Flexico: Machine Translation Sustainability via Self-Adaptation

MARIA CASIMIRO, INESC-ID, IST, University of Lisbon & S3D, Carnegie Mellon University, Portugal
PAOLO ROMANO, INESC-ID, IST, University of Lisbon, Portugal
JOSÉ G. C. DE SOUZA, OutSystems, Portugal
AMIN M. KHAN, INESC-ID, Portugal
DAVID GARLAN, S3D, Carnegie Mellon University, USA

Machine Translation (MT) is the backbone of multiple systems and applications leveraged everyday by users. Despite 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. To meet the translation performance expectations of users, engineers periodically update (fine-tune) MT models to guarantee translation quality. However, with ever-growing machine learning models, fine-tuning operations become more expensive, raising serious sustainability concerns. Furthermore, not all fine-tunings guarantee increased translation quality, thus wasting compute resources.

To address this issue and enhance the sustainability of MT systems, we present Flexico, an approach to engineer self-adaptive MT systems, leveraging (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 9 domains demonstrates the predictive performance of the models that estimate the expected benefits of fine-tuning and their domain-generalizability. Flexico improves the sustainability of MT systems when compared to naive baselines and decreases the number of fine-tunings while preserving high translation quality.

CCS Concepts: • **Computing methodologies** → *Machine learning*; **Machine translation**; • **Software and its engineering** → **Operational analysis**.

Additional Key Words and Phrases: Model Fine-tune, Machine Translation, Self-Adaptation, Sustainability.

1 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 [30]. 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 [33], code changes [72] and bug fixing [73], or code completion [22].

Problem. In recent years, the MT field has embraced deep learning models and sequence-to-sequence architectures [63]. However, in spite of the continuous translation quality improvements observed when generating

Authors' Contact Information: Maria Casimiro, maria.casimiro@tecnico.pt, INESC-ID, IST, University of Lisbon & S3D, Carnegie Mellon University, Lisbon, Portugal; Paolo Romano, romano@inesc-id.pt, INESC-ID, IST, University of Lisbon, Lisbon, Portugal; José G. C. de Souza, OutSystems, Lisbon, Portugal, jose.camargo.souza@gmail.com; Amin M. Khan, amk@inesc-id.pt, INESC-ID, Lisbon, Portugal; David Garlan, garlan@cs.cmu.edu, S3D, Carnegie Mellon University, Pittsburgh, PA, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1556-4703/2026/9-ART
https://doi.org/10.1145/3843772

Just Accepted

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 [27, 65].

Existing solutions. 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 [61]. 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.

Limitations of existing work. 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 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.

Solution. 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 improvement to MT evaluation metrics [56, 57, 59]) outweigh the costs of the fine-tuning process (quantifiable for instance via the monetary cost required to execute the fine-tune). Specifically, consider a MT system tasked with translating news from a popular news-source, *Znn.com* [21, 62], from a *source* language into a *target* language. As time passes the popularity of different news domains (e.g., sports, politics) also varies: for example, 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. This means that the data distribution that the MT model needs to serve at inference time differs from the data distribution used to train it. Thus, when the news domain changes, the MT model might need to be fine-tuned such that higher translation quality for the emerging domain is achieved. 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 and the need for adaptation is evaluated based on this cost-benefits trade-off.

Flexico. 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 MT models, and probabilistic model checking techniques to determine when to execute the fine-tuning with the goal of optimizing quality of service.

Two main challenges hinder the adoption of existing self-adaptation frameworks [19] 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 [19] 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 lies in estimating 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.

Just Accepted

Contributions. 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.

Extended contributions. This article extends our previous work [16] by introducing the following contributions:

(1) We enriched the description of the background on machine translation and typical MT metrics, by providing more details on how they are used, on the differences between them, and how FLEXICO uses them (see Section 2).
(2) We detailed the description of the PRISM formal model, clarifying how it looks like and how it performs the necessary planning; clarified in what ways FLEXICO is modular, extensible, and abstract and expanded the description of how the MAPE-K loop is used in FLEXICO, further detailing each step (see Section 5).
(3) We clarified the purpose of evaluating MT quality resorting to fixed reference test sets and the actual meaning of FLEXICO's predictions, adding illustrative examples based on the running scenario throughout (see Section 5).
(4) We expanded FLEXICO's evaluation by considering two additional running scenarios: the first considers that fine-tune cost changes dynamically according to the actual cost of renting cloud virtual machines (VM), specifically spot instances in the Google Cloud [23, 43]; and the second one considers that VM costs vary dynamically and that news importance also varies (see Section 6.5).
(5) We conducted a new study accounting for the energy consumption and CO_2 emissions of the cloud VM used to fine-tune the MT model, to quantify FLEXICO's sustainability improvements more precisely (see Section 6.6).
(6) We extended the evaluation section by: including key-takeaways for each research questions, to highlight the key findings of our empirical study (see Section 6); extending the discussion on FLEXICO's latency (see Section 6.7); and integrating in the main body of the paper results that previously had to be presented only in an appendix (see Tables 3 and 4).

The remainder of this paper is organized as follows. Section 2 discusses related work on the domains of MT and Self-Adaptive systems and provides background on probabilistic model checking. Section 3 presents related work and discusses the limitations of existing work for determining when to adapt ML models. Section 4 introduces a running example of a MT system that is used throughout. Section 5 presents FLEXICO and Section 6 evaluates it. Section 7 discusses practical considerations and Section 8 concludes the paper. FLEXICO's source code is available at https://github.com/marialcasimiro/flexico.

2 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 [11, 13, 19, 51]. 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 [40], 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 [58], for two main reasons. First, they support non-deterministic

Just Accepted

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 (i.e., one sentence may have multiple correct translations). 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 in combination to measure precision, fluency, and overall translation quality, such as COMET22 [59], its variant that does not require reference translation sentences COMETKiwi [60], chrF (CHaRacter-level F-score) [56], and BLEU [54] henceforth referred to as its most recent implementation, SacreBLEU [57].

COMET22 [59]. COMET22 is a neural network-based, state-of-the-art metric that 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. COMET22 is trained on human-annotations that label specific error types (e.g. mistranslation, omission, addition, terminology) and captures semantic issues, reordering mistakes, and contextual information (via the embeddings), making it more aligned with human perceptions of translation quality. However, since it relies on neural networks, it has a higher computational cost than chrF and BLEU.

COMETKiwi [60]. COMETKiwi is also a learned metric (i.e., based on a neural network model) and is a Quality-Estimation (QE) variant of COMET22. This means that COMETKiwi can output estimates of translation quality having access only to the source sentence and the translation hypothesis, thus eschewing the need for expensive human reference translations. Additionally, since it belongs to the family of COMET models it can also flag meaning and language quality issues of automatic translations. However, since these models do not have access to the reference translation, they have less context and typically correlate less strongly with human scores compared to COMET22.

chrF (CHaRacter-level F-score) [56]. chrF implements a string overlap algorithm at the character-level that compares an automatic machine translation (hypothesis) against a reference translation, using character n-grams instead of word n-grams (i.e., analyzing sequences of n characters). For this reason, it is sensitive to morphological variants (e.g. inflections on the number, gender and verb tense) and small spelling differences. chrF is based on the F1-score and the intuition is that "precision" accounts for the n-grams that match (percentage of n-grams in the hypothesis which have a counterpart in the reference), and "recall" accounts for not missing content (percentage of character n-grams in the reference which are also present in the hypothesis). chrF thus operates under the assumption that a good translation is a perfect match with the reference translation. Advantages of this metric are that it is: simple and quick to compute, not requiring additional tools or knowledge/data sources; language-agnostic, tokenization-independent and shows good correlations with human judgments. However,

[1]Embeddings or embedded representations are transformations of words/sentences into lower dimensional spaces, preserving the semantic relationships between the transformed words/sentences [12].

Just Accepted

since the default value is 6-grams, chrF is less sensitive to word-order for longer character sequences and has no direct modeling of semantics or syntax.

SacreBLEU [57]. SacreBLEU is a standardized implementation of the BLEU metric, which allows researchers to reproduce MT evaluations. SacreBLEU standardizes preprocessing steps such as the tokenization, and how to handle special characters such as quotes. BLEU (Bilingual Evaluation Understudy) scores translation hypotheses by comparing them against reference translations, based on word n-gram overlaps, and computing the "precision" of the translated sentence – how many n-grams overlap with the words of the reference sentence (typically 1- to 4-grams). Thus, if a translated sentence has a high SacreBLEU score, it means it has a large word overlap with the reference translation. However, since the n-gram overlap focuses only on precision, it may not penalize missing content enough. BLEU explicitly includes the Brevity Penalty (BP) to compensate for its precision-only nature. This BP penalizes missing content (e.g. when the translation is too short compared to the reference). While the penalty may still not be enough in a linguistic sense, this mechanism addresses recall and missing content. Additionally, BLEU-derived metrics may also have a poor performance at recognizing valid paraphrases or synonyms.

