Machine Translation for Human Translators

Michael Denkowski

CMU-LTI-15-004

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies*

Abstract

While machine translation is sometimes sufficient for conveying information across language barriers, many scenarios still require precise human-quality translation that MT is currently unable to deliver. Governments and international organizations such as the United Nations require accurate translations of content dealing with complex geopolitical issues. Community-driven projects such as Wikipedia rely on volunteer translators to bring accurate information to diverse language communities. As the amount of data requiring translation has continued to increase, the idea of using machine translation to improve the speed of human translation has gained significant traction. In the frequently employed practice of post-editing, a MT system outputs an initial translation and a human translator edits it for correctness, ideally saving time over translating from scratch. While general improvements in MT quality have led to productivity gains with this technique, the idea of designing translation systems specifically for post-editing has only recently caught on in research and commercial communities.

In this work, we present extensions to key components of statistical machine translation systems aimed directly at reducing the amount of work required from human translators. We cast MT for post-editing as an online learning task where new training instances are created as humans edit system output and introduce an adaptive MT system that immediately learns from this human feedback. New translation rules are learned from the data and both feature scores and weights are updated after each sentence is post-edited. An extended feature set allows making fine-grained distinctions between background and post-editing data on a per-translation basis. We describe a simulated post-editing paradigm wherein existing reference translations are used as a stand-in for human editing during system tuning, allowing our adaptive systems to be built and deployed without any seed post-editing data.

We present a highly tunable automatic evaluation metric that scores hypothesis-reference pairs according to several statistics that are directly interpretable as measures of post-editing effort. Once an adaptive system is deployed and sufficient post-editing data is collected, our metric can be tuned to fit editing effort for a specific translation task. This version of the metric can then be plugged back into the translation system for further optimization.

To both evaluate the impact of our techniques and collect post-editing data to refine our systems, we present a web-based post-editing interface that connects human translators to our adaptive systems and automatically collects several types of highly accurate data while they work. In a series of simulated and live post-editing experiments, we show that while many of our presented techniques yield significant improvement on their own, the true potential of adaptive MT is realized when all techniques are combined. Translation systems that update both the translation grammar and weight vector after each sentence is post-edited yield super-additive gains over baseline systems across languages and domains, including low resource scenarios. Optimizing systems toward custom, task-specific metrics further boosts performance. Compared to static baselines, our adaptive MT systems produce translations that require less mechanical effort to correct and are preferred by human translators. Every software component developed as part of this work is made publicly available under an open source license.

Acknowledgements[1]

This work would not have been possible without the wealth of ideas brought to life in conversations with my advisor, Alon Lavie, and my committee during my time at Carnegie Mellon University. I thank Alon for encouraging me to take a global perspective of the machine translation community and industry, considering the people and technology involved in every step of the translation process. Alon also encouraged working on a wide range of MT tasks, focusing research efforts where they could have the most significant impact. Many of these tasks, from human and automatic MT evaluation to large scale system building, came together to form this line of work. Finally, Alon's emphasis on collaboration led to many connections that were instrumental to bringing this work together.

I also thank the other members of my committee: Chris Dyer, Jaime Carbonell, and Gregory Shreve. Chris helped me to frame many of the research problems in this work, drawing connections between the MT and machine learning communities. One of the central themes of this work, casting MT for post-editing as an online learning task, was born from an animated research discussion with Chris and Alon. Jaime helped me to frame this work both in the history of computer-aided translation and in the current MT research landscape. Gregory helped me to connect this work to the translation studies community and provided a vital link that has led to further collaboration. Though not officially on my committee, I thank Isabel Lacruz for her invaluable help in organizing human translators for all of our experiments.

I thank my colleagues in the CMU machine translation group, with whom I have had more productive research conversations than I can recall: Jonathan Clark, Greg Hanneman, Kenneth Heafield, and Austin Matthews. Jon helped with hypothesis testing, allowing for reporting results more reliably. Greg and Austin provided valuable feedback on many parts of this work. Kenneth significantly improved the efficiency of our group's MT systems, allowing much larger experiments. I also thank the following CMU students working outside of my immediate research area that gave valuable perspective on this work: Kevin Gimpel, Matthew Marge, and Nathan Schneider.

