FLEXICO: Sustainable Machine Translation via Self-Adaptation

Maria Casimiro
INESC-ID / IST & S3D, CMU
maria.casimiro@tecnico.pt

Paolo Romano
INESC-ID / IST
romano@inesc-id.pt

José G. C. de Souza
Unbabel
jose.camargo.souza@gmail.com

Amin M. Khan
INESC-ID
amk@inesc-id.pt

David Garlan
S3D, CMU
garlan@cs.cmu.edu

***Abstract*—Machine Translation (MT) is the backbone of a plethora of systems and applications that are present in users' everyday lives. Despite the research efforts and progress in the MT domain, translation remains a challenging task and MT systems struggle when translating rare words, named entities, domain-specific terminology, idiomatic expressions and culturally specific terms. Thus, to meet the translation performance expectations of users, engineers are tasked with periodically updating (fine-tuning) MT models to guarantee high translation quality. However, with ever-growing machine learning models, fine-tuning operations become increasingly more expensive, raising serious concerns from a sustainability perspective. Furthermore, not all fine-tunings are guaranteed to lead to increased translation quality, thus corresponding to wasted compute resource.**

To address this issue and enhance the sustainability of MT systems, we present FLEXICO, a new approach to engineer self-adaptive MT systems, which leverages (i) ML-based regressors to estimate the expected benefits of fine-tuning MT models; and (ii) probabilistic model checking techniques to automate the reasoning about when the benefits of fine-tuning outweigh its costs. Our empirical evaluation on two MT models and language-pairs and across up to 9 domains demonstrates the predictive performance of the black-box models that estimate the expected benefits of fine-tuning, as well as their domain-generalizability. Finally, we show that FLEXICO improves the sustainability of MT systems when compared to naive baselines, decreasing the number of fine-tunings while preserving high translation quality.

***Index Terms*—Model Fine-tune, Machine Translation, Self-Adaptation, Sustainability.**

I. INTRODUCTION

Machine translation (MT), one of the long-standing tasks in Natural Language Processing (NLP), addresses the problem of automatically translating text from a *source* language to a *target* language [2]. Typical MT applications include adapting products, services, and content to different languages and cultures; and teaching non-native speakers to understand content with language learning tools (e.g., Duolingo [3]). Software engineers have also started exploring the capabilities of MT models for tasks such as automatic program repair [4], code changes [5] and bug fixing [6], or code completion [7].

In recent years, the MT field has embraced deep learning models and sequence-to-sequence architectures [8]. However, in spite of the continuous translation quality improvements observed when generating translations with MT systems, several challenges remain. Specifically, MT systems struggle when translating rare words, named entities, domain-specific terminology, idiomatic expressions, and culturally specific terms. These problems can occur frequently, due to data shift between the data used to train the model and the new content being translated. Mistranslations may be of different severity and cause financial loss or have tremendous implications in the worst case, such as potentially jeopardizing personal safety and health, or even leading to political conflicts [9], [10].

To address these issues, a large body of work in the area of MT has focused on the problem of domain adaptation: creating approaches that allow MT systems to perform better in the face of shifting environments [11]. A well-known approach to address this challenge is fine-tuning the MT model. This involves adapting models pre-trained on large-scale, general-purpose translation domains to improve their performance on specific domains (e.g., medical, legal) by training on domain-specific data. Fine-tuning can be performed for instance to include domain-specific terms or to adjust the style and formality level of the translation output.

However, MT models are growing not only in complexity, but also in size. This continuous growth led to a significant increase in the time, cost, and energy spent fine-tuning them. Furthermore, oftentimes the fine-tuned model performs sub-optimally and thus the fine-tune could have been avoided. Finally, although improving the fine-tuning approaches makes them more sustainable and/or computationally efficient, it does not prevent superfluous MT model updates. This calls for a generic solution that is applicable to all types of MT systems and that contributes both to improving user experience — via higher translation quality — and to more sustainable systems — by avoiding unnecessary fine-tunings.

To balance these conflicting goals, this work explores the idea of *Self-Adaptive Machine Translation Systems*: exploit self-adaptation mechanisms to determine whether the expected benefits of fine-tuning the MT model (quantifiable via the

Support for this research was provided by Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) and ANI through the Carnegie Mellon Portugal Program under Grant SFRH/BD/150643/2020 and via project with reference UIDB/50021/2020. Part of this work was supported by the EU's Horizon Europe Research and Innovation Actions (UTTER, contract 101070631) and innovation programme under Grant Agreement No 101189689, and by the Portuguese Recovery and Resilience Plan through project C645008882- 00000055 (Center for Responsible AI). This work used the Bridges-2 GPU and Ocean resources at the Pittsburgh Supercomputing Center through allocation CIS230074 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program [1], which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

2157-2321/25/$31.00 ©2025 IEEE
DOI 10.1109/SEAMS66627.2025.00024

improvement to MT evaluation metrics [12]–[14]) outweigh the costs of the fine-tuning process. Hence, the goal of a self-adaptive MT system is to maximize the quality of the automatically translated sentences while minimizing the costs incurred to create the best MT model.

To pursue these goals, we introduce FLEXICO, a new approach to engineer self-adaptive MT systems. FLEXICO leverages black-box predictors to estimate the expected benefits of fine-tuning an MT model, and probabilistic model checking techniques to determine when to execute the fine-tuning with the goal of optimizing quality of service (QoS).

Two main challenges hinder the adoption of existing self-adaptation frameworks [15] to the MT domain. First, existing approaches target much simpler ML components, namely classifiers, whereas MT models solve sequence-to-sequence problems. The difference in the underlying ML task (classification vs sequence-to-sequence) renders current approaches for formally modeling ML components [15] infeasible for MT models. To tackle this challenge, FLEXICO introduces the *MT Quality Matrix* (MTQM) a new abstraction that enables model checking tools to reason about the variations in translation quality of MT models.

The second challenge is how to estimate the expected benefits of fine-tuning MT models. Specifically, we propose a set of MT-specific features, capturing a myriad of linguistic characteristics (lexicographic, semantic), and investigate how these features impact the predictive quality of the models that estimate the expected benefits of fine-tuning MT models.

The main contributions of this work are:

- FLEXICO: an approach to engineer self-adaptive machine translation systems;
- An exploration of the impact of MT-specific features on the prediction of the benefits of fine-tuning MT models;
- Two datasets characterizing the impact of fine-tuning MT models, along with features representing the system and its environment upon fine-tuning.

FLEXICO's source code is available at https://github.com/marialcasimiro/flexico.

II. BACKGROUND

Self-Adaptive Systems and Probabilistic Model Checking. Self-adaptive systems react to changes in the environment by adapting themselves such that they optimize system utility [15]–[18]. Adaptation actions, also known as *tactics*, typically have varying costs and offer different benefits to the system. Thus, self-adaptive systems rely on planners to select the tactic that optimizes system utility.

A common approach for instantiating the planner is via probabilistic model checking techniques [19], which require a formal model of the system under adaptation and a formal property encoding the goal of the system. In the self-adaptive systems literature, probabilistic model checking techniques are often used, especially via formalisms such as Markov Decision Processes [20], for two main reasons. First, they support non-deterministic state transitions: states where multiple actions can be executed and it is up to the model checker to determine which action leads to the satisfaction of the formal property under verification. Self-adaptive systems leverage this capability to have the model checker create the optimal adaptation plan. Second, formal models support associating rewards with states or state transitions, which enables reasoning about system behaviors (e.g., charging a cost when the MT model is fine-tuned). By encoding system utility as a set of rewards and by specifying the formal property as a function of these rewards, the model checker optimizes system utility.