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 [36]. Recent work investigated how differences in MT metric scores, both among different and the same MT metrics, are perceived by humans [37]. 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. FLEXICO uses a combination of these standard MT metrics to evaluate the quality of the MT system. Based on these evaluations, FLEXICO can then build and use dedicated predictors (Sections 5.3 and 5.5) that learn to predict how each of these MT metrics is expected to vary when the MT model is fine-tuned with a specific set of source-language and target-language sentences. For the purpose of this work we only considered these four commonly used MT metrics, however FLEXICO can be trivially extended to support additional MT metrics.

3 Related Work

Adaptive ML-enabled Systems. As defined by Lewis et al. [45], ML-enabled systems are "software systems that rely on one or more ML components". Their widespread use [17, 52] opened research questions related to how to adapt existing software architecture methodologies [45, 52] and software development processes [44] 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 [10, 18, 19, 39, 42, 48, 52] by retraining the under-performing ML component [18, 19] or replacing it by one that is expected to perform better under the current environmental conditions [39, 48]. Differently from our work, which focuses on fine-tuning MT models, existing approaches are applicable only to simpler ML tasks, namely classification problems, and consider different adaptation tactics, such as model retrain.

Domain Adaptation for MT. Many approaches have been proposed to adapt MT models for one or more specific textual genres or domains, such as modifying the underlying architecture of the MT model or training the MT model with alternative training schemes (e.g., regularized training, curriculum learning, or instance weighting) [61]. 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 [74]. 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

Just Accepted

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 [5, 8, 28, 32, 41, 64, 67, 75, 76]. Recent research efforts have focused on three main areas: (i) studying the energy consumption associated with training/fine-tuning ML models [8, 64, 76] (ii) creating tools for self-reporting and evaluating the energy consumption and carbon emissions of training ML models [5, 28, 41]; (iii) researching green architectural tactics [32] and creating systems that account for the ML model's energy consumption when selecting the ML model to use [67]. 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 most closely related work to Flexico is the framework for adapting ML-enabled systems, proposed by Casimiro et al. [19], 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 [71]. 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 (to predict the expected benefits of each adaptation tactic) and model checking (to verify a pre-specified property – e.g., maximize system utility – and create an optimal adaptation strategy), the framework proposed by [19] 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?"** (Section 5.2). 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?"** (Sections 5.3, 5.4, and 5.5).

4 Running Example

Consider a machine translation system tasked with translating news from a popular news-source, *Znn.com* [21, 62], 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 [61]. 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.

Just Accepted

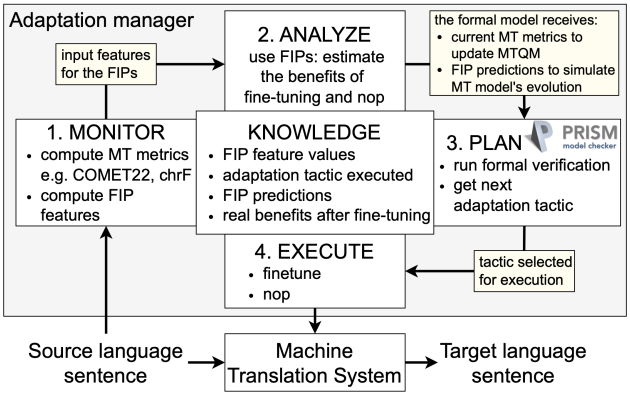


Fig. 1. Self-Adaptive Machine Translation System.

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-tune $\Delta\mathcal{Q}$ – translation quality is provided via a combination of state-of-the-art MT evaluation metrics such as COMET22 [59] and chrF [56]; and (ii) costs incurred by the news website. The costs incurred by *Znn.com* 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., tactic *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 and/or energy consumption 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 update.

Based on these costs, and on the set $\mathbf{M}$ 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:

$$\textsc{Optimize}(SysU) = \textit{choose tactic that minimizes total cost}$$

$$\textsc{Optimal Tactic} = \underset{\text{tactic}\in\{\text{finetune, nop}\}}{\arg\min} \ \textsc{cost}(\text{tactic})$$

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

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

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.

Just Accepted

5 Flexico

This section introduces Flexico, a generic approach for creating Self-Adaptive Machine Translation systems (SAMTS). The goal of an SAMTS is to optimize its system utility, which is inherently dependent on both the operational context and system characteristics. To account for this dependency, one of the key requirements for the SAMTS is that it should be ***modular***, allowing the system to be easily adapted to different scenarios and utility definitions. Flexico achieves modularity by separating two core tasks: estimating the expected benefits of adaptation and analyzing the cost–benefit trade-offs of alternative adaptation tactics. The estimation of adaptation benefits is delegated to Fine-tune Impact Predictors (FIPs), described in Sections 5.3 and 5.5. The second task – evaluating the trade-offs between adaptation costs and expected benefits – is performed using probabilistic model checking. Flexico also incorporates modularity within its formal model through a clear separation of concerns. In particular, adaptation tactics and the MT model are encapsulated in independent modules. Section 5.2 further elaborates on this.

Additionally, and although the *Znn.com* use case considers a single tactic for adapting 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). Furthermore, each user or system may have: (i) distinct preferences or requirements regarding the evaluation metrics used to assess translation quality and (ii) different optimization goals that require modifying how system utility is defined. For these reasons, the SAMTS is designed to be ***extensible***. Flexico's modular architecture supports this requirement by facilitating the update (addition/removal) of adaptation tactics and MT evaluation metrics within the corresponding modules of the formal model. Extensibility is further supported by the design of the FIPs that are built specifically for each tactic. As a result, adding new tactics to the SAMTS repertoire only requires the development and integration of the corresponding FIPs. Finally, system utility can also be easily modified to cater to different optimization goals, without requiring changes to other system modules.

Lastly, to balance the expected benefits of fine-tuning the MT model against its associated costs, the SAMTS must estimate the expected benefits of fine-tuning and their impact on overall system utility. In Flexico, this trade-off analysis is performed by a probabilistic model checker. To enable such reasoning, an ***abstract*** representation of the MT model is required, as directly encoding complex MT models in the formal model would be infeasible. The *Machine Translation Quality Matrix* (MTQM) (Section 5.2) provides this abstraction. Specifically, the MTQM enables Flexico to reason about the impact of different adaptation tactics on the predictive performance of MT models without requiring a faithful representation of the underlying model architecture within the formal model.

In the following sections we (i) provide an overview of the components required by SAMTS (Section 5.1); (ii) explain how Flexico formally models MT models (Section 5.2), addressing challenge 1 (cf. last paragraph of Section 3); and (iii) how the benefits of fine-tuning are estimated by leveraging fixed test-sets (Section 5.3) and MT specific features (Section 5.4) via FIPs (Section 5.5), addressing challenge 2 (cf. last paragraph of Section 3).

5.1 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 [34] 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 translation quality of the MT model currently in use via MT-specific metrics (e.g., COMET22 [59], chrF [56], SacreBLEU [57]) and features for estimating the expected benefits of fine-tuning; 2) **A**nalyzes the expected costs and benefits 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

Just Accepted

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). Below, we describe each MAPE-K stage in more detail.

Monitor stage: feature monitor. This stage is responsible for collecting all the information that the analyze stage needs to estimate the expected benefits of fine-tuning the MT model. Specifically, this information can be obtained by keeping track of the sentences that the MT model receives for translations, the corresponding translations, and the actual correct translations, also known as reference translations. These are not always available and can be obtained by resorting to translation experts. With these sentence triplets, monitor can compute MT specific metrics (cf. Section 2) that characterize the current translation quality of the system. When reference translations are not available, monitor can only compute MT specific metrics, such as COMETKiwi [60], that do not rely on these golden translations. Then, based on the input sentences and corresponding translations, monitor further computes features that are used by the FIPs in the analyze stage to estimate the expected benefits of adaptation. These features, discussed in detail in Section 5.4, are designed to capture environment shifts by evaluating the sentence pairs at the semantic and lexical levels.

If new adaptation tactics are added to FLEXICO's repertoire, it is necessary to revise the data collected by this stage to ensure that the data required to analyze the expected benefits of the new tactics is monitored.

Analyze stage: Fine-tune Impact Predictors. This stage consumes the features computed by the monitor stage to query the FIPs and estimate the expected benefits of fine-tuning. Section 5.3 explains how fixed test sets are used to estimate the benefits of fine-tuning and Section 5.5 presents a detailed description of the FIPs.