I also thank everyone I have worked with at Safaba: Ryan Carlson, Matthew Fiorillo, Kartik Goyal, Udi Hershkovich, Laura Kieras, Robert Olszewski, and Sagi Perel. Working together to build production quality MT pipelines gave me a greater appreciation for the practical challenges of bringing developments from the research community to real world applications, in particular the importance of keeping real world constraints and end users in mind throughout the research and development process.

I finally thank my undergraduate advisors: Charles Hannon, J. Richard Rinewalt, and Antonio Sanchez. They originally introduced me to the area of natural language processing and afforded me the opportunity to work on research projects as an undergraduate. Their enthusiasm for pursuing knowledge was one of my inspirations for starting a graduate career in computer science.

[1]This work is supported in part by the National Science Foundation under grant IIS-0915327, by the Qatar National Research Fund (a member of the Qatar Foundation) under grant NPRP 09-1140-1-177, and by the NSF-sponsored Extreme Science and Engineering Discovery Environment program under grant TG-CCR110017.

Contents

Chapter 1

Introduction

Modern machine translation services such as Google Translate[1] and Microsoft's Bing Translator[2] have made significant strides toward allowing users to read content in other languages. These systems, built on decades of contributions from academic and commercial research, focus largely on this use case, aiming to maximize human understandability of MT output. For example, if an English speaking user wants to read an article posted on a Chinese language news site, a machine translation may contain the following lines[3]:

> UK GMT at 10:11 on March 20, a rare solar eclipse spectacle will come to Europe.
>
> This is the 1954 total solar eclipse once again usher in mainland Norway.
>
> The next solar eclipse occurs recent times and the country was March 9, 2016 Sumatra;

This translation is quite useful for casual readers, allowing them to glean key information from the article such as the event (a solar eclipse), location (mainland Norway), and time (10:11 on March 20). However, the grammatical errors and likely mistranslations throughout the text would prevent this article from being published as-is in English; readers would be unable to trust the information as they would be relying on their ability to guess what information is missing or mistranslated. If this article were to be published in English, it would require professional human translation. In fact, the ever-increasing need for highly accurate translations of complex content has led to the development of a vibrant professional translation industry. Global businesses, government organizations, and other projects employing translators spent an estimated \$37.19 billion worldwide on translation services in 2014 (DePalma et al., 2014).

1.1 Machine Translation for Post-Editing

As the demand for human quality translation increases, the idea of leveraging machine translation to improve the speed of human translation grows increasingly attractive. While MT is unable to directly produce publishable translations, recent work in academia and industry has shown significant success with the task of *post-editing*, having bilingual translators correct MT output rather than translate from scratch. When used with human post-editing, machine translation plays a fundamentally different role than in the traditional *assimiliation* use case. As human translators must edit MT output to produce human quality translations, the quality of MT is directly tied to editing difficulty rather than understandability. Minor disfluencies must be corrected even if they would not impair comprehension, while mistranslations can be resolved by retranslating words in the source sentence. As such, the types of translations that are best for post-editing are often

[1] `https://translate.google.com/`

[2] `http://www.bing.com/translator/`

[3] These lines are taken from a Google translation of an article on the Chinese language version of the Xinhua news website (`www.xinhuanet.com/`) collected March 23, 2015.

quite different from those best for assimilation (Snover et al., 2009; Denkowski and Lavie, 2010a). This reveals a mismatch where MT systems used for post-editing are engineered for and evaluated on a totally different task.

Beyond requiring different types of translations, assimilation and post-editing differ in terms of data availability. Machine translation is traditionally treated as a *batch* learning and prediction task. The various steps in model estimation (word alignment, phrase extraction, feature weight optimization, etc.) are conducted sequentially, resulting in a translation system with a static set of models and feature weights. This system is then used to translate unseen text. If new training data becomes available, the system must be entirely rebuilt, a process taking hours or days. In post-editing, the very act of translating with the system generates new training data. Post-editors provide a stream of human quality translations of input sentences as the system translates. As new data is immediately available after each sentence is translated, MT with post-editing can be treated as an *online* learning task that proceeds in a series of trials. For each input, the system first makes a prediction by generating a translation hypothesis. It is then shown a "gold standard" output, the post-edited translation. Finally, the system can use the newly generated bilingual sentence pair to update any components capable of making incremental updates. In traditional MT systems, this model update step is totally absent as batch models cannot be updated. Instead, the highly valuable data points generated by post-editing are simply added to the pool of new data to be included next time the system is rebuilt. As retraining is an expensive process, systems typically remain static for weeks or months. As a result, standard MT systems repeat the same translation errors despite constant correction and translators are forced to spend an unnecessarily large amount of their time repeating the same work.