Machine Translation Evaluation. A key challenge in the MT domain is the automatic evaluation of translation quality, as there is usually no single correct translation hypothesis. Reference translations are typically created by translation experts and are thus scarce, not scalable to produce, and costly to obtain. Given the challenging nature of automatic translation evaluation, multiple metrics are commonly used to measure precision, fluency, and overall translation quality, such as COMET22 [12], its variant that does not require references COMETKiwi [21], chrF [13], and BLEU [22] henceforth referred to as its most recent implementation, SacreBLEU [14]. chrF implements a string overlap algorithm at the character-level that checks an hypothesis translation against a reference translation, assuming that a good translation is a string match with the reference translation. Differently, COMET22 calculates the similarity between the source sentence, its translation and a reference translation using embedded representations from a multilingual large language model[1], thus considering similarity between sentences rather than strict string matching as string overlap metrics such as chrF and BLEU. In practice, a combination of these metrics is often used to get a comprehensive evaluation of MT models, as each metric has its strengths and weaknesses [24]. Recent work investigated how differences in MT metric scores, both among different and the same MT metrics, are perceived by humans [25]. This study is particularly relevant as the performance variations that lead to human perceivable differences can be used to quantify the benefits that a MT model fine-tune should yield.

III. RELATED WORK

Adaptive ML-enabled Systems. As defined by Lewis et al. [26], ML-enabled systems are "software systems that rely on one or more ML components". Their widespread use [27], [28] opened research questions related to how to adapt existing software architecture methodologies [26], [28] and software development processes [29] to cope with the challenges raised by ML-enabled systems. Specifically, in the domain of self-adaptive systems, recent work has proposed approaches that deal with adapting ML components of ML-enabled systems [15], [28], [30]–[34] by retraining the underperforming ML component [15], [32] or replacing it by one

[1]Embeddings or embedded representations are transformations of words/sentences into lower dimensional spaces, preserving the semantic relationships between the transformed words/sentences [23].

that is expected to perform better under the current environmental conditions [33], [34]. Differently from our work, which focuses on fine-tuning of MT models, existing approaches are applicable only to simpler ML tasks, namely classification problems, and consider different adaptation tactics.

Domain Adaptation for MT. Many approaches have been proposed to adapt MT models for one or more specific textual genres or domains [11]. The self-adaptation methodology described in this paper could be applied to other domain adaptation approaches; however, we focus on fine-tuning, which became popular with neural machine translation [35]. Fine-tuning MT models involves adapting models pre-trained on large-scale, general-purpose translation domains to improve their performance on specific domains (e.g., medical, legal) by training on new, domain-specific data.

Existing fine-tuning approaches for MT domain adaptation simply adapt the MT model periodically, evaluate its performance, and then decide whether to put it into production or discard it. FLEXICO is the first approach to dynamically determine whether the expected benefits of fine-tuning outweigh the costs of adaptation and improve overall system utility, preventing superfluous fine-tunings.

Sustainable Machine Learning. Research on sustainable AI has gained traction recently, highlighting the role that ML plays in global warming and calling for greater accountability and transparency in reporting the environmental costs of model training [36]–[44]. Recent research efforts have focused on three main areas: (i) studying the energy consumption associated with training/fine-tuning ML models [37]–[39] (ii) creating tools for self-reporting and evaluating the energy consumption and carbon emissions of training ML models [40]–[42]; (iii) researching green architectural tactics [43] and creating systems that account for the ML model's energy consumption when selecting the ML model to use [44]. FLEXICO shares the same sustainability goals as these works and pursues them by preventing unnecessary fine-tuning of MT models.

Limitations of existing work. The work that is most closely related to FLEXICO is the framework for adapting ML-enabled systems, proposed by Casimiro et al. [15], which targets ML classifiers (i.e., their task is to determine which class, among a finite set of alternatives, an input belongs to). This method, analogously to FLEXICO, relies on probabilistic model checking to determine the expected optimal adaptation strategy. Specifically, they sub-divide the problem into two: (i) estimating the expected benefits of adapting an ML model and (ii) computing the expected optimal adaptation strategy.

For sub-problem (i), the authors create Adaptation Impact Predictors (AIPs): models trained to predict the quality of an ML classifier at time $t+1$ after re-training and not retraining the ML classifier at time t. For sub-problem (ii), the authors use probabilistic model checking methods and formally model the ML classifier via a confusion matrix [45]. This provides a high-level abstraction over the implementation details of the ML classifier, enabling model checking tools to reason about the evolution of the predictive quality of the ML classifier upon adaptation and compute metrics of interest such as accuracy.

Thus, coupling the AIPs (predict the expected benefits of each adaptation tactic) and model checking (verify a pre-specified property – e.g., maximize system utility – and create an optimal adaptation strategy), the framework proposed by [15] was shown to be effective when self-adapting systems that rely on ML classifiers in the fraud detection domain.

However, this methodology is not applicable to MT systems: confusion matrices cannot be used with MT models (i.e., sequence-to-sequence problems) as there are no classes and accuracy is not a valid MT evaluation metric since there is no single correct translation. Thus, the **first challenge** that arises when creating self-adaptive MT systems corresponds to **"how to formally model MT models?"** (§ V-B). Second, previous work proposed a set of features to predict the benefits of retraining that are tailored to the fraud detection domain and cannot be reused directly for MT systems. Thus, a **second challenge** corresponds to **"how to predict the benefits of fine-tuning MT models?"** (§ V-C, V-D, and V-E).

IV. RUNNING EXAMPLE

Consider a machine translation system tasked with translating news from a popular news-source, *Znn.com* [46], [47], from a *source* language into a *target* language.

As time passes and new events occur, the distribution of data that the model needs to serve at inference time differs from the distribution of data used to train it. Due to this variability, the MT model might have a decline in translation quality while it tries to cope with new terms, morphological constructions, or different word meanings, among other linguistic phenomena [11]. Further, the popularity of different news domains (e.g., sports, politics) also varies over time: for instance, if an important election is coming up, most news will likely be about politics; yet, if the Olympics are starting, there might be a surge of sports news. When the news domain changes, the MT model may require adaptation such that higher translation quality for the emerging domain is achieved.

The need for adaptation is evaluated based on the expected benefits of fine-tuning the MT model and the cost of doing so, such that the utility of the news website is maximized. Specifically, we focus on two main quality attributes: (i) variation of the translation quality upon a model fine-tuning $\Delta\mathcal{Q}$ – provided via a combination of state-of-the-art MT evaluation metrics such as COMET22 [12] and chrF [13]; and (ii) costs incurred by the news website. The costs incurred by *Znn* are defined in terms of the expected and actual $\Delta\mathcal{Q}$ as follows: (a) if the MT model is not fine tuned (i.e., *nop*) and the actual $\Delta\mathcal{Q}$ had the MT model been fine-tuned exceeds a pre-defined threshold T, the system incurs a missed opportunity cost $\mathcal{O}$; (b) if the MT model is fine-tuned, the adaptive action will incur a cost $\mathcal{F}$, capturing, e.g., cloud provisioning or energy costs; (c) if the MT model is fine-tuned, but its performance improvement is sub-par, i.e., below T, the system will incur a penalty $\mathcal{R}$, which embodies a "regret" cost (e.g., from the sustainability perspective) for having performed a superfluous

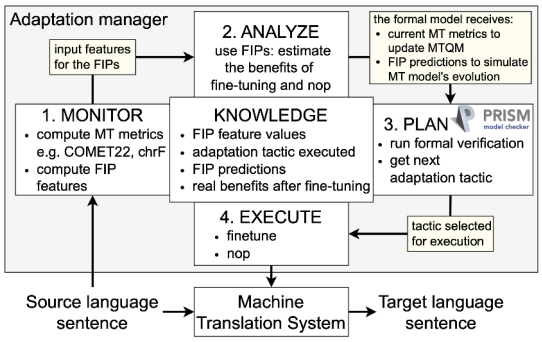


Fig. 1: Self-Adaptive Machine Translation System.

update. Based on these costs, and on the set **M** of MT metrics of interest, n news domains each with its own translation quality $\mathcal{Q}_{i,m}, 1 \leq i \leq n, \forall m \in \mathbf{M}$ and importance/weight w_i, the system utility $SysU$ of the MT system is defined as:

$$\text{OPTIMIZE}(SysU) = \textit{choose tactic that minimizes total cost}$$

$$\text{OPTIMAL TACTIC} = \underset{\text{tactic} \in \{\text{finetune, nop}\}}{\arg\min} \text{COST(tactic)}$$

$$\text{COST(nop)} = \sum_{m=1}^{\mathbf{M}} \mathcal{O} \times \text{BOOL}((\frac{1}{n}\sum_{i=1}^{n} w_i \times \Delta\mathcal{Q}_{im}) > T)$$

$$\text{COST(finetune)} = \mathcal{F} + \sum_{m=1}^{\mathbf{M}} \mathcal{R} \times \text{BOOL}((\frac{1}{n}\sum_{i=1}^{n} w_i \times \Delta\mathcal{Q}_{im}) < T)$$

The goal of the self-adaptive MT system is thus to minimize the overall costs incurred. The following section elaborates on how FLEXICO pursues this goal.