Additionally, the analyze stage should also estimate the costs of the fine-tune so that plan can understand whether the benefits outweigh the costs and if adaptation is worth it. Thus, to estimate the expected costs of the fine-tuning process, the analyze component can, for instance, either assume fixed fine-tune costs (Section 6.5, scenarios A and B) or consider the costs of renting virtual machines in the cloud and estimate the expected duration of the fine-tune to compute its corresponding expected cost (Section 6.5, scenario C). The costs and benefits predictions are then used to plan the adaptation strategy.

When new adaptation tactics are added to FLEXICO's repertoire, the analyze stage must be updated with the FIPs that will estimate the expected benefits and costs of executing each new adaptation tactic. All these predictions will then be used by the plan stage.

Plan stage: probabilistic model checker. This stage combines a verification property, a formal model of the MT system, predictions from the analyze stage, and features computed by the monitor stage (such as the current translation quality of the MT system) to select the expected optimal adaptation tactic. Specifically, the verification property specifies the task of the model checker when formally verifying the formal model of the MT system. For FLEXICO, the verification property is expressed using Probabilistic Reward Computation Tree Logic [4] (a formal property specification language used in model checking) as $R^{\text{costs}}_{\text{min}=?}[F\,\text{end}]$ and encodes the goal defined in Section 4: select an adaptation tactic that optimizes system utility, i.e., minimizes the costs incurred by the system.

The formal model of the system encodes the definition of system utility (Listing 1), contains the repertoire of adaptation tactics that can be employed to adapt the system (Listing 2) and models how the execution of each tactic impacts the MT model (Listing 3). Thus, whenever FLEXICO's repertoire is extended, the adaptation manager and the MT system modules of the formal model must be updated to account for the new adaptation tactics. Additionally, the new tactic also needs a dedicated module in the formal model, akin to the module for fine-tune

Just Accepted

Listing 1. Definition of system utility in the formal model.

```
// system utility definition: whenever the simulation clock advances (there is a `tick`)
// the system is charged a cost that depends on the tactic that was executed
formula cost_incurred = (currTactic = finetune ? finetuneCost : nopCost);

rewards "costs"
    [tick] true & (time>0) : cost_incurred;
endrewards
```

Listing 2. Repertoire of adaptation tactics in the adaptation manager module.

```
module adaptation_manager
    selectTactic : bool init false;     // whether it is the right moment to select a tactic
    currTactic : [noTactic .. finetune] init nop;  // current tactic selected to be executed

    // when there is a new batch of data, the system's state evolves to enable tactic selection and resets the currTactic variable
    [newBatch] !selectTactic -> (selectTactic'=true)&(currTactic'=noTactic);

    // repertoire of adaptation tactics
    [nop_start]      (selectTactic=true)             -> (selectTactic'=false)&(currTactic'=nop);
    [finetune_start] (selectTactic=true)&(newData>0) -> (selectTactic'=false)&(currTactic'=finetune);
endmodule
```

Listing 3. Update to metric COMET22 upon the execution of a finetune tactic.

```
module mts
    // INIT_COMET22: COMET22 score of the MT system at the start of the formal verification process.
    // When set to -1, metric COMET22 is not in use
    comet22_perf : [0 .. 100] init INIT_COMET22;

    // impact of a fine-tune on the MT system. After fine-tuning completes, the pool of new data for future fine-tuning is reset
    [finetune_complete] true ->
        1:(finetune_comet22_delta'=round(finetuneDelta_comet22)&(sysState'=updateMetric)&(newData'=0);

    [] (sysState=updateMetric) -> 1:(comet22_perf'=new_comet22_perf)&(sysState'=sysWait);
endmodule

// average variation of the COMET22 MT metric when a fine-tune is executed
formula finetuneDelta_comet22 = ((1/numNewsTopics) * ((weight_Finance * fipDeltaFinance_comet22)
    + (weight_Entertainment * fipDeltaEntertainment_comet22) + (weight_TravelTourism * fipDeltaTravelTourism_comet22)
    + (weight_HealthWellness * fipDeltaHealthWellness_comet22) + (weight_Sports * fipDeltaSports_comet22)
    + (weight_Environment * fipDeltaEnvironment_comet22) + (weight_Governance * fipDeltaGovernance_comet22)
    )
));

// update to the COMET22 MT metric in case of fine-tune
formula new_comet22_perf = (INIT_COMET22 = -1 ? 0 : (
    (comet22_perf + finetuneDelta_comet22) >= 100 ? 100 : comet22_perf + finetuneDelta_comet22
));
```

(Listing 4), to specify the constraints for its execution and to prevent simultaneous executions of multiple tactics. Section 5.2 provides additional details on how to formally model MT systems to be used by FLEXICO.

Just Accepted

Listing 4. Formal module of the fine-tune tactic.

```
module finetune
    finetune_state : [0 .. 1] init 0;  // whether the fine-tune is ongoing
    finetune_go : bool init false; // whether the fine-tune process can be started

    // finetune tactic NOT applicable: no new data - DON'T start
    [finetune_not_applicable] (selectTactic)&(finetune_go=true)&(newData=0) -> (finetune_go'=false);

    // another tactic has been selected so don't execute this one
    [finetune_no_start] (currTactic!=noTactic)&(finetune_go=true)&(finetune_state=0) -> (finetune_go'=
     false);

    // finetune tactic applicable - start
    [finetune_start] (selectTactic)&(finetune_go=true)&(newData>0) -> (finetune_state'=1);

    // finetune tactic COMPLETE
    [finetune_complete] (finetune_go=true)&(finetune_state=1) -> (finetune_go'=false)&(finetune_state'=0);

    [tick] !finetune_go -> 1:(finetune_go'=true);
endmodule
```

Execute stage: actuators. This stage is responsible for triggering the execution of the adaptation tactic selected by the plan component and for guaranteeing and overseeing its execution. For FLEXICO, this corresponds to providing the means to execute the fine-tune with the sentences received for translation and corresponding reference translations that haven't yet been employed to fine-tune the model. To obey FLEXICO's extensibility requirement, whenever new adaptation tactics are added to FLEXICO's repertoire, this stage needs to be updated to provide actuators for the new tactics. For the current fine-tune tactic, the actuator is the API for the fine-tune function.

Knowledge. The knowledge component stores the outcome of each adaptation process, thus keeping track of: the features extracted during the monitoring stage, the FIP predictions computed in the analyze stage, and the adaptation tactic selected by the plan stage (i.e., fine-tune or not). When the plan stage determines that the expected optimal adaptation tactic is to fine-tune the MT model, the outcome of the execution of the adaptation tactic is also stored in the knowledge component. This corresponds to the quality of the fine-tuned MT model when evaluated on the fixed test sets with the selected MT metrics. All the information stored in the knowledge component can be used to improve the FIPs by also fine-tuning/re-training them with this new data. However, since our goal with FLEXICO was to improve MT systems, we did not explore this avenue and defer it for future work.

5.2 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 2 having such a model is required by probabilistic model checkers. However, 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, each row represents a specific MT evaluation metric (e.g., COMET22 [59], chrF [56], SacreBLEU [57]), and thus each cell contains the quality of translations for a specific domain, quantified via a specific MT metric. Figure 2

Just Accepted

			...	
COMET22	75	80	...	78
CHRF	73	82	...	79
⋮	⋮	⋮	⋱	⋮
SacreBleu	77	81	...	81

Fig. 2. Machine Translation Quality Matrix (MTQM): columns represent relevant domains to the MT system such as sports ( ), finance ( ), and tourism ( ); rows represent different MT evaluation metrics (e.g., COMET22 [59], chrF [56], SacreBLEU [57], which vary in the range $[0, 100]$).

Listing 5. Representation of the MTQM row for metric COMET22 using the syntax of the PRISM model checker.

```
// expected/predicted (variation in) target metric (e.g. comet22) score due to a model finetune for each news domain
const double fipDeltaFinance_comet22;
const double fipDeltaEntertainment_comet22;
const double fipDeltaTravelAndtourism_comet22;
const double fipDeltaHealthAndwellness_comet22;
const double fipDeltaSports_comet22;
const double fipDeltaEnvironment_comet22;
const double fipDeltaGovernance_comet22;
```

shows an example of this matrix, with domains like sports, finance, and tourism; and Listing 5 demonstrates how to represent a row of the MTQM. Specifically, each MTQM cell is defined as a constant and left under-specified such that whenever the formal verification process is triggered, the formal model can receive updated values for the MTQM.

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 Flexico to reason 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.com* (see Section 4).