This examination of the post-editing task and the limitations of standard MT systems highlights two areas where machine translation technology could better serve humans. First, translation systems capable of learning immediately from human feedback could avoid repeating the same mistakes. Second, by learning what types of translation errors are most costly for post-editing, systems' incremental learning could be guided by a more reliable objective function. Our work explores both of these points with a variety of extensions to standard MT systems.

The rest of this document is organized as follows. The following sections of this chapter present thesis statements, a summary of research contributions, details of the common setup used for all experiments, and summaries of the remaining major chapters. Chapter 3 describes our various extensions to standard translation models to facilitate online learning for MT. Chapter 4 describes an end-to-end post-editing pipeline using our original TransCenter interface and the results of live translation experiments conducted using this pipeline. Chapter 5 describes the challenges of MT evaluation for post-editing and experiments using our Meteor metric to predict editing effort for system optimization. Chapter 6 describes experiments with two low-resource languages: Dari and Pashto. Chapter 7 concludes the document with a summary of major results, discussion of promising future directions for each area of our work, and final remarks on the practical challenges of putting our adaptive MT technology into production for real world tasks in the professional translation industry. Appendix A lists all software and data released as part of our work.

1.2 Thesis Statements

We have introduced the components of current statistical machine translation systems and discussed initial efforts to integrate MT with human translation workflows. While general improvements in MT quality have led to improved performance and increased interest in this application, there has been relatively little work on designing translation systems specifically for post-editing. In this work, we present extensions to key components of MT pipelines that significantly reduce the amount of work required from human translators. We make the following central claims.

- The amount of work required of human translators can be reduced by translation systems that immediately learn from editor feedback.

- The usability of translations can be improved by automatically identifying the most costly types of translation errors and tuning MT systems to avoid them.
- The most significant gains in post-editing productivity are realized when several system components can learn in unison.

1.3 Research Contributions

As part of this work, we make the following contributions to the research community. These contributions form a unified framework that we term "real time adaptive machine translation":

Online Translation Grammar Adaptation: We have developed a method for immediately incorporating post-editing data into a translation model (Chapter 3). Rather than building a single monolithic translation grammar, we index the training bitext and extract a sentence level grammar for each input (Levenberg et al., 2010; Lopez, 2008a). As MT hypotheses are post-edited, the newly created source-target sentence pairs are immediately indexed in a separate data structure (Denkowski et al., 2014a). Subsequent grammars are extracted from the union of the background and post-edited data, allowing statistics to be shared. An additional "post-edit support" feature marks rules that are consistent with post-edited data, allowing an optimizer to prefer rules learned from or confirmed by humans. Translating with an adaptive grammar is shown to significantly improve performance on post-editing tasks.

Runtime Feature Weight Adaptation: We further leverage post-editing data by running an online learning algorithm that continuously updates feature weights during decoding (Chapter 3). We use a version of the margin infused relaxed algorithm (MIRA) to learn initial feature weights on fixed data during system development (Chiang et al., 2008; Eidelman, 2012). When decoding, the same algorithm is run on post-edited data to update feature weights after each sentence is processed. This allows the system's feature weights to scale in conjunction with translation grammars that add new data (Denkowski et al., 2014a). In many cases, updating both the grammar and feature weights yields super additive improvement over updating either independently.

Optimization and Evaluation with Simulated Post-Editing: We have developed a workflow for training and deploying adaptive MT systems for human translators using only the data normally available for building MT systems. We use pre-existing reference translations to simulate post-editing during optimization (Section 1.4.2), eliminating the need for live post-editing or even post-edited data during system development (Denkowski et al., 2014a). This allows us to build and evaluate (also using simulated post-editing) adaptive systems using any translation data set.

Validation Experiments with Simulated and Live Post-Editing: We demonstrate the effectiveness of our adaptive MT systems in both simulated and live post-editing scenarios (Chapter 4). Simulated post-editing experiments show consistent improvements across language directions and target domains while live experiments show that specific system extensions lead to real gains in translator productivity (Denkowski et al., 2014a; Denkowski et al., 2014b).