V. FLEXICO

This section introduces FLEXICO, a generic approach to create Self-Adaptive Machine Translation systems (SAMTS). The goal of an SAMTS is to optimize its system utility, which is intrinsically context and system dependent. To account for this dependency, one of the key requirements for the SAMTS is that it should be ***modular***, such that it can be straightforwardly modified to cater to different scenarios and system utility definitions. Additionally, and although the *Znn* use case considers a single tactic to adapt the MT model, there are more actions (i.e., tactics) that one may wish to consider (e.g., querying different MT models depending on the domain of a news article – sports vs politics). For this reason, the SAMTS should be ***extensible***. Lastly, to balance the expected benefits of fine-tuning the MT model and the costs of doing so, the SAMTS needs to predict the expected benefits of fine-tuning and their impact on overall system utility. This calls for an ***abstract*** solution to formally model any MT model.

In the following sections we (i) provide an overview of the components required by SAMTS (§ V-A); (ii) explain how FLEXICO formally models MT models (§ V-B), addressing challenge 1; and (iii) how the benefits of fine-tuning are estimated by leveraging fixed test-sets (§ V-C) and MT specific features (§ V-D) via Fine-tuning Impact Predictors (FIPs) (§ V-E), addressing challenge 2.

A. Self-Adaptive Machine Translation Systems

FLEXICO is a framework to create SAMTS that automates the decision of when to fine-tune an MT model. As shown in Figure 1, the decision of whether to fine-tune is guided by a MAPE-K loop [48] which: 1) **M**onitors the MT system, keeping track of the sentences it receives and the translations it outputs – at this stage the adaptation manager computes the quality of the MT model currently in use via MT-specific metrics (e.g., COMET22 [12], chrF [13], SacreBLEU [14]) and features for estimating the expected benefits of fine-tuning; 2) **A**nalyzes the expected benefits/costs[2] of each tactic, which are predicted by the FIPs, and sends these predictions to the 3) **P**lan component that creates an adaptation plan as follows. The FIP predictions and the current MT metrics (computed by Monitor) are sent as input variables to the formal verification process. The formal model has place-holders for these variables and when formal verification is triggered, the formal model is (re-)built with the current values for these variables. The formal verification process then selects the next adaptation tactic to execute based on the system utility definition and on the expected benefits of adaptation; 4) this triggers the **E**xecution of the selected adaptation tactic – fine-tune the MT model or keep using the current one; 5) finally, **K**nowledge stores the outcomes of the MAPE process (i.e., feature values, adaptation tactic executed, expected benefits of the adaptation and real benefits when the tactic selected is to fine-tune the MT model). Thus, the key components required to instantiate SAMTS are:

- *Feature monitor:* to compute the features required for estimating the benefits of fine-tuning (**M**onitor);
- *Fine-tune Impact Predictor* (FIP) [15]: to estimate the expected benefits of fine-tuning the MT model (**A**nalyze);
- *Formal model of the MT system*: to reason about and extract the expected optimal adaptation tactic via probabilistic model checking (**P**lan);
- *Actuators*: to execute the selected tactic (**E**xecute).

B. Formally Modeling MT Models

Leveraging probabilistic model checking methods to instantiate self-adaptive MT systems and determine the expected optimal adaptation tactic (fine-tune or not fine-tune) requires creating formal models of the systems under adaptation. As described in Section II having such a model is required by probabilistic model checkers. However, in order to ensure the tractability of the model checking process, a key challenge is determining an adequate abstraction level to describe complex MT systems (e.g., Transformers) in the formal model.

In FLEXICO, we address this problem by proposing an abstraction called *Machine Translation Quality Matrix* (MTQM). The MTQM is a matrix in which each column represents a domain that is relevant to the MT system, and each row represents a specific MT evaluation metric (e.g., COMET22 [12],

[2]Predicting the costs of fine-tuning models is quite straightforward given the availability of historical data on prior runs [49], [50], so our focus is on the less explored and more challenging problem of predicting the benefits.

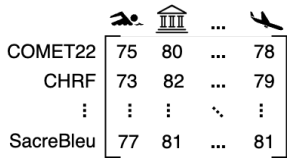


Fig. 2: Machine Translation Quality Matrix (MTQM): columns represent relevant domains to the MT system; rows represent different MT evaluation metrics (e.g., COMET22 [12], chrF [13], SacreBLEU [14], which vary in the range $[0, 100]$).

chrF [13], SacreBLEU [14]). Figure 2 shows an example of this matrix, with domains like sports, finance, and tourism.

This design choice is motivated by our goal to enable automatic reasoning about the impact that fine-tuning (or not) the MT model has on translation quality. Given its relative simplicity, the MTQM is lightweight and compact enough to be employed in model checking tools, while still allowing reasoning in a robust way (using multiple MT quality metrics) over a range of alternative domains. This is crucial to enable the employment of complex, but easily customizable, utility functions, such as the one considered by *Znn* (see § IV).

C. Estimating the Expected Benefits of Fine-tuning

To estimate the expected benefits of fine-tuning, FLEXICO introduces Fine-tuning Impact Predictors (FIPs). FIPs build on and specialize the idea of AIPs [15] by introducing several novel methodological aspects to be effectively employed in the context of MT systems. These include:

1. Identifying a set of **MT specific features** to capture data-set shifts in the input and output of MT models, while achieving an acceptable trade off between effectiveness (i.e., predictive power) and computational costs (which are incurred both when training and querying the FIPs). See Section V-D.

2. Introducing a different methodology to estimate the variations of the predictive quality of the MT model: differently from existing work [15], FLEXICO does not attempt to predict the expected performance of the MT system at a specific time in the future, which entails having to account for the type of inputs the system will be subject to in the future. Instead, we **evaluate the MT model on fixed reference test-sets** that represent the expected input distributions across domains.

A test set is a collection of sentences in the source language along with a corresponding reference translation in the target language. Each domain usually has its own test set with sentences sampled from a large pool of domain-specific data. The test sets are disjoint from any data used to train or fine-tune models. When a test set is the same across different steps of fine-tuning, we call it a fixed test set. Fixed test sets have been used for a long time in academic benchmarks in the MT domain (WMT [51], [52] and IWSLT [53] shared tasks).

By relying on fixed test-sets FLEXICO avoids having to predict the quality of the MT model at a specific time in the future, thus decoupling the problems of (i) estimating future evolutions of the input distributions to the MT model and (ii) predicting how the translation quality varies due to a model fine-tune. Given the impossibility of perfectly predicting future inputs, it is indeed desirable to put in production robust MT models that generalize to different contents and provide high quality for the widest plausible range of inputs. The fixed test sets aim precisely at describing such a wide range of input domains for MT systems. By using multiple fixed test sets, representative of various domains, we are able to understand what model performs best across domains.

3. By leveraging the notion of domain-specific test sets, we propose two alternative FIP designs: domain-agnostic and domain-specific FIPs. Both variants predict how the translation quality varies after a fine-tune, however domain-agnostic FIPs predict for any target domain, while domain-specific FIPs predict for a specific domain. These variants lead to different trade-offs regarding generalization to unknown domains versus predictive quality on known domains (§ V-E).

D. MT-specific Features for FIPs

The AIPs leverage three types of features, namely: (i) *basic features*, that capture information on the current state of the ML model and simple statistics on environmental events (e.g., amount of new sentences to translate); (ii) *output characteristics features*, that capture how the ML model is performing in the current operational environment (e.g., current COMET22 score of the MT model); and (iii) *input characteristics features*, that capture different types of environment shifts.

Out of the three feature groups, the basic features are the most re-usable, given their domain-agnostic nature. The remaining feature groups are strongly domain (and possibly application) dependent. In this work we propose and evaluate the use of a specialization of these two sets of features for MT models: *content-aware* and *MT-quality* features.