To create an MTQM, a system such as *Znn.com*, which stores news articles and corresponding translations collected over time, would start by fetching all (or, for example, a random subset) of the news articles and translations for a specific period of time, for instance the past week of data. *Znn.com* would then compute each MT metric of interest (e.g., COMET22, chrF, and SacreBLEU as in Figure 2) on the selected data. These metrics have standard open-source implementations that can be leveraged by users without requiring them to re-implement each MT metric's underlying logic. So, internally, *Znn.com* would only have to pass the sentences of the original news articles (source sentences), the corresponding machine translations (hypothesis translation sentences) and the correct translations (reference sentences) to the library/package that implements each MT metric and save the metric's value. The MTQM would then encode how well the MT system performed during the selected period: the higher the values of each MT metric, the better the translation quality. A perfect MT system would thus be one for which the MTQM has value 100 on all cells.

5.3 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 [19] by introducing several novel methodological aspects to be effectively employed in the context of MT systems. These include:

Just Accepted

(1) Identifying a set of **MT specific features** (see Section 5.4.) to capture dataset 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).
(2) Introducing a new methodology to estimate the variations of the predictive quality of the MT model: differently from existing work [19], 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 (e.g., politics, sports) 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 [1, 35] and IWSLT [2] shared tasks). This approach allows researchers to reproduce results and compare the quality of different MT models (e.g., trained with different data) over time. Additionally, since fixed test-sets are domain specific, they can be used to understand how data (and the environment) is changing over time. For instance, if a newly fine-tuned model has higher translation quality in the sports domain with respect to its prior version, this can suggest the occurrence of a data set shift that led to the generation of new valuable training data in this domain.
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) Proposing two alternative FIP designs, by leveraging the notion of fixed reference test sets: 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 (Section 5.5).

Thus, the benefits of a fine-tune can be computed by leveraging both the MTQM and the fixed reference test-sets as follows. Suppose *Znn.com*'s MT model is fine-tuned periodically [2] (e.g., each week). Then, at the end of the week one can generate a new MTQM, evaluating *Znn.com*'s translation quality using the fixed test-sets and the MT metrics of interest. By comparing the MTQM's of two consecutive weeks, one can see how a single fine-tune influenced translation quality across domains and MT metrics. For example, if for week w *Znn.com* had a COMET22 score of 75 for sports news (as shown in Figure 2) and in week $w + 1$ the fine-tuned model's COMET22 score is 77 for sports news, this means there is an improvement of 2 COMET22 points as a consequence of fine-tuning. Since some weeks may have more news from a specific domain (e.g., sports), the benefits of a fine-tune are not always positive for all domains (i.e., translation quality may degrade – e.g., finance news may see a decrease of 3 COMET22 points, moving from 80 to 77). FLEXICO's FIPs are built precisely to predict these variations in translation quality: FLEXICO uses the new data available to fine-tune the MT model to compute

[2]Note that if the MT model is never finetuned, its MTQM is always the same, because neither the MT model nor the test-sets will have changed.

Just Accepted

MT-specific features (Section 5.4); then these features are used to query the FIPs; finally, the FIPs output the expected increase/decrease in translation quality for a specific MT metric and domain.

5.4 MT-specific Features for FIPs

The AIPs [19] 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 feature sets for MT models: *content-aware* and *MT-quality* features.

Content-aware features. These features are inspired by work on statistical MT [38] 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 [49], embeddings [12] 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 [46] distance between 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 [12]. Comparing embedding distributions in the new and old data sets (e.g., through cosine and Euclidean [24] 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 [59], chrF [56], SacreBLEU [57], COMET22Kiwi [60]. 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.

5.5 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.

Just Accepted

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 5.4, 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 and generic (domain-agnostic) FIPs.

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.com*, this approach requires having $\mathbf{M} \times n$ FIPs . As defined in Section 4, $\mathbf{M}$ refers to the set of MT metrics selected for evaluating translation quality and n refers to the distinct news domains under consideration. 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.

Domain-agnostic (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.

6 Evaluation

This section investigates the following research questions:

RQ1 How accurately can the Fine-tune Impact Predictors (FIPs) predict the benefits of fine-tuning? (Section 6.2)
RQ2 What features lead to higher FIP quality and how expensive is it to train and query the FIPs? (Section 6.3)
RQ3 Can the FIPs generalize across domains? (Section 6.4)
RQ4 What are FLEXICO's utility gains? (Section 6.5)
RQ5 How does FLEXICO contribute to sustainability and to the reduction of CO_2 emissions? (Section 6.6)
RQ6 Is FLEXICO tractable for online adaptation? (Section 6.7)

6.1 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.com*, uses the OpusMT model [6, 7, 70] available through HuggingFace and LDC's 'Hong Kong News Parallel Text' [47], which contains news data from 1997 to 2000. To create the fixed test sets, we used chatGPT [53] to assign domains, which are not present in the original dataset,

Just Accepted

Table 1. COMET22 score of the G-FIPs and average COMET22 score of the S-FIPs (considering 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**

Just Accepted

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.

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 [47] (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. The FID was created by fine-tuning the OpusMT model 147 times, yielding 10585 FID samples.

English-French (En-Fr). For this use case, we employ the Opus datasets [68] and the Tatoeba Opus MT model [66, 69]. 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.

6.2 FIP Prediction Quality (RQ1)

RQ1: FIP Prediction Quality (see Table 1)

The Specific-FIPs predict the expected COMET22 variation on average with up to 89% Pearson Correlation Coefficient (PCC) and outperform the Generic-FIPs. The Generic-FIPs achieve lower, but still comparable, results in the En-Zh use case (76% PCC), but significantly worse in the En-Fr use case (33% PCC), due to its more challenging nature.

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 [20], Random Forest (RF) and Linear (Lin) regressors from the SKLearn Library [55]; (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 1, 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 5.4 (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 consider 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 best performing modeling approaches (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 Section 6.1). 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.

6.3 FIP Feature Importance and Cost (RQ2)

RQ2: FIP Feature Importance and Cost (see Tables 2, 3, 4) and Figure 3

- **Feature importance:** for scenarios with more homogeneous domains, consider an FIP feature set with content-aware features. For use-cases with more heterogeneous domains, MT-quality features can help FIPs improve their prediction quality.
- **Feature computation cost:** computing FIP features is up to 80× cheaper than fine-tuning for small MT models – these savings increase further when considering larger MT models, for which fine-tuning takes longer; FIP feature computation cost is independent from MT model size.
- **FID construction cost:** FLEXICO's FID generation methodology yields $O(n^2)$ samples based on n fine-tunings, and our study shows that n can be reduced up to 5× while still leading to FIPs with high MAE and PCC.

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

Just Accepted

Table 2. 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

Table 3. Performance of the specific FIPs when predicting COMET22 with RF models for the En-Fr use case. Each test set (i.e., Elitr, Elrc, EuBookshop, PhP, Tanzil, Tedx-fr, Tedx-fr_ca) is a dataset from Opus [68]. Consult the appendix [14] for more information about each test set.

Test set	Elitr		Elrc		EuBookshop		Php		Tanzil		Tedx-fr		Tedx-fr_ca	
Feature Set	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC	MAE	PCC
All	0.0014	57.52	**0.0016**	**57.42**	**0.0008**	61.24	**0.0010**	61.50	0.0090	58.44	0.0006	07.07	0.0008	71.42
All + Kiwi	0.0014	57.05	0.0017	07.92	0.0011	-00.32	0.0018	06.34	0.0063	42.98	0.0007	-00.36	0.0010	31.76
Basic	0.0013	66.62	0.0020	10.43	0.0010	18.29	0.0012	01.73	0.0020	-10.47	0.0009	03.10	0.0009	17.41
Basic + ContAware	**0.0012**	**67.11**	0.0018	-10.19	0.0011	03.14	0.0015	-00.97	0.0032	-13.97	0.0008	-02.68	0.0009	02.68
Basic + Kiwi	0.0014	53.83	0.0018	05.17	0.0011	14.72	0.0012	04.86	**0.0017**	46.11	0.0009	08.40	0.0008	45.31
Basic + MTQual	0.0014	58.16	**0.0016**	53.19	0.0010	**65.65**	0.0012	64.41	0.0027	59.69	**0.0004**	46.92	**0.0006**	**77.01**
ContAware	0.0027	14.90	0.0024	-22.55	0.0016	-17.10	**0.0010**	-03.38	0.0104	18.88	0.0015	21.52	0.0011	-08.58
ContAware-no-ngrams	0.0028	10.77	0.0025	-14.07	0.0019	03.03	**0.0010**	-05.25	0.0105	18.77	0.0015	17.00	0.0011	15.86
ContAware + Kiwi	0.0022	-08.58	0.0017	10.33	0.0014	-12.93	0.0014	05.00	0.0065	66.00	0.0008	23.50	0.0010	32.24
ContAware + MTQual	0.0024	05.90	**0.0016**	53.88	**0.0008**	60.09	0.0011	**65.02**	0.0091	**72.98**	**0.0004**	**54.23**	0.0008	70.55
MTQual	0.0023	07.95	0.0022	37.52	0.0010	57.52	0.0015	52.53	0.0033	20.99	**0.0004**	29.15	0.0009	64.64

(i) the 3 feature sets presented in Section 5.4 (i.e., Basic, Content-aware, MT-quality — note that the MT-quality set includes all four MT metrics: COMET22 [59], chrF [56], SacreBLEU [57], COMET22Kiwi [60]);

(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 (see Table 2), 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 Table 3) 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.