Automatic MT Evaluation Metrics for Post-Editing: We have developed an advanced automatic MT evaluation metric capable of fitting various measures of editing effort (Chapter 5). Originally designed for assimilation tasks, our Meteor metric is demonstrated to have strong predictive power for various types of edit operations (Denkowski and Lavie, 2011). When a version of Meteor tuned to post-editing effort is used as an objective function for system optimization, the resulting translations require less effort to edit than those from a BLEU-optimized system.

End-to-End MT Post-Editing Workflow: We have released software that constitutes a full, automated framework for building and deploying adaptive MT systems. This consists of the following components described in Section 1.4: Implementations of real time adaptive MT in the `cdec` and Moses toolkits plus tools for automating system building (Denkowski et al., 2014a), the TransCenter web-based post-editing interface for interacting with adaptive MT systems (Denkowski and Lavie, 2012b), and the Meteor metric for optimizing and evaluating systems for post-editing (Denkowski and Lavie, 2011).

1.4 Experimental Framework

To evaluate the impact of our work, we have assembled a test suite that represents a variety of real-world translation scenarios in four language directions: from English into and out of Spanish and Arabic. For each direction, we have extensive bilingual and monolingual data for model estimation as well as in-domain and out-of-domain evaluation sets with reference translations. We build a traditional machine translation system for each scenario to serve as both a baseline to compare results against and as a platform for implementing our extensions to standard translation models. Data and tools are selected with a focus on reproducibility. Spanish–English resources are freely available online and Arabic–English resources are available with a Linguistic Data Consortium (LDC) subscription. All tools other than MADA (Habash et al., 2009) are freely available under open source licenses.

Tools: We use several natural language processing toolkits to process training data and build translation systems. Part of our work includes significant contributions to some of these tools, specifically the suffix array grammar extractor, `cdec`, Moses, Meteor, and TransCenter.

- `cdec`: a statistical machine translation framework including a SCFG decoder, a word aligner, and implementations of several learning algorithms for structured prediction models (Dyer et al., 2010; Dyer et al., 2013). `cdec` is released under the Apache License at `http://github.com/redpony/cdec`.

- Moses: a widely used statistical machine translation framework including a phrase-based decoder, implementation of suffix array based translation models, and several optimizers (Koehn et al., 2007). Moses is released under the GNU LGPL at `http://statmt.org/moses/`.

- KenLM: a toolkit for estimating and conducting inference with N-gram language models (Heafield, 2011; Heafield et al., 2013). KenLM is released under the GNU LGPL at `kheafield.com/code/kenlm/` and also included with `cdec` and Moses.

- MADA: an Arabic natural language processing toolkit for tokenization, diacrization, morphological disambiguation, part-of-speech tagging, stemming and lemmatization (Habash et al., 2009). MADA is released under a non-commercial use license at `http://www1.ccls.columbia.edu/MADA/`.

- Meteor: an automatic metric for evaluation and optimization of machine translation systems (Banerjee and Lavie, 2005; Denkowski and Lavie, 2011). Meteor is released under the GNU LGPL at `http://github.com/mjdenkowski/meteor`.

- MultEval: implementation of several statistical significance tests for machine translation evaluation (Clark et al., 2011a). MultEval is released under the GNU LGPL at `http://github.com/jhclark/multeval`.

	Training Data		Evaluation Sets (sents)			
	Bilingual (sents)	Monolingual (words)	*WMT11*	WMT12	TED1	TED2
Spanish–English	2,104,313	1,175,142,205	*3003*	3003	2688	2978
English–Spanish	2,104,313	304,262,351	*3003*	3003	2688	2978

	Training Data		Evaluation Sets (sents)			
	Bilingual (sents)	Monolingual (words)	*MT08*	MT09	TED1	TED2
Arabic–English	5,027,793	651,957,491	*1356*	1313	2690	2846
English–Arabic	5,027,793	168,323,504	*1356*	1313	2690	2846

Table 1.1: Training, development, and evaluation data sizes for all experimental scenarios. Italics indicate that a data set is used for system optimization. The MT08 and MT09 sets have 4 English reference translations for each Arabic source sentence. All other data sets, including the English–Arabic directions of MT08 and MT09, have a single reference for each source sentence.