Content-aware features. These features are inspired by work on statistical MT [54] and are designed to capture environment shifts by analyzing textual information at the semantic and lexical levels. Concretely, these features compare an "old data set", i.e., sentences with which the MT model has been fine-tuned, and a "new data set", i.e., sentences (previously unseen by the MT model) with reference translations that have become available since the last model update and that can be used to fine-tune the model again. We leverage the concepts of n-grams [55], embeddings [23] and sentence overlaps:

- n-grams capture information about the context, word order, phrase structure, and linguistic patterns in sentences or texts. By computing the JensenShannon [56] distance between the n-gram distributions in the new and old data sets, we capture variations at the lexical level (e.g., if sentences started having more/less typos in the new data).
- Embeddings capture variations at the semantic level. They are representations of sentences in lower-dimensional spaces that preserve the semantic relation between the transformed words/sentences [23]. Comparing embedding distributions in the new and old data sets (e.g., through cosine and Euclidean [57] distances) allows us to capture new domains that arise in the new data, thus signaling shifts in the environment.

• Sentence overlaps allow us to capture information at the lexical level by measuring common words between the new and old data sets. We thus capture semantics information and detect new domains: if the overlap between new and old data is low, that likely means that the environment is changing.

MT-quality features. These features capture the translation performance of the MT model, both in the current environment (recently translated sentences) and in fixed MT test sets. Specifically, we consider the following MT metrics: COMET22 [12], chrF [13], SacreBLEU [14], COMET22Kiwi [21]. All MT-quality features except for COMET22Kiwi assume the availability of references (i.e. human translations) for the data available for fine-tuning. Thus, including exclusively COMET22Kiwi among the features used by FIPs (or avoiding the use of MT-quality features) allows FLEXICO to estimate whether to fine-tune an MT model even before incurring the cost of obtaining references for the new sentences. In this case, FLEXICO enables an additional dimension of cost reduction: beyond saving the cost of superfluous fine-tunings, FLEXICO also saves the costs of obtaining references for sentences that are not expected to be used to improve the MT model.

E. Fine-Tuning Impact Predictors (FIPs)

This section presents the methodology to build FIPs, which are regression models that predict the expected variation of MT quality metrics due to a fine-tune.

FIPs are trained using an FID (Fine-tune Impact Dataset) created as follows. We fine-tune the target MT model N times and, for each fine-tune, evaluate the translation quality of the fine-tuned model on the fixed test sets using MT metrics. This yields the MTQM for each fine-tuned model $M_i, i < N$. When fine-tuning model i, we consider all the data available up to i. This includes data with which model $i-1$ was fine-tuned as well as new data collected since. Next, for each pair of fine-tuned models M_i, M_j, where $i < j$, we generate a new sample for the FID by computing: (i) the features described in Section V-D, assuming that the *old data* is the data used to fine-tune M_i and that the *new data* is the data collected between times i and j; (ii) the variation in translation quality, given as the difference between the MTQMs for M_j and M_i. Note that this approach is quite efficient as it generates FIDs of size $O(N^2)$ based on N fine-tunings. Two alternative FIP variants, exploring different trade-offs between generalization and prediction quality, can be created with the FID:

Domain-specific FIPs. As the name suggests, these FIPs estimate the expected difference in MT performance for a specific domain (e.g., sports or finance). With these FIPs, if our goal was to predict the variation in COMET22 and in the sports domain due to a fine-tune, we would have to query the FIP that had exclusively been trained with examples describing COMET22 variations and evaluated based on the sports domain test set. For *Znn*, this approach requires having $\mathbf{M} \times n$ FIPs. Having more FIPs is more expensive (more models to train), but since they are tailored to each domain, they are expected to achieve higher predictive quality.

TABLE I: Performance of the G-FIPs and average performance of the S-FIPs (across domains and with the best feature set for each domain).

Use Case	Model	MAE		PCC	
		Generic	Specific	Generic	Specific
En-Zh	RF	0.0049	**0.0032**	76.62	**86.44**
	XGB	0.0053	**0.0048**	76.51	**84.83**
	Lin	0.0066	**0.0054**	68.30	**89.07**
En-Fr	RF	**0.0013**	0.0021	30.09	**65.63**
	XGB	**0.0012**	0.0016	33.72	**62.33**
	Lin	0.0018	**0.0016**	25.11	**70.49**

Generic FIPs. These are more ambitious predictors that are domain-agnostic and can be queried to predict the variations in translation quality with respect to an *arbitrary* domain, including domains not seen in the training phase. Generic FIPs exploit the fact that both the content-aware and MT-quality feature sets encode information about the different domains. In fact, the content-aware features describe data shifts between the new-data set and the domain test set, and the MT-quality features quantify the performance of the MT model for the domain test set. This can be exploited by FIPs to learn how to adjust their predictions to the characteristics of the different domains. We train generic FIPs with a training set obtained by merging the training sets of domain-specific FIPs. This way, during training, generic FIP are exposed to data characterizing how translation quality varies across different domains.

VI. EVALUATION

This section investigates the following research questions:

RQ1 How accurately can the Fine-tune Impact Predictors (FIPs) predict the benefits of fine-tuning? (§ VI-B)
RQ2 What features lead to higher FIP quality and how expensive is it to train and query the FIPs? (§ VI-C)
RQ3 Can the FIPs generalize across domains? (§ VI-D)
RQ4 What are FLEXICO's utility gains? (§ VI-E)
RQ5 Is FLEXICO tractable for online adaptation? (§ VI-F)

A. Use Cases

We consider two use cases accounting for two languages pairs and for differing types/contents of data. For both use cases, the MT model is evaluated on fixed test sets.

English-Chinese (En-Zh). This use case, inspired by *Znn*, uses the OpusMT model [58]–[60] available through HuggingFace and LDC's 'Hong Kong News Parallel Text' [61], which contains news data from 1997 to 2000. To create the fixed test sets, we used chatGPT [62] to assign domains, which are not present in the original dataset, to each news article as follows. To bound chatGPT's output, we created a list of news domains that was passed to chatGPT. This list was created by browsing the news categories of multiple online news sites and by asking chatGPT to provide a list of news domains. With this list, we prompted chatGPT to give us the top 3 domains (from the list provided) that it associated with each news article, along with a number between 0 and 100 for each domain selected, for us to sort the domains from most to least important.

TABLE II: Performance of the specific FIPs when predicting COMET22 with RF models for the En-Zh use case.

Test set	Entertainment		Environment		Finance		Governance		Health & Wellness		Sports		Travel & Tourism	
Feature Set	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC
All	0.0098	61.14	0.0110	92.08	0.0078	88.82	0.0025	95.62	0.0021	90.63	0.0046	76.31	**0.0033**	84.75
All + Kiwi	0.0102	61.73	0.0115	91.74	0.0082	88.61	0.0026	95.55	0.0022	90.73	0.0052	75.30	**0.0033**	84.51
Basic	0.0107	61.52	0.0096	91.88	0.0065	91.20	0.0028	95.15	0.0022	90.61	0.0042	74.98	0.0034	85.08
Basic + ContAware	0.0104	60.64	0.0096	91.50	0.0064	**91.29**	0.0026	95.06	0.0019	90.80	0.0052	74.41	0.0034	84.21
Basic + Kiwi	0.0105	62.65	0.0114	92.21	0.0082	86.75	0.0025	95.89	0.0025	89.91	0.0040	76.11	**0.0033**	85.60
Basic + MTQual	0.0100	61.47	0.0109	92.21	0.0076	87.11	0.0023	**95.97**	0.0023	89.49	0.0036	76.63	0.0035	**85.66**
ContAware	0.0066	48.79	0.0054	80.02	0.0106	89.26	0.0026	93.72	0.0018	89.37	0.0051	69.02	0.0039	80.31
ContAware-no-ngrams	0.0032	**64.65**	0.0040	**95.66**	**0.0018**	89.92	**0.0015**	95.00	**0.0011**	**91.94**	**0.0020**	**79.90**	0.0040	85.15
ContAware + Kiwi	0.0064	51.67	0.0061	78.82	0.0120	87.87	0.0027	93.93	0.0022	87.84	0.0048	70.12	0.0037	80.34
ContAware + MTQual	0.0061	50.50	0.0052	79.84	0.0118	87.59	0.0028	93.75	0.0020	89.16	0.0044	69.07	0.0038	80.85
MTQual	**0.0026**	-13.14	**0.0026**	-4.65	0.0040	-48.10	0.0025	-9.08	0.0023	-18.69	0.0033	-16.65	0.0068	-28.18

After getting domains for all articles, we isolated the articles of year 2000, grouped them by the domain with the highest score, and selected the domains with the most articles and that could be more different from each other to create 7 fixed test sets. Note that `Governance` was not an original category, and we created it by merging three categories: `politics`, `Law and legal issues`, and `Business and economy`.