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

Just Accepted

Table 4. Cost (in USD) of feature set computation for a single FID sample — i.e., whenever the FIPs are queried, we pay these costs.

	Feature set								Fine-tune
	All	Basic	ContAware	MTQual	Basic + MTQual	Sent-overlaps	Embeddings	n-grams	
Cost-En-Zh	7.49e-03	6.46e-05	4.82e-03	2.61e-03	2.67e-03	1.28e-07	1.09e-03	3.73e-03	8.38e-02
Cost-En-Fr	5.06e-02	1.82e-04	3.69e-02	1.35e-02	1.37e-02	2.36e-07	3.98e-03	3.30e-02	4.80e-01

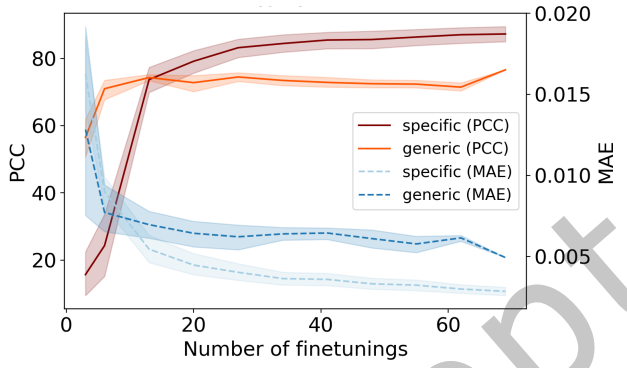


Fig. 3. MAE and PCC for G-FIPs (RF, En-Zh, COMET22), varying the number of finetunings to create the FID.

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 Section 5.4). 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 (Table 4) 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 where these measurements were collected, 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 (Section 5.5), our FID was obtained by fine-tuning the MT models 147 and 105 times. Out of these, 69 and 57 fine-tunings were used for training the FIPs for the En-Zh and En-Fr use-cases, respectively, and the remaining ones for testing the FIPs. 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

Just Accepted

Table 5. 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

Just Accepted

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.

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.

6.4 FIP Domain Generalization (RQ3)

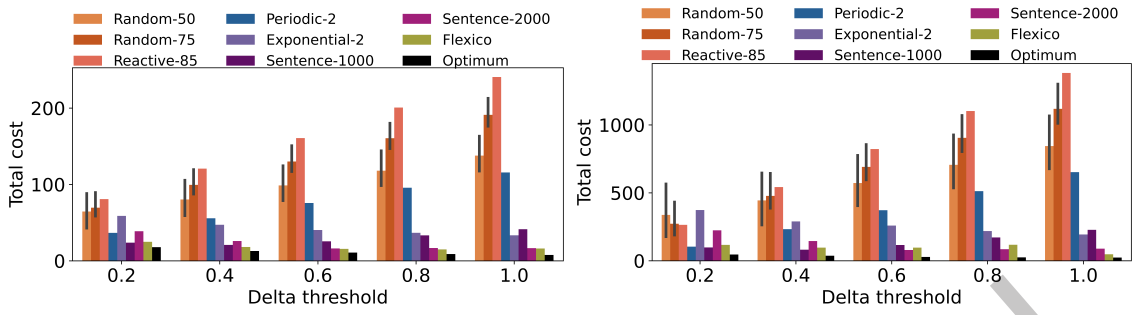
RQ3: FIP Domain Generalization (see Table 5)

The Generic-FIPs estimate the expected benefits of fine-tuning with PCC up to 95% (average of 88%) on domains unseen in the training set (En-Zh use case), which is on-par with the average performance of the Specific-FIPs (PCCs of up to 86% for random forest models).

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 5 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 6.2 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 [14].

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 2), 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.



(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).

6.5 MT System Utility Improvement (RQ4)

RQ4: System Utility Improvement (see Figures 4, 7, and 8)

FLEXICO performs on average 65.4% (scenarios A, C) and 49.2% (scenarios B, D) less fine-tunings than the periodic baseline, which shows how it can avoid superfluous fine-tunes. Additionally, FLEXICO is the baseline that gets closer to the Optimum oracle, beating the best baseline, Sentence-2000, by 24.28% (233.39 vs 308.24), 23.38% (1296.01 vs 1691.49), 21.98% (1.07 vs 1.37), and 17.16% (6.36 vs 7.67) on scenarios A, B, C, and D respectively.

To evaluate the improvement in system utility achieved by FLEXICO, we used it to create a self-adaptive version of the *Znn.com* use case (see Section 4). Specifically, the set **M** of MT metrics of interest is given by COMET22 [59] and CHRF [56]. We consider 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 $2 \times \texttt{finetune cost} \times \Delta_{\texttt{real}}$ and the incorrect fine-tune penalty to $2 \times \texttt{finetune cost} \times (\Delta t - \Delta_{\texttt{real}})$, where Δ_{real} denotes the real benefits accrued when the MT model is fine-tuned.

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 four distinct evaluation scenarios. 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. 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

Just Accepted

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.

In practice, to make Flexico work for each scenario, the main modifications that were performed were at the level of the formal model. For scenario B, the formal model requires two main changes: it needs to reason about the whole week, and thus simulates how the quality of the MT system evolves throughout the week; and it needs to be informed about how the importance of each news topic varies throughout the week such that it can reason about how fine-tuning on each week-day will impact the overall translation quality of the MT system at the end of the week.

For these two scenarios, we consider that fine-tune cost is fixed and can be set to one of [1, 5, 10]. This is an initial simplifying assumption which allows us to focus our attention in these two scenarios on the problem of estimating the benefits of a fine-tune (rather than its cost).

Then, in **scenario C**, we again assume that all news domains are equally important, but relax the assumption on the fixed fine-tune cost and consider instead that each finetune is performed by renting cloud spot instances [23]. This means that the fine-tune cost varies over time, accompanying the market fluctuations. Additionally, we allow Flexico to select the availability zone (i.e., the region where the cloud virtual machine (VM) is physically located) in order to minimize the product of monetary cost and non-carbon-free emissions (see Figure 5). For this scenario, we get historical data regarding spot instance cost from SpotLake [43] and data about Google's carbon free energy and grid CO_2 emissions from the Internet Archive's Wayback Machine [31]. For both data sources, we consider data from the past year (i.e., from 2024-07-01 to 2025-06-30). Since cloud VM costs are typically specified per hour, if we can estimate how long a fine-tune is expected to take, we can straightforwardly compute its expected cost. We show that, akin to existing work [15, 25], by relying on historical data one can estimate how long a fine-tune will take and thus also its expected cost (see Figure 6).

Finally, in **scenario D**, we consider that: (i) the importance of the different types of news varies throughout the week (same as scenario B); and (ii) that each finetune is performed by renting cloud spot instances, hence having variable fine-tune costs over time (as in scenario C). In this scenario, the probabilistic model checker determines whether and when to fine-tune (i.e., the best week-day to deploy the fine-tune) as well as the optimal availability zone in which to deploy the fine-tune.

Scenarios A and B. Let us start by analyzing the results for scenarios A and B. 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 A and B. The results for scenario A (Figure 4a) demonstrate that Flexico consistently improves system utility over the naive baselines, getting closer to the optimum oracle. The results for scenario B (Figure 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, for the delta thresholds displayed in Figure 4, Flexico performs on average 9 and 13.2 adaptations while the periodic baseline performs 26, corresponding to 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. The results obtained with higher fine-tune costs are only presented in the appendix [14] due to space constraints, however they show the same trends as the results displayed in Figure 4. Specifically, for scenario A Flexico consistently beats the baselines, getting closer to the optimum oracle. For scenario B, Flexico is the most robust baseline, ensuring consistent gains regardless of how the delta thresholds is set.