- Suffix array grammar extractor: an implementation of suffix array-based grammar extraction for hierarchical phrase-based machine translation that has been repackaged as part of `cdec` (Lopez, 2008a; Chahuneau et al., 2012).

- TransCenter: a web-based framework for post-editing data collection and analysis that integrates with real time adaptive MT systems (Denkowski and Lavie, 2012b; Denkowski et al., 2014b). TransCenter is released under the GNU LGPL at `http://github.com/mjdenkowski/transcenter`.

Data: We select four language directions for translation post-editing experiments: Spanish-to-English, English-to-Spanish, Arabic-to-English, and English-to-Arabic. Bilingual resources are identical between directions for each language pair while monolingual resources are unique for each language direction. Training data for Spanish–English includes all constrained resources for the 2013 ACL Workshop on Statistical Machine Translation (WMT)[4] (Bojar et al., 2013), consisting of European parliamentary proceedings and news commentary. Training data for Arabic–English includes all constrained bilingual resources for the 2012 NIST Open Machine Translation Evaluation (OpenMT12)[5] (Przybocki, 2012), consisting largely of news, and a selection from the English Gigaword (Parker et al., 2011). For each language direction, we have four evaluation sets: two drawn from similar domains as the training data, and two drawn from broader domains. For similar-domain sets, we use the 2011 and 2012 WMT news test sets for Spanish–English (dev and devtest from WMT13) and the 2008 and 2009 OpenMT mixed news and weblog test sets for Arabic–English. For broad-domain sets, we use sections of the Web Inventory of Transcribed and Translated Talks (WIT^3) corpus[6] (Cettolo et al., 2012) that contains multilingual transcriptions of TED[7] talks. The test sets *TED1* and *TED2* each contain bilingual transcriptions of 10 TED talks delivered from a wide range of speakers on a variety of topics. All data sets are pre-segmented into sentences that are grouped by document. We apply further *tokenization*, splitting sentences into individual words that can be processed by alignment and translation models. English and Spanish are tokenized with a general-purpose tokenizer included with *cdec* while Arabic is tokenized using MADA. Details for all training and evaluation data sets after tokenization are shown in Table 1.1.

[4]`http://statmt.org/wmt13/translation-task.html`
[5]`http://www.nist.gov/itl/iad/mig/openmt12.cfm`
[6]`https://wit3.fbk.eu/`
[7]`http://www.ted.com/pages/about`

	Feature	Definition
Phrase Features	`CoherentP(e|f)`	Eqn 2.21
	`Count(f,e)`	Eqn 2.18
	`SampleCount(f)`	Eqn 2.19
	`Singleton(f)`	Eqn 2.20
	`Singleton(f,e)`	Eqn 2.20
Lexical Features	`MaxLex(e|f)`	Eqn 2.11
	`MaxLex(f|e)`	Eqn 2.11
Language Model Features	`LM(E)`	Eqn 2.14
	`LM_OOV(E)`	Eqn 2.15
Derivation Features	`Arity(0)`	§ 2.2.4
	`Arity(1)`	§ 2.2.4
	`Arity(2)`	§ 2.2.4
	`GlueCount`	§ 2.2.4
	`PassThroughCount`	§ 2.2.5
	`WordCount`	§ 2.2.5

Table 1.2: Standard (baseline) feature set for hierarchical phrase-based machine translation with suffix array grammars

1.4.1 Baseline System

System Building: Translation systems are built using the same methods for each language pair. Data is word aligned source-to-target and target-to-source using the `fast_align` word aligner included with `cdec` (Dyer et al., 2013). Alignments are symmetrized using the `grow-diag-final-and` heuristic (Koehn et al., 2005). Translation grammars with the standard feature set listed in Table 1.2 (see §2.2.2–2.2.6 for feature descriptions) are extracted using the suffix array grammar extractor. Unpruned, modified Kneser-Ney smoothed (Chen and Goodman, 1996) language models are estimated using KenLM. Two baseline optimization scenarios are evaluated. In the first, feature weights are learned using the implementation of lattice-based minimum error rate training (Och, 2003; Macherey et al., 2008) included with `cdec`. In the second, the cutting plane version of the margin infused relaxed algorithm (Chiang et al., 2008; Eidelman, 2012) is used. MERT and MIRA optimize the BLEU score (Papineni et al., 2002) on the development set for the language pair (WMT10 for Spanish–English or MT08 for Arabic–English).