The remaining years of the Hong Kong News dataset [61] (i.e. years 1997, 1998, and 1999) are used to (i) create the FID that is used to train the FIPs (years 1997 and 1998) and (ii) test the FIPs (year 1999). When computing the content-aware and the MT-quality features for the FID, we used a sample of 10% of the data available to fine-tune the model to compute these features. The FID was created by fine-tuning the OpusMT model 147 times, which yields 10585 FID samples.

English-French (En-Fr). For this use case, we employ the Opus datasets [63] and the Tatoeba Opus MT model [64], [65]. To create our own dataset out of the Opus dataset, we compared the training data of the Tatoeba Opus MT model with each Opus dataset, to guarantee that we only used Opus datasets whose data had not previously been used to train the base MT model. From this analysis, we obtained 7 datasets from domains such as religion, legal and finance.

After gathering these datasets, and assigning some of them only for test purposes due to their smaller size (less than 2000 sentences), we split the remaining ones into chunks of 10000 sentences. The "left-over" of these fine-tuning sets was used as fixed domain test sets for MT quality evaluation. Finally, we randomly ordered the fine-tuning data chunks.

To compute the content-aware and the MT-quality features for the FID, we used a sample of 10% of the data available for each fine-tune to compute these features. Overall, we use 70% of the FID for training the FIPs and 30% for evaluation. The FID was created by fine-tuning the Tatoeba Opus MT model 105 times, yielding 5356 FID samples.

B. FIP Prediction Quality (RQ1)

To evaluate the prediction quality of the FIPs we conducted a study varying three key aspects of the FIP building process: (i) 3 model types – XGBoost (XGB) trees [66], Random Forest (RF) and Linear (Lin) regressors from the SKLearn Library [67]; (ii) 3 feature sets – basic, content-aware, all; (iii) 2 versions of the target – domain-specific or generic. The quality of the FIP predictions is evaluated with Mean Absolute Error (MAE) and Pearson Correlation Coefficient (PCC).

Let us start by comparing, in Table I, the prediction quality of the generic and domain-specific FIPs for COMET22, trained with the best performing sub-set of features among the ones presented in Section V-D (this set can be discovered, e.g. using a validation set). We report the MAE and PCC across all domains, for both use cases and considering the three above-mentioned model types for the FIPs. The use case for which both FIP variants achieve the highest PCC is En-Zh, where S-FIPs and G-FIPs achieve an average PCC of 89.07 and 76.62 for the modeling approaches with the best performance (Lin and RF, respectively). En-Fr is more challenging for the FIPs, since it encompasses data originating from a broader set of domains and since the FIPs have been trained with approximately half the data available for the En-Zh use case (see § VI-A). In fact, for En-Fr, the PCC of the S-FIPs and G-FIPs drops to 70.49 (Lin) and 33.72 (XGB), respectively.

Overall, in both use cases, the G-FIPs achieve lower performance than the S-FIPs, both in terms of MAE and PCC. The superiority of S-FIPs is not surprising, as being queried solely in the domain in which they were trained inherently favors them. The performance gap between the two FIP variants is not very large in the En-Zh use case (the PCC of the best G-FIPs is 76.62%), but it is quite large in the En-Fr use case (where the PCC of the best G-FIPs drops to 33.72%). This can be explained by recalling that the En-Fr use case spans a broader range of domains, creating more challenging conditions for domain-agnostic predictors such as G-FIPs.

C. FIP Feature Importance and Cost (RQ2)

We now focus on the S-FIPs that achieved higher predictive performance than the generic FIPs for both use cases. Specifically, we analyze how different combinations of feature sets impact the predictive performance of S-FIPs.

For this study, we consider: (i) the 3 feature sets presented in Section V-D (i.e., Basic, Content-aware, MT-quality); (ii) all 3 feature sets together (i.e., All); (iii) combinations of the 3 feature sets (Basic + Content-aware, Basic + MT-quality, Content-aware + MT-quality); (iv) combinations of the COMET22Kiwi metric with the Basic, Content-aware, and both to evaluate the expected quality should no reference sentences be available; (v) a version of the content-aware

features that does not account for the n-grams (content-aware-no-ngrams). As we will show later, this is motivated by the high computational cost of computing n-gram features compared to the remaining content-aware features.

In the En-Zh use case, the content-aware-no-ngrams feature set (i.e., sentence overlaps and embedding features) performs best across the domains considered. In the En-Fr use case (see appendix [68]) we see much stronger variations, with different feature sets having substantially different performances across domains. On average, the best-performing feature set is Basic + MTQual, followed by content-aware + MTQual.

Further, looking at the feature set combinations that use MT quality metrics[3], we see that for En-Zh these features are typically not needed: only 2 out of the 7 domains achieve the highest PCC with a feature set using MT-quality aware features (Governance, Travel & Tourism). Differently, for En-Fr, only 1 out of 7 domains (Elitr) does not require MT-quality aware features to achieve the highest predictive performance.

This suggests that for scenarios with less heterogeneous domains, MT-quality features are less important to the S-FIPs. In such scenarios, S-FIPs can be used to decide whether to ask human annotators for references (see § V-D). Ultimately, the best feature set for the FIPs is dataset/domain dependent and can be discovered via classical feature selection techniques.

Feature computation cost. The features employed by the FIPs impact how much it costs to query them (when a FIP is queried, its input features need to be computed), so we investigated how much it costs to compute each feature set. The results (see appendix [68]) are upper bounds of the actual feature computation times, as the feature computation process was not optimized and other users could concurrently access the machines, possibly introducing noise in the measurements.

Specifically, for the En-Zh S-FIPs which achieve the highest predictive performance with the content-aware-no-ngrams feature set, this represents cost savings of around 80× w.r.t. the cost of fine-tuning the MT model. We see that the trends for the En-Fr use case are similar, albeit smaller, allowing for cost savings of around 40× when considering the Basic + MTQual feature set and comparing to the fine-tune cost.

Note that due to resource limitations, we considered relatively small MT models. Since the cost of feature computation is independent from the MT model, unlike the fine-tuning cost that is strongly dependent on the MT model's size, for larger MT models we expect to see even larger cost savings.

Computational cost of building the FID. Finally, we analyze the cost of creating the FID. As described (§ V-E), our FID was obtained by fine-tuning the MT models 147 and 105 times (69 and 57 fine-tunings for training the FIPs, En-Zh and En-Fr, resp.). Despite the methodology for FID generation producing $O(n^2)$ samples out of n fine-tunings, fine-tunings can be costly and the cost of feature computation also adds up. To understand whether we can reduce the cost associated with creating the FIDs, we explore how much data is actually needed to train accurate FIPs, by conducting a sensitivity study analyzing the FIPs' prediction quality (measured via PCC and MAE) as we vary the size of the G-FIPs' training set.

[3]All MT metrics, except for COMET22Kiwi, require reference translations, which are obtained by paying human annotators.

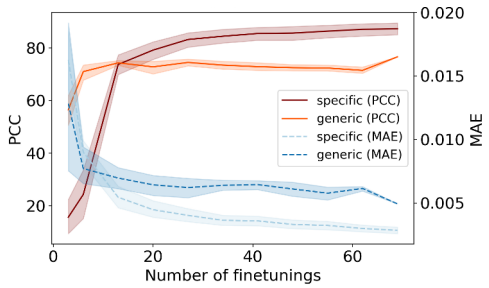


Fig. 3: MAE and PCC for G-FIPs (RF, En-Zh, COMET22), varying the number of finetunings to create the FID.