Scenario C. Let us start by analyzing the dynamic nature of the spot instances' cost and carbon emissions. Figure 5 shows how the average hourly price (in US dollars) of renting a google n1-standard-2 VM multiplied by

Just Accepted

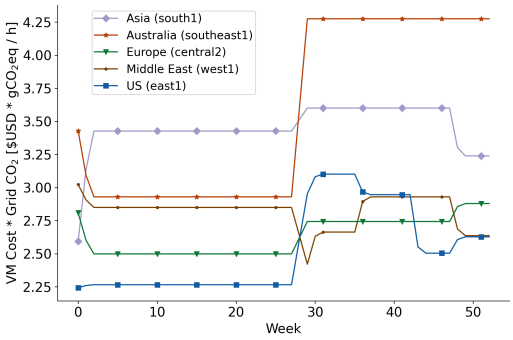


Fig. 5. The plot shows the evolution over time of a cost function defined as the product between the average hourly cost of renting a VM equipped with a V100 GPU in the Google Cloud and the per hour carbon free emissions in different geographical regions.

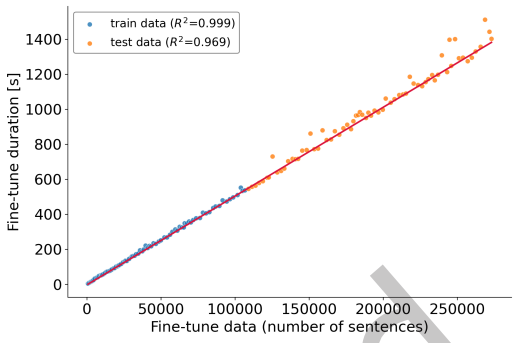


Fig. 6. Estimation of the finetune duration as a function of the amount of data available for a fine-tune (for the En-Zh use-case).

its hourly carbon-free emissions [4] vary over time across five different regions — a metric that aims to combine the economical cost of renting cloud resources and the corresponding environmental impact. We consider the use of a VM that hosts a Nvidia Tesla V100 GPU, i.e., the GPU model used to fine-tune our MT models. This plot shows that there is no availability zone that is optimal over the entirety of the considered time period. This data motivates the necessity to continuously monitor how both the spot instance cost and carbon free energy evolve over time to guarantee the best trade-offs between monetary cost and CO_2 emissions.

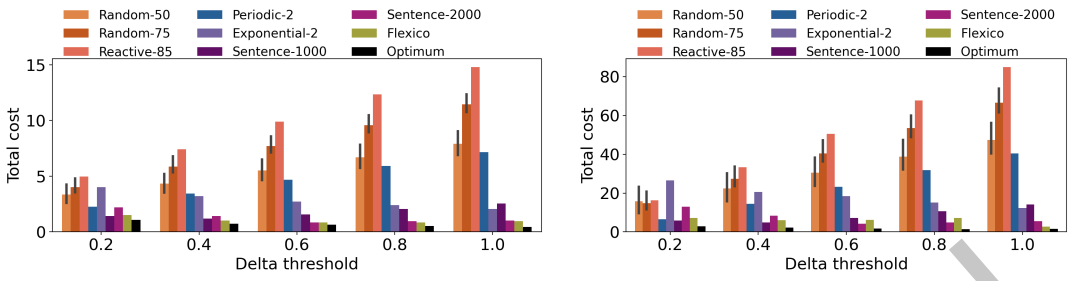
Since the spot instance cost and grid carbon are specified per hour, we are required to estimate the duration of a fine-tune, in order to be able to precisely compute the actual cost incurred by each baseline. Figure 6 shows how the amount of data for a fine-tune is an excellent proxy to estimate the fine-tune duration (considering a target VM type and MT model). The plot shows that training a linear regression using the FID's training data, we obtain an r^2 of 0.999 and when the same regression is applied to the test data, the r^2 decreases only slightly to 0.969.

Finally, regarding system utility, Figure 7a shows that FLEXICO keeps on improving system utility over the baselines even when considering dynamic cost environments. In this scenario, FLEXICO performs on average 9 adaptations while the periodic baseline performs 26, again corresponding to 65.4% less fine-tunings. Additionally, we can once more notice the sensitivity of Sentence-*n* to the correct tuning of its parameter (*n*) and to the delta threshold. Namely, the Sentence-1000 and Sentence-2000 baselines degrade and improve (respectively) as the delta threshold increases. Conversely, FLEXICO performs closer to the Optimum oracle across delta thresholds, even if it is marginally beaten in some cases (e.g., for delta threshold of 0.2, Sentence-1000 has a slightly lower total cost than FLEXICO – 1.41 vs 1.51).

Scenario D. This scenario combines the dynamic aspects of Scenarios B and C by accounting for varying news topic importance throughout the week and performing fine-tuning online, thereby exposing the process to fluctuations in cloud market prices. The results (Figure 7b) show that FLEXICO is the most consistent baseline

[4]GCFE stands for Google carbon free emissions and denotes the percentage of energy consumed by Google VMs that is carbon free. Thus, to compute the non-carbon free emissions, we compute 1-GCFE [%] and multiply by the CO_2 that the grid emits [gCO_2eq/kWh] and by the GPU's TDP (Thermal Design Power) – represents the maximum amount of heat a graphics card is designed to dissipate under typical operating conditions, typically measured in Watts. Since we did not have exact results for this parameter, we adopted a conservative approach and considered the GPU's maximum TDP of 0.3 kW.

Just Accepted



(a) Scenario C. (b) Scenario D.

Fig. 7. Total cost (system utility) for each baseline as a function of the target delta improvement.

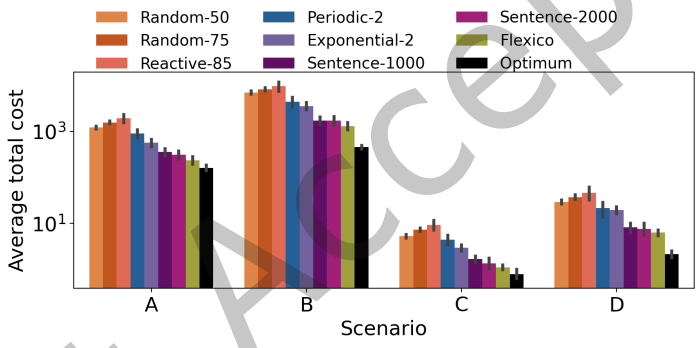
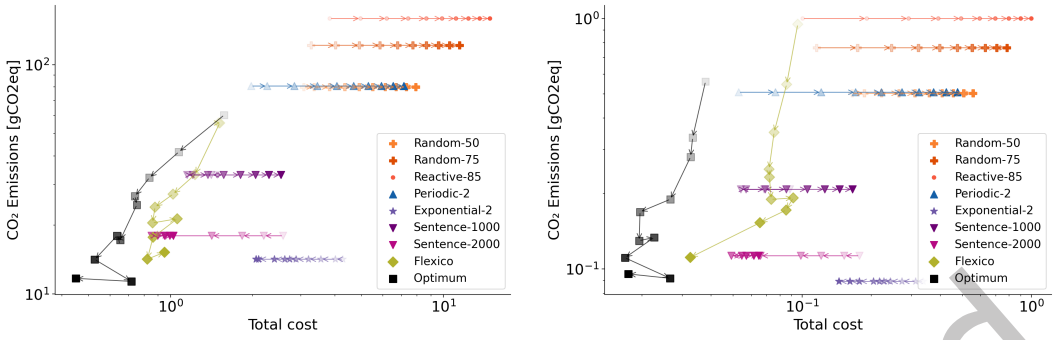


Fig. 8. Average total cost (and standard deviation) across delta thresholds for each baseline and for each scenario.

Just Accepted

across delta thresholds, while the Sentence-1000 baseline is better for lower delta threshold and the Sentence-2000 baseline improves over Flexico for larger delta thresholds. This is the same trend observed in scenario B (cf. Figure 4b).

Overall, comparing the results obtained for the four scenarios (see Figure 8) we can clearly see that Flexico is the baseline that gets closer to the Optimum oracle regardless of the scenario considered. On average, Flexico beats the best baseline, Sentence-2000, by 24.28% (233.39 VS 308.24), 23.38% (1296.01 VS 1691.49), 19.27% (1.11 VS 1.37), and 17.16% (6.36 VS 7.67) on scenarios A, B, C, and D respectively. This demonstrates how Flexico can predict the expected benefits of a fine-tune, analyze their impact on overall system utility, and determine whether to adapt based on which tactic benefits the system the most.



(a) Scenario C.

(b) Scenario D.

Fig. 9. CO_2 emissions and total cost incurred by each baseline. Marker transparency indicates delta threshold (more transparent → lower delta threshold). The arrows denote moving from the lowest to the highest delta threshold.

