoRA tuning approach on model variant and size of model parameters, we picked T5-small, T5-large, BART-base and mBART-large for comparison. We used the same 1 M misspelling-correction samples for training, and fine-tuning each variant for 6 epochs. Table 9 shows the training speed comparison with the size of trainable parameters. Figure 5 shows the precision evaluated on speller US golden set, where Char-Tagger is trained from scratch, T5 and BART variants are fine-tuned and LoRA-tuned respectively. There are 3 observations drawn from this figure: (1) Precision drops on smaller models with LoRA PEFT-tuning. (2) As the number of model parameters increases, mBART-large and t5-large close some gaps with traditional fine-tuning. (3) Even with much smaller adaptor compared to baseline Char-Tagger, PEFT-tuned T5-large still outperforms Char-Tagger trained from scratch, which shows great potential for bigger language models adapted for the task.

C PRE-TRAINED MODELS

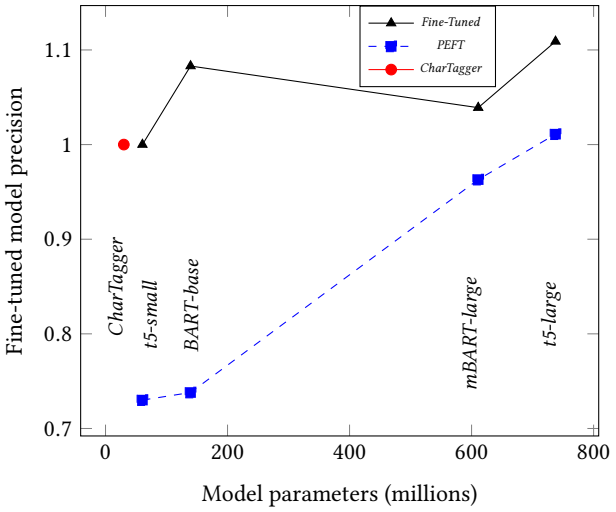
BART [15] has a similar encoder-decoder architecture with a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). The pre-training of BART involves learning through corrupting inputs. The authors had explored various of such perturbations of the input, however the final released version is trained through *sentence shuffling* and *token masking*. The objective behind BART is to achieve causal language modelling. It matches the performance of RoBERTa with comparable training resources on GLUE and SQuAD, achieving new state-of-the-art results on a range of abstract dialogue, question answering and summarizing tasks. The multilingual version[17] or mBART-50 is created using the original mBART-large-cc25 checkpoint by extending its embedding layers with randomly initialized vectors for an extra set of 25 language tokens and then pre-trained on 50 languages.

T5 [22] is an encoder-decoder model and has been pre-trained on a plethora of varied unsupervised and supervised tasks. T5 has

Table 9: Parameter-Efficient Fine-tuning Training Performance.

Base Model	Fine-tuning Approach	#Parameters	Time (s/epoch)	Training Speedup
T5-small	Fine-tuned	60.5 M trainable	2014	1x
	LoRA PEFT-tuned	60.5 M freezed + 0.3 M trainable	1055	1.9X
mBART-large	Fine-tuned	610.9 M trainable	7087	1x
	LoRA PEFT-tuned	610.9 M freezed + 1.2 M trainable	3274	2.2X

Figure 5: Parameter-Efficient Fine-tuned Model Evaluation.The y-axis precision scores are masked.



proven to be performant on a set of tasks including text summarizing, question answering, and text classification. The pre-training includes both supervised and self-supervised training. It is trained on C4[6] as a multi-task mixture of unsupervised and supervised tasks, whereas supervised training is conducted on downstream tasks provided by the GLUE and SuperGLUE benchmarks (by converting them into text-to-text tasks). The multilingual version mT5 [27] covers 101 languages and was pre-trained only in a self-supervised manner. We also explored Flan-T5 [5] which is an improved version of T5 and was developed by scaling the number of tasks and model size, and fine-tuning on chain-of-thought[26] data.

There are two primary differences between T5 and BART: (1) The way input sequences are corrupted in the pre-training phase. T5 does *token masking*, while BART does both *token masking* and *sentence shuffling*. (2) T5 was trained on a multitude of supervised and unsupervised tasks while for BART it was causal training alone, i.e to reconstruct the original corrupted input. In our work, we focus primarily on these two classes of models and take different variants of them, including multilingual models. Table 3 lists the models used in our experiments along with their number of parameters.

[6]https://www.tensorflow.org/datasets/catalog/c4