We fixed the G-FIP feature set to *Basic + MTQual* (i.e., the feature combination of the best performing G-FIPs) and tested with the random forest model. We repeated each sample size 10 times for reliability. Figure 3 displays the results and shows that decreasing the number of fine-tunings performed to 20% (≈ 13) of the original number of fine-tunings in the train set (69) still guarantees quite high MAE and PCC.

D. FIP Domain Generalization (RQ3)

To evaluate the domain generalization capability of the G-FIPs, we conduct a leave-one-out cross-validation study by leaving each domain test set out of the G-FIPs' train-set and then evaluating the predictive performance of the FIP on the test set that was left out. Table III shows the results obtained for the specific case of random forest FIPs for the En-Zh use case. We focus this study on the En-Zh use case since we have already established in Section VI-B that the G-FIPs achieve poor performance in the En-Fr use case due to its challenging characteristics (S-FIPs are recommended in such scenarios). We still report the full results in the appendix [68].

Overall, we observe very high PCC values (above 85% and up to 95%) for all domains except Entertainment, where the G-FIPs achieve ≈67% PCC. The best G-FIPs, similarly to the S-FIPs for En-Zh (Table II), also do not use feature sets with MT quality metrics that require references.

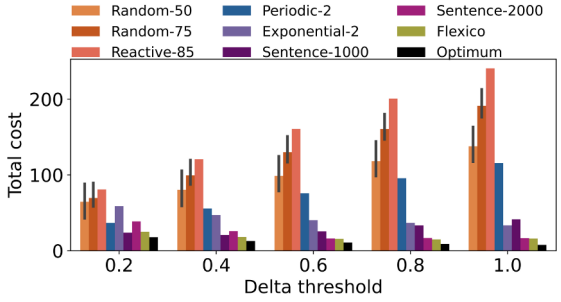
These results confirm that in the En-Zh use case the G-FIPs are very accurate when challenged with out-of-distribution queries regarding domains not seen in training.

E. MT System Utility Improvement (RQ4)

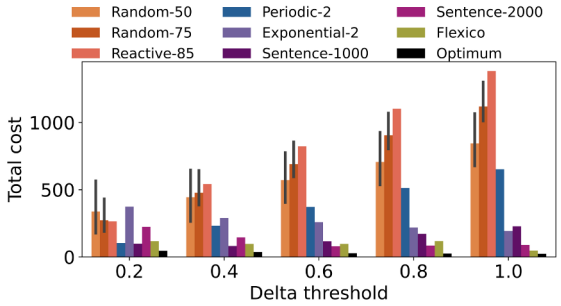
To evaluate the improvement in system utility achieved by FLEXICO, we used it to create a self-adaptive version of the *Znn* use case (see § IV). Specifically, the set **M** of MT metrics of interest is given by COMET22 [12] and CHRF [13]. We consider fine-tune costs of [1, 5, 10] and thresholds for the delta improvement (i.e. minimum improvement that we want to achieve with a fine-tune), given by Δt, from 0.1 to 1.0 with intervals of 0.1. The missed opportunity cost is set to $2\times$finetune cost$\times$FIP_pred and the incorrect fine-tune penalty to $2 \times$ finetune cost $\times$ ($\Delta t -$ FIP_pred).

TABLE III: Leave-one-out cross validation performance of the generic FIPs for En-Zh trained with RF models.

Test set	Entertainment		Environment		Finance		Governance		Health & Wellness		Sports		Travel & Tourism	
Feature Set	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC
All	0.0065	57.89	0.0038	88.98	0.0033	88.27	0.0076	92.67	0.0056	90.68	0.0051	85.85	0.0040	77.44
All + Kiwi	0.0063	56.98	0.0048	90.32	0.0038	90.50	0.0075	92.90	0.0073	89.12	0.0049	85.82	0.0039	77.20
Basic	0.0062	61.39	0.0044	88.71	0.0040	86.20	0.0070	92.55	0.0075	90.05	0.0056	86.67	**0.0031**	84.25
Basic + ContAware	0.0064	60.81	0.0041	89.70	0.0040	88.69	0.0087	91.98	0.0086	88.66	0.0051	84.69	0.0039	75.33
Basic + Kiwi	0.0054	60.13	0.0059	84.84	0.0026	84.61	0.0045	92.66	0.0082	91.31	0.0072	86.10	0.0050	83.50
Basic + MTQual	0.0069	59.64	0.0029	83.98	0.0026	80.96	0.0045	94.16	0.0062	90.65	0.0075	86.15	0.0060	83.72
ContAware	0.0049	67.02	0.0026	93.82	0.0025	94.41	0.0034	76.62	**0.0019**	87.22	**0.0015**	**86.96**	0.0043	85.53
ContAware-no-ngrams	**0.0020**	65.02	**0.0016**	**95.62**	**0.0009**	94.01	0.0038	**94.44**	0.0041	**92.25**	**0.0015**	86.39	0.0063	85.87
ContAware + Kiwi	0.0043	65.58	0.0031	94.11	0.0023	94.27	0.0039	62.46	0.0020	91.21	0.0017	85.42	0.0045	**86.76**
ContAware + MTQual	0.0035	**67.15**	0.0027	93.10	0.0023	**94.57**	0.0040	65.07	**0.0019**	91.08	0.0016	86.16	0.0038	84.96
MTQual	0.0038	13.51	0.0025	17.06	0.0026	16.42	**0.0024**	11.38	0.0028	21.25	0.0038	07.80	0.0062	09.54



(a) Scenario A.



(b) Scenario B.

Fig. 4: Total cost for each baseline as a function of the target delta improvement for MT metrics (fine-tune cost set to 1).

FLEXICO uses the generic En-Zh FIPs with all the features, and we compare it against the following baselines:

- **Periodic-2**: fine-tune the model at every two time steps;
- **Exponential-2**: fine-tune the model with an exponentially increasing period of base 2;
- **Random-n**: fine-tune at each time step with $n\%$ probability;
- **Sentence-n**: fine-tune if at least n new sentences exist;
- **Reactive-85**: fine-tune if any target MT metric is below 85;
- **Optimum**: ideal oracle that has perfect knowledge up to 5 steps into the future, and thus knows exactly the actual benefits of fine-tuning the MT model.

We consider two distinct evaluation scenarios: (i) in **scenario A** we assume that all news domains (e.g., sports, finance) have the same importance for readers throughout the week. Thus, FLEXICO decides whether to fine-tune the model before the start of each week; (ii) in **scenario B** we consider that the importance of the different types of news varies throughout the week. For example, if there are football games on Sunday, readers may be very interested in sports news on Monday. However, later on in the week, they may focus their attention on politics or finance news. In this scenario, FLEXICO has the flexibility of deciding *when* during the week to perform a model fine-tuning. This allows FLEXICO to ensure that the MT model performs better in sports news on Monday, while giving it the freedom to prioritize finance news later in the week. Since our focus is on determining *when* to fine-tune and not on estimating the future distribution of news per domain, we assume that the importance of each topic for the current week is given by the percentage of articles published in each topic in the previous week.

Figure 4 compares the utility (i.e., total cost) attained by each baseline for a fine-tune cost of 1, varying the delta thresholds, for both scenarios considered. The results for scenario A (Fig. 4a) demonstrate that FLEXICO consistently improves system utility over the naive baselines, getting closer to the optimum oracle. The results for scenario B (Fig. 4b) show that some baselines beat FLEXICO for some specific settings (e.g., sentence-2000 for larger delta thresholds). However, none of the baselines achieves robust performances across all delta thresholds (e.g., sentence-2000 for a delta threshold of 0.2 yields 91% higher total cost than FLEXICO). Overall, FLEXICO performs on average 65.4% and 49.2% less fine-tunings than the periodic baseline in scenarios A and B, respectively. This demonstrates how FLEXICO supports the creation of more sustainable SAMTS.

F. FLEXICO*'s Latency for Online Planning (RQ5)*

FLEXICO's decision-making latency plays a crucial role in determining its applicability across different scenarios, applications, and domains. Table IV displays the latency measurements for FLEXICO when deciding on the next tactic to execute, for both scenarios considered in Section VI-E. As expected, formal verification is the bottleneck, with an average execution time of 2.5 seconds for scenario B, which considers a more complex formal model. However, considering the significantly longer time required to fine-tune large models, such as MT models, we argue that a decision-making latency of approximately 3 seconds is well-suited for deciding online whether to fine-tune MT systems.