6.6 Energy Consumption and CO_2 Emissions (RQ5)

RQ5: CO_2 Emissions (see Figure 9)

FLEXICO can directly factor in the environmental impact of a fine-tune when deciding whether to adapt, thus not performing superfluous fine-tunes. Additionally, FLEXICO emits up to almost an order of magnitude less CO_2 than the periodic baseline, while improving system utility by up to almost an order of magnitude.

To demonstrate how FLEXICO can directly account for sustainability goals in the system utility definition, we analyze, for scenario C, the amount of CO_2 emissions produced by each baseline. Recall that for scenario C, fine-tune cost is defined as $\texttt{spotInstanceCost} \times (1 - \texttt{GCFE}) \times \texttt{gridCO}_2 \times \texttt{finetuneDuration} \times \texttt{TDP}_{\texttt{GPU}}$, so CO_2 emissions are directly factored in when FLEXICO is analyzing whether to fine-tune (cf. Figure 5).

The results are displayed in Figure 9, which shows the total cost (system utility) and CO_2 emissions incurred by each baseline for each delta threshold. Lower delta thresholds are represented by markers with higher transparency and the arrows denote moving from the lowest to the highest delta threshold. We see that as the delta threshold increases (i.e., MT quality needs to improve more for a fine-tune to be considered worthy), the CO_2 emitted by the Optimum oracle and by FLEXICO decreases. This is due to FLEXICO's ability to weigh the cost/benefits trade-offs of performing a fine-tune. Instead, the remaining baselines always consume the same amount of CO_2 regardless of the delta threshold considered because they are guided by static rules to determine whether to adapt. While for some baselines (e.g., Exponential-2) this may lead to a low CO_2 consumption, it comes at the expense of quite a high total cost (recall that penalties are charged whenever baselines miss adaptation opportunities, cf. 4). Additionally, we can also see that for the highest delta thresholds, FLEXICO emits almost an order of magnitude less CO_2 than the periodic baseline.

Just Accepted

Table 6. Latency of the adaptation strategy extraction process (in seconds), for both experimental scenarios.

	Avg ± stdev [s]			95-perc [s]			99-perc [s]		
Scenario	A	B	C	A	B	C	A	B	C
Formal verification	2.10 ± 0.13	2.43 ± 0.48	2.05 ± 0.04	2.48	2.87	2.14	2.68	3.17	2.63
MAPE-K	2.32 ± 0.14	2.66 ± 0.52	2.26 ± 0.05	2.74	3.14	2.36	2.96	3.46	2.86

6.7 Flexico's Latency for Online Planning (RQ6)

RQ6: Flexico's Decision Latency (see Table 6)

Flexico's decision-making latency of approximately 3 seconds is suitable for deciding online whether to fine-tune (or not) an MT model, particularly when considering the typically long fine-tuning durations.

Flexico's decision-making latency plays a crucial role in determining its applicability across different scenarios, applications, and domains: if users need to wait too long for Flexico's decision, such that executing a fine-tune is faster than waiting for Flexico to output the expected best tactic to execute, then users can simply deploy the fine-tune and compare the real benefits of fine-tuning with the benefits of not fine-tuning to decide which MT model to deploy. Although this is only possible if users do not have constraints regarding the monetary or energy costs of superfluous fine-tunes, which is typically not the case, we still wish for Flexico to provide recommendations quickly. Table 6 displays the latency measurements for Flexico when deciding on the next tactic to execute, for both scenarios considered in Section 6.5. As expected, the formal verification process 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.

6.8 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 [14] for all results).

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 [14].

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

Just Accepted

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.

7 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.

Additionally, when additional adaptation tactics are introduced or when a different set of MT evaluation metrics is required, Flexico must be updated accordingly. With respect to adaptation tactics, and as discussed in Section 5.1, extending Flexico's repertoire of adaptation tactics requires several modifications. First, FIPs and actuators (the mechanisms through which Flexico executes a tactic) must be implemented for each new tactic. Second, the adaptation manager must be updated so that it becomes aware of the newly available tactics and can consider them during decision making. Third, the monitoring component must be extended to ensure that all data required to compute the features used by the FIPs is available. Finally, the formal model must be updated to capture the impact of each new tactic on the MT system or MT model.

When the set of MT evaluation metrics changes, Flexico must also be extended to support their computation. Specifically, functions for computing the new metrics must be integrated into the system so that they can be invoked during evaluation. In addition, the MTQM (see Section 5.2) must be updated to reflect the revised set of metrics used to assess translation quality.

Regarding the FIP features, we argue that the *Basic* and *Content-aware* feature sets (see Section 5.4) 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 [50] to contain information regarding model adaptations (e.g. retrain, fine-tune). *Adaptation Cards* are meant to be publicly available on repositories such as GitHub [26] or HuggingFace [29] 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 5.4 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.

Just Accepted

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.

8 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 the environmental impact of model updates.

Acknowledgments

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 [9], which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

[1] [n. d.]. https://www2.statmt.org/wmt24/translation-task.html

[2] Milind Agarwal, Sweta Agrawal, Antonios Anastasopoulos, Luisa Bentivogli, Ondřej Bojar, Claudia Borg, Marine Carpuat, Roldano Cattoni, Mauro Cettolo, Mingda Chen, William Chen, Khalid Choukri, Alexandra Chronopoulou, Anna Currey, Thierry Declerck, Qianqian Dong, Kevin Duh, Yannick Estève, Marcello Federico, Souhir Gahbiche, Barry Haddow, Benjamin Hsu, Phu Mon Htut, Hirofumi Inaguma, Dávid Javorský, John Judge, Yasumasa Kano, Tom Ko, Rishu Kumar, Pengwei Li, Xutai Ma, Prashant Mathur, Evgeny Matusov, Paul McNamee, John P. McCrae, Kenton Murray, Maria Nadejde, Satoshi Nakamura, Matteo Negri, Ha Nguyen, Jan Niehues, Xing Niu, Atul Kr. Ojha, John E. Ortega, Proyag Pal, Juan Pino, Lonneke van der Plas, Peter Polák, Elijah Rippeth, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Yun Tang, Brian Thompson, Kevin Tran, Marco Turchi, Alex Waibel, Mingxuan Wang, Shinji Watanabe, and Rodolfo Zevallos. 2023. FINDINGS OF THE IWSLT 2023 EVALUATION CAMPAIGN. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*. Association for Computational Linguistics.

[3] Pangkuh Ajisoko. 2020. The use of Duolingo apps to improve English vocabulary learning. *International Journal of Emerging Technologies in Learning (iJET)* 15, 7 (2020).

[4] S. Andova, H. Hermanns, and J.-P Katoen. 2003. Discrete-time rewards model-checked. In *In Procs. of FORMATS* (Marseille, France). Springer.

[5] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051* (2020).

[6] Language Technology Research Group at the University of Helsinki. 2020. HuggingFace Helsinki-NLP/opus-mt-en-zh. https://huggingface.co/Helsinki-NLP/opus-mt-en-zh Accessed: 2024-10-31.

[7] Language Technology Research Group at the University of Helsinki. 2020. HuggingFace Helsinki-NLP/opus-mt-zh-en. https://huggingface.co/Helsinki-NLP/opus-mt-zh-en Accessed: 2024-12-11.