Automatic Evaluation: We use the `cdec` decoder to translate all evaluation sets into the target language using the same set of feature weights learned from the development set. To simulate a scenario where no in-domain data is available, we do not re-tune systems for TED talks. Translations are evaluated automatically with BLEU (Papineni et al., 2002), TER (Snover et al., 2006), and Meteor 1.4 (Denkowski and Lavie, 2011). When comparing extended systems to baselines, we use the following techniques described by Clark et al. (2011a) to test for statistically significant differences in score. First, we account for optimizer instability by running MERT or MIRA 3 times for each language direction and decoding the evaluation sets with each set of weights. The reported metric score for a data set is the average of three independent tune-test cycles. Second, we use the three outputs for each evaluation set to conduct approximate randomization, a statistical significance test that computes a probability p that differences in score between the baseline and extended system arose by chance by randomly shuffling hypotheses between systems and optimizer runs (Noreen, 1989).

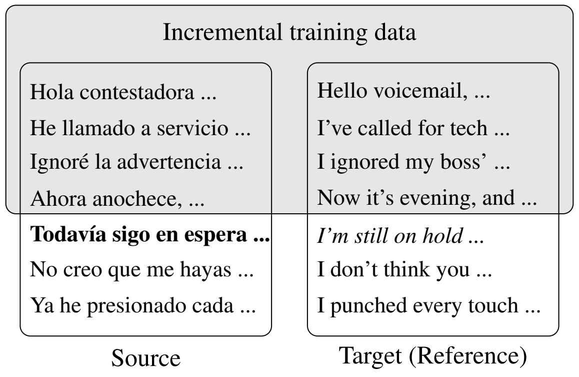


Figure 1.1: Context when translating an input sentence (bold) with simulated post-editing. Previous sentences and references (shaded) are added to the training data. *After* the current sentence is translated, it is aligned to the reference (italic) and added to the context for the next sentence.

1.4.2 System Building for Post-Editing

Simulated Post-Editing: In post-editing scenarios, the humans continuously edit machine translation outputs into production-quality translations, providing an additional, constant stream of data absent in batch translation. This data consists of highly domain-relevant reference translations that are minimally different from MT outputs, making them ideal for learning. However, true post-editing data is infeasible to collect during system development and internal testing as standard MT pipelines require tens of thousands of sentences to be translated with low latency. To address this problem, Hardt and Elming (2010) formulate the task of simulated post-editing, wherein pre-generated reference translations are used as a stand-in for actual post-editing. This approximation is equivalent to the case where humans edit each translation hypothesis to be identical to the reference rather than simply correcting the MT output to be grammatical and meaning-equivalent to the source. Our work uses this approximation when optimizing and evaluating adaptive MT systems.

In our simulated post-editing tasks, decoding (for both the test corpus and each pass over the development corpus during optimization) begins with baseline models trained on standard bilingual and monolingual data. After each sentence is translated, the following steps (explained in detail in Chapter 3 take place in order: First, MIRA uses the new source–reference pair to update weights for the current models. Second, the source is aligned to the reference (using the model learned from word-aligning the background data) and the new sentence pair is used to update the translation grammar. As sentences are translated, the models gain valuable context information, allowing them to zero in on the target document and translator. Context is reset at the start of each development or test corpus by discarding all incremental data from the translation model. For evaluation sets, the weight vector is also reset to the values chosen by optimization. For development sets, the weight vector persists. This setup, which allows a uniform approach to tuning and decoding, is visualized in Figure 1.1. We use simulated post-editing to evaluate each extension we make to standard MT systems across all language directions and domains listed above.

Live Post-Editing: To validate that gains in simulated post-editing scenarios translate to real improvements in translator efficiency, we employ a pool of human translators to work with our adaptive MT systems. These translators are students at Kent State University's Institute for Applied Linguistics pursuing careers as professional translators for the language directions we are interested in. Given the time and resource costs to conduct these live trials, we choose one language direction, Spanish-to-English, and a few key system conditions to evaluate. In each condition, a baseline system is compared to an extended system on a fixed set of documents. Each translator is asked to post-edit machine translations for each document. The translations are provided by either the baseline or extended system, without the translator knowing which system he or she is working with. Conditions are shuffled such that (1) each document is translated and post-edited multiple times with each system and (2) each translator only sees each document once. The results can be aggregated and