TABLE IV: Latency of the adaptation strategy extraction process (in seconds), for both experimental scenarios.

Scenario	Avg ± stdev [s] A	Avg ± stdev [s] B	95-perc [s] A	95-perc [s] B	99-perc [s] A	99-perc [s] B
Formal verification	2.10 ± 0.13	2.43 ± 0.48	2.48	2.87	2.68	3.17
MAPE-K	2.32 ± 0.14	2.66 ± 0.52	2.74	3.14	2.96	3.46

G. Threats to Validity

External validity. The results presented cannot be generalized beyond the datasets, language pairs, domains, and models used. To mitigate this threat, we used two MT models, two language pairs (one of a widely used European language pair, and one of high importance due to the number of speakers), and considered multiple domains (see appendix [68]).

Internal validity. The results of the study on the FIP features and on FIP predictive performance may be affected by (i) adding new MT-specific features to any of the feature sets described and by (ii) considering different feature combinations other than those tested. To mitigate this threat, we tested 7 feature set combinations across 4 MT metrics and with 3 different modeling approaches. Due to space constraints, some of these results are in the appendix [68].

Similarly, the conclusions regarding the improvements to system utility are intrinsically dependent on the definition of system utility and on the execution contexts evaluated (regarding fine-tune cost, MT delta threshold, MT metrics, underlying FIP model and features). For this reason, we tested 3 different values of fine-tune cost and studied the variation of the improvement for several delta thresholds.

Finally, fine-tune latency is not accounted for in the current study. As demonstrated by existing work on self-adaptive systems, accounting for tactic latency may affect the selected adaptation strategy, since tactics may be proactively enacted such that their benefits are collected when they are needed. Yet, the focus of this work is on understanding if the impact of fine-tuning MT models can be estimated, as a first step in the direction of creating self-adaptive MT systems. Having validated that these benefits can be estimated, FLEXICO paves the way to investigating, in future works, the impact of fine-tune latency on the sustainability of self-adaptive MT systems.

VII. Practical Considerations

For practitioners wanting to implement FLEXICO, the following elements must be tailored to their specific application needs: (i) system utility definition – this is intrinsically system dependent, as some systems may be more concerned with latency and throughput requirements, whereas for others user-experience may be the most important quality attribute; (ii) formal model of the system – although some components of the formal model are generic across all FLEXICO implementations (e.g., the abstraction over the MT model, the representation of the system's environment of operation, and the adaptation manager with the repertoire of tactics for adaptation, such as fine-tune), each system may have specific components that are critical to the system's overall utility and that should be accounted for when developing the formal model of the MT system to be adapted by FLEXICO.

Regarding the FIP features, we argue that the *Basic* and *Content-aware* feature sets (see § V-D) are generic across MT systems and can be reused when implementing FLEXICO for tasks in the natural language processing domain. Regarding the *MT-quality* features, although these are specific for MT tasks, FLEXICO supports replacing these features with performance features specific to the task at hand.

Collaborative FIP construction. The FID and FIP need to be built based on a target MT model. In a typical production pipeline, it is common practice to periodically fine-tune an MT model and deploy it only if it improves over the quality of the current model version. Thus, the FID could be constructed without additional costs during the initial deployment phase, before instantiating the FIPs and enabling self-adaptation. An interesting alternative approach to creating the FID, which we plan to explore in future work, is that of *Adaptation Cards*: extending the concept of Model Cards [69] to contain information regarding model adaptations (e.g. retrain, fine-tune). *Adaptation Cards* are meant to be publicly available on repositories such as GitHub [70] or HuggingFace [71] and to contain information about the impact of adapting an MT model with diverse data. For instance, an *Adaptation Card* characterizing a model fine-tune would make available (i) a pointer to the target MT model, (ii) (a subset of) the features introduced in Section V-D providing information on the statistical characteristics of the data set used for the fine-tune, (iii) the quality of the MT model before and after fine-tuning evaluated using a standardized set of MT evaluation metrics and test sets. This approach does not require users to make the data used for fine-tuning publicly available, but only the resulting features, limiting potential privacy concerns.

This will allow practitioners to reuse the data collected by others to improve their MT systems or to create an FID for FLEXICO. Promoting such data reuse is a step towards greener AI as the environmental cost of these expensive model updates/tests can be diluted across multiple usages of the data.

VIII. Conclusion

We present FLEXICO, an approach to create self-adaptive machine translation (MT) systems that automates the reasoning of when to fine-tune MT models by keeping into account both the predicted benefits of fine-tuning and the costs it incurs. This allows for avoiding ineffective fine-tunes and enhances the sustainability of MT systems' pipelines.

Our results across two datasets, two MT models and multiple domains demonstrate the predictive capability of the FIPs and the improvements to system utility enabled by FLEXICO when compared against simpler, model-free baselines, such as periodically fine-tuning the MT model. FLEXICO substantially reduces the number of fine-tunings required to obtain close-to-optimal system utility, leading to significant improvements in terms of environmental impact of model updates.

REFERENCES

[1] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns, "Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support," in *Practice and Experience in Advanced Research Computing*, 2023, pp. 173–176.

[2] J. Hutchins, "Machine translation: A concise history," *Computer aided translation: Theory and practice*, vol. 13, no. 29-70, 2007.

[3] P. Ajisoko, "The use of duolingo apps to improve english vocabulary learning," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 7, 2020.

[4] N. Jiang, T. Lutellier, and L. Tan, "Cure: Code-aware neural machine translation for automatic program repair," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021.

[5] M. Tufano, J. Pantiuchina, C. Watson, G. Bavota, and D. Poshyvanyk, "On learning meaningful code changes via neural machine translation," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019.

[6] M. Tufano, C. Watson, G. Bavota, M. D. Penta, M. White, and D. Poshyvanyk, "An empirical study on learning bug-fixing patches in the wild via neural machine translation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 4, 2019.

[7] M. Ciniselli, N. Cooper, L. Pascarella, A. Mastropaolo, E. Aghajani, D. Poshyvanyk, M. Di Penta, and G. Bavota, "An empirical study on the usage of transformer models for code completion," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, 2021.

[8] F. Stahlberg, "Neural machine translation: A review," *Journal of Artificial Intelligence Research*, vol. 69, 2020.

[9] P. He, C. Meister, and Z. Su, "Testing machine translation via referential transparency," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021.

[10] Z. Sun, J. M. Zhang, Y. Xiong, M. Harman, M. Papadakis, and L. Zhang, "Improving machine translation systems via isotopic replacement," in *Proceedings of the 44th international conference on software engineering*, 2022.

[11] D. Saunders, "Domain adaptation and multi-domain adaptation for neural machine translation: A survey," *Journal of Artificial Intelligence Research*, vol. 75, 2022.

[12] R. Rei, J. G. C. de Souza, D. Alves, C. Zerva, A. C. Farinha, T. Glushkova, A. Lavie, L. Coheur, and A. F. T. Martins, "COMET-22: Unbabel-IST 2022 submission for the metrics shared task," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Association for Computational Linguistics, 2022.

[13] M. Popović, "chrF: character n-gram F-score for automatic MT evaluation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2015.

[14] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, 2018.

[15] M. Casimiro, P. Romano, D. Garlan, and L. Rodrigues, "Towards a framework for adapting machine learning components," in *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2022.

[16] R. Calinescu *et al.*, "Synthesis and verification of self-aware computing systems," in *Self-Aware Computing Systems*. Springer, 2017.

[17] J. Cámara, W. Peng, D. Garlan, and B. Schmerl, "Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation," *Science of Computer Programming*, vol. 167, 2018.

[18] G. Moreno, J. Cámara, D. Garlan, and M. Klein, "Uncertainty reduction in self-adaptive systems," in *Procs. of SEAMS*, 2018.

[19] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic model checking: Advances and applications," *Formal System Verification: State-of the-Art and Future Trends*, 2018.

[20] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[21] R. Rei, M. Treviso, N. M. Guerreiro, C. Zerva, A. C. Farinha, C. Maroti, J. G. C. de Souza, T. Glushkova, D. Alves, L. Coheur, A. Lavie, and A. F. T. Martins, "CometKiwi: IST-unbabel 2022 submission for the quality estimation shared task," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Association for Computational Linguistics, 2022.