[8] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency.*

Just Accepted

[9] Timothy J Boerner, Stephen Deems, Thomas R Furlani, Shelley L Knuth, and John Towns. 2023. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing*. 173–176.
[10] Tomáš Bureš. 2021. Self-Adaptation 2.0. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE.
[11] Radu Calinescu et al. 2017. Synthesis and verification of self-aware computing systems. In *Self-Aware Computing Systems*. Springer.
[12] Jose Camacho-Collados and Mohammad Taher Pilehvar. 2020. Embeddings in natural language processing. In *Proceedings of the 28th international conference on computational linguistics: tutorial abstracts*.
[13] J. Cámara, W. Peng, D. Garlan, and B. Schmerl. 2018. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Science of Computer Programming* 167 (2018).
[14] Maria Casimiro. 2024. Appendix — FLEXICO: Sustainable Machine Translation via Self-Adaptation. https://doi.org/10.5281/zenodo.14452761
[15] Maria Casimiro, Diego Didona, Paolo Romano, Luis Rodrigues, Willy Zwaenepoel, and David Garlan. 2020. Lynceus: Cost-efficient tuning and provisioning of data analytic jobs. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 56–66.
[16] Maria Casimiro, Paolo Romano, José G. C. De Souza, Amin M. Khan, and David Garlan. 2025. Flexico: Sustainable Machine Translation via Self-Adaptation. In *2025 IEEE/ACM 20th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 147–158. doi:10.1109/SEAMS66627.2025.00024
[17] Maria Casimiro, Paolo Romano, David Garlan, Gabriel Moreno, Eunsuk Kang, and Mark Klein. 2021. Self-Adaptation for Machine Learning Based Systems. In *ECSA 2021 Companion Volume, Virtual (originally: Växjö, Sweden), 13-17 September, 2021 (CEUR Workshop Proceedings, Vol. 2978)*. CEUR-WS.org.
[18] Maria Casimiro, Paolo Romano, David Garlan, Gabriel A. Moreno, Eunsuk Kang, and Mark Klein. 2022. Self-adaptive Machine Learning Systems: Research Challenges and Opportunities. In *Software Architecture*. Springer International Publishing.
[19] Maria Casimiro, Paolo Romano, David Garlan, and Luís Rodrigues. 2022. Towards a Framework for Adapting Machine Learning Components. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE.
[20] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*.
[21] Shang-Wen Cheng, David Garlan, and Bradley Schmerl. 2006. Architecture-based Self-adaptation in the Presence of Multiple Objectives. In *ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*.
[22] Matteo Ciniselli, Nathan Cooper, Luca Pascarella, Antonio Mastropaolo, Emad Aghajani, Denys Poshyvanyk, Massimiliano Di Penta, and Gabriele Bavota. 2021. An empirical study on the usage of transformer models for code completion. *IEEE Transactions on Software Engineering* 48, 12 (2021).
[23] Google Cloud. [n. d.]. Spot VMs. https://cloud.google.com/solutions/spot-vms?hl=en Accessed: 2025-07-17.
[24] Per-Erik Danielsson. 1980. Euclidean distance mapping. *Computer Graphics and image processing* 14, 3 (1980).
[25] Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Houston, Texas, USA) *(ASPLOS '13)*. Association for Computing Machinery, New York, NY, USA, 77–88. doi:10.1145/2451116.2451125
[26] github. 2020. GitHub. https://github.com/
[27] Pinjia He, Clara Meister, and Zhendong Su. 2021. Testing machine translation via referential transparency. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE.
[28] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research* 21, 248 (2020).
[29] HuggingFace. 2016. HuggingFace. https://huggingface.co/
[30] John Hutchins. 2007. Machine translation: A concise history. *Computer aided translation: Theory and practice* 13, 29-70 (2007).
[31] Wayback Machine Internet Archive. [n. d.]. Crawl of carbon free energy for Google Cloud regions. https://web.archive.org/web/20211001000000*/https://cloud.google.com/sustainability/region-carbon Accessed: 2025-07-17.
[32] Heli Järvenpää, Patricia Lago, Justus Bogner, Grace Lewis, Henry Muccini, and Ipek Ozkaya. 2024. A synthesis of green architectural tactics for ml-enabled systems. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*.
[33] Nan Jiang, Thibaud Lutellier, and Lin Tan. 2021. Cure: Code-aware neural machine translation for automatic program repair. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE.
[34] Jeffrey O. Kephart and David M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003).
[35] Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Philipp Koehn, Benjamin Marie, Christof Monz, Makoto Morishita, Kenton Murray, Makoto Nagata, Toshiaki Nakazawa, Martin Popel, Maja Popović, and Mariya Shmatova. 2023. Findings of the 2023

Just Accepted

Conference on Machine Translation (WMT23): LLMs Are Here but Not Quite There Yet. In *Proceedings of the Eighth Conference on Machine Translation*. Association for Computational Linguistics, Singapore.
[36] Tom Kocmi, Christian Federmann, Roman Grundkiewicz, Marcin Junczys-Dowmunt, Hitokazu Matsushita, and Arul Menezes. 2021. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. *arXiv preprint arXiv:2107.10821* (2021).
[37] Tom Kocmi, Vilém Zouhar, Christian Federmann, and Matt Post. 2024. Navigating the metrics maze: Reconciling score magnitudes and accuracies. *arXiv preprint arXiv:2401.06760* (2024).
[38] Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.
[39] Shubham Kulkarni, Arya Marda, and Karthik Vaidhyanathan. 2023. 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.
[40] Marta Kwiatkowska, Gethin Norman, and David Parker. 2018. Probabilistic model checking: Advances and applications. *Formal System Verification: State-of the-Art and Future Trends* (2018).
[41] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700* (2019).
[42] Michael Austin Langford and Betty H.C. Cheng. 2021. "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.
[43] Sungjae Lee, Jaeil Hwang, and Kyungyong Lee. 2022. SpotLake: Diverse Spot Instance Dataset Archive Service. In *2022 IEEE International Symposium on Workload Characterization (IISWC)*. 242–255. doi:10.1109/IISWC55918.2022.00029
[44] Grace A Lewis, Stephany Bellomo, and Ipek Ozkaya. 2021. Characterizing and detecting mismatch in machine-learning-enabled systems. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE.
[45] Grace A. Lewis, Ipek Ozkaya, and Xiwei Xu. 2021. Software Architecture Challenges for ML Systems. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE.
[46] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991).
[47] Xiaoyi Ma. 2000. Hong Kong News Parallel Text. https://catalog.ldc.upenn.edu/LDC2000T46
[48] Arya Marda, Shubham Kulkarni, and Karthik Vaidhyanathan. 2024. 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)*.
[49] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Google Books Team, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science* 331, 6014 (2011).
[50] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*.
[51] G. Moreno, J. Cámara, D. Garlan, and M. Klein. 2018. Uncertainty reduction in self-adaptive systems. In *Procs. of SEAMS*.
[52] Henry Muccini and Karthik Vaidhyanathan. 2021. 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.
[53] OpenAI. 2022. ChatGPT. https://openai.com/chatgpt/
[54] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
[55] Fabian Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011).
[56] Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.
[57] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics.
[58] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
[59] Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. 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.
[60] Ricardo Rei, Marcos Treviso, Nuno M. Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José G. C. de Souza, Taisiya Glushkova, Duarte Alves, Luisa Coheur, Alon Lavie, and André F. T. Martins. 2022. 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.
[61] Danielle Saunders. 2022. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *Journal of Artificial Intelligence Research* 75 (2022).
[62] Bradley Schmerl, Javier Cámara, Jeffrey Gennari, David Garlan, Paulo Casanova, Gabriel A. Moreno, Thomas J. Glazier, and Jeffrey M. Barnes. 2014. Architecture-Based Self-Protection: Composing and Reasoning about Denial-of-Service Mitigations. In *HotSoS 2014: 2014*

Just Accepted

Symposium and Bootcamp on the Science of Security.
[63] Felix Stahlberg. 2020. Neural machine translation: A review. *Journal of Artificial Intelligence Research* 69 (2020).
[64] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34.
[65] Zeyu Sun, Jie M Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. 2022. Improving machine translation systems via isotopic replacement. In *Proceedings of the 44th international conference on software engineering*.
[66] Tatoeba. 2021. Opus-eng-fra MT model. https://github.com/Helsinki-NLP/Tatoeba-Challenge/blob/master/models/eng-fra/opus-2021-02-22.yml
[67] Meghana Tedla, Shubham Kulkarni, and Karthik Vaidhyanathan. 2024. EcoMLS: A Self-Adaptation Approach for Architecting Green ML-Enabled Systems. *arXiv preprint arXiv:2404.11411* (2024).
[68] Jörg Tiedemann. 2003. OPUS: The Open Parallel Corpus. https://opus.nlpl.eu/
[69] Jörg Tiedemann. 2020. 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.
[70] Jörg Tiedemann and Santhosh Thottingal. 2020. 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.
[71] J. Townsend. 1971. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics* 9, 1 (1971).
[72] Michele Tufano, Jevgenija Pantiuchina, Cody Watson, Gabriele Bavota, and Denys Poshyvanyk. 2019. On learning meaningful code changes via neural machine translation. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE.
[73] Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. 2019. An empirical study on learning bug-fixing patches in the wild via neural machine translation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28, 4 (2019).
[74] Marco Turchi, Matteo Negri, M Farajian, and Marcello Federico. 2017. Continuous learning from human post-edits for neural machine translation. *The Prague Bulletin of Mathematical Linguistics* 108, 1 (2017).
[75] Aimee Van Wynsberghe. 2021. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics* 1, 3 (2021).
[76] Roberto Verdecchia, Luís Cruz, June Sallou, Michelle Lin, James Wickenden, and Estelle Hotellier. 2022. Data-centric green ai an exploratory empirical study. In *2022 international conference on ICT for sustainability (ICT4S)*. IEEE.

Received 20 November 2025; revised 20 May 2026; accepted 24 July 2026

Just Accepted