[22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

[23] J. Camacho-Collados and M. T. Pilehvar, "Embeddings in natural language processing," in *Proceedings of the 28th international conference on computational linguistics: tutorial abstracts*, 2020.

[24] T. Kocmi, C. Federmann, R. Grundkiewicz, M. Junczys-Dowmunt, H. Matsushita, and A. Menezes, "To ship or not to ship: An extensive evaluation of automatic metrics for machine translation," *arXiv preprint arXiv:2107.10821*, 2021.

[25] T. Kocmi, V. Zouhar, C. Federmann, and M. Post, "Navigating the metrics maze: Reconciling score magnitudes and accuracies," *arXiv preprint arXiv:2401.06760*, 2024.

[26] G. A. Lewis, I. Ozkaya, and X. Xu, "Software architecture challenges for ml systems," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021.

[27] M. Casimiro, P. Romano, D. Garlan, G. Moreno, E. Kang, and M. Klein, "Self-adaptation for machine learning based systems," in *ECSA 2021 Companion Volume, Virtual (originally: Växjö, Sweden), 13-17 September, 2021*, ser. CEUR Workshop Proceedings, vol. 2978. CEUR-WS.org, 2021.

[28] H. Muccini and K. Vaidhyanathan, "Software architecture for ml-based systems: What exists and what lies ahead," in *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE, 2021.

[29] G. A. Lewis, S. Bellomo, and I. Ozkaya, "Characterizing and detecting mismatch in machine-learning-enabled systems," in *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE, 2021.

[30] T. Bureš, "Self-adaptation 2.0," in *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2021.

[31] M. A. Langford and B. H. Cheng, ""know what you know": Predicting behavior for learning-enabled systems when facing uncertainty," in *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2021.

[32] M. Casimiro, P. Romano, D. Garlan, G. A. Moreno, E. Kang, and M. Klein, "Self-adaptive machine learning systems: Research challenges and opportunities," in *Software Architecture*. Springer International Publishing, 2022.

[33] S. Kulkarni, A. Marda, and K. Vaidhyanathan, "Towards self-adaptive machine learning-enabled systems through qos-aware model switching," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023.

[34] A. Marda, S. Kulkarni, and K. Vaidhyanathan, "Switch: An exemplar for evaluating self-adaptive ml-enabled systems," in *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2024.

[35] M. Turchi, M. Negri, M. Farajian, and M. Federico, "Continuous learning from human post-edits for neural machine translation," *The Prague Bulletin of Mathematical Linguistics*, vol. 108, no. 1, 2017.

[36] A. Van Wynsberghe, "Sustainable ai: Ai for sustainability and the sustainability of ai," *AI and Ethics*, vol. 1, no. 3, 2021.

[37] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 09, 2020.

[38] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021.

[39] R. Verdecchia, L. Cruz, J. Sallou, M. Lin, J. Wickenden, and E. Hotellier, "Data-centric green ai an exploratory empirical study," in *2022 international conference on ICT for sustainability (ICT4S)*. IEEE, 2022.

[40] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

[41] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," *arXiv preprint arXiv:2007.03051*, 2020.

[42] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *Journal of Machine Learning Research*, vol. 21, no. 248, 2020.

[43] H. Järvenpää, P. Lago, J. Bogner, G. Lewis, H. Muccini, and I. Ozkaya, "A synthesis of green architectural tactics for ml-enabled systems," in

Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society, 2024.
[44] M. Tedla, S. Kulkarni, and K. Vaidhyanathan, "Ecomls: A self-adaptation approach for architecting green ml-enabled systems," *arXiv preprint arXiv:2404.11411*, 2024.
[45] J. Townsend, "Theoretical analysis of an alphabetic confusion matrix," *Perception & Psychophysics*, vol. 9, no. 1, 1971.
[46] S.-W. Cheng, D. Garlan, and B. Schmerl, "Architecture-based self-adaptation in the presence of multiple objectives," in *ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2006.
[47] B. Schmerl, J. Cámara, J. Gennari, D. Garlan, P. Casanova, G. A. Moreno, T. J. Glazier, and J. M. Barnes, "Architecture-based self-protection: Composing and reasoning about denial-of-service mitigations," in *HotSoS 2014: 2014 Symposium and Bootcamp on the Science of Security*, 2014.
[48] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, 2003.
[49] M. Casimiro, D. Didona, P. Romano, L. Rodrigues, W. Zwaenepoel, and D. Garlan, "Lynceus: Cost-efficient tuning and provisioning of data analytic jobs," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 56–66.
[50] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 77–88. [Online]. Available: https://doi.org/10.1145/2451116.2451125
[51] T. Kocmi, E. Avramidis, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, M. Freitag, T. Gowda, R. Grundkiewicz, B. Haddow, P. Koehn, B. Marie, C. Monz, M. Morishita, K. Murray, M. Nagata, T. Nakazawa, M. Popel, M. Popović, and M. Shmatova, "Findings of the 2023 conference on machine translation (WMT23): LLMs are here but not quite there yet," in *Proceedings of the Eighth Conference on Machine Translation*. Singapore: Association for Computational Linguistics, 2023.
[52] [Online]. Available: https://www2.statmt.org/wmt24/translation-task.html
[53] M. Agarwal, S. Agrawal, A. Anastasopoulos, L. Bentivogli, O. Bojar, C. Borg, M. Carpuat, R. Cattoni, M. Cettolo, M. Chen, W. Chen, K. Choukri, A. Chronopoulou, A. Currey, T. Declerck, Q. Dong, K. Duh, Y. Estève, M. Federico, S. Gahbiche, B. Haddow, B. Hsu, P. Mon Htut, H. Inaguma, D. Javorský, J. Judge, Y. Kano, T. Ko, R. Kumar, P. Li, X. Ma, P. Mathur, E. Matusov, P. McNamee, J. P. McCrae, K. Murray, M. Nadejde, S. Nakamura, M. Negri, H. Nguyen, J. Niehues, X. Niu, A. Kr. Ojha, J. E. Ortega, P. Pal, J. Pino, L. van der Plas, P. Polák, E. Rippeth, E. Salesky, J. Shi, M. Sperber, S. Stüker, K. Sudoh, Y. Tang, B. Thompson, K. Tran, M. Turchi, A. Waibel, M. Wang, S. Watanabe, and R. Zevallos, "FINDINGS OF THE IWSLT 2023 EVALUATION CAMPAIGN," in *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*. Association for Computational Linguistics, 2023.
[54] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.
[55] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig *et al.*, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, 2011.
[56] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, 1991.
[57] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, 1980.
[58] J. Tiedemann and S. Thottingal, "OPUS-MT — Building open translation services for the World," in *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.
[59] L. T. R. G. at the University of Helsinki, "Huggingface helsinki-nlp/opus-mt-en-zh," 2020, accessed: 2024-10-31. [Online]. Available: https://huggingface.co/Helsinki-NLP/opus-mt-en-zh
[60] ——, "Huggingface helsinki-nlp/opus-mt-zh-en," 2020, accessed: 2024-12-11. [Online]. Available: https://huggingface.co/Helsinki-NLP/opus-mt-zh-en
[61] X. Ma, "Hong kong news parallel text," 2000. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2000T46
[62] OpenAI, "Chatgpt," 2022. [Online]. Available: https://openai.com/chatgpt/
[63] J. Tiedemann, "Parallel data, tools and interfaces in opus," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), 2012.
[64] ——, "The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT," in *Proceedings of the Fifth Conference on Machine Translation*. Association for Computational Linguistics, 2020.
[65] Tatoeba, "Opus-eng-fra mt model," 2021. [Online]. Available: https://github.com/Helsinki-NLP/Tatoeba-Challenge/blob/master/models/eng-fra/opus-2021-02-22.yml
[66] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.
[67] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, 2011.
[68] M. Casimiro, "Appendix — flexico: Sustainable machine translation via self-adaptation," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.14452761
[69] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *Proceedings of the conference on fairness, accountability, and transparency*, 2019.
[70] github, "Github," 2020. [Online]. Available: https://github.com/
[71] HuggingFace, "Huggingface," 2016. [Online]. Available: https://huggingface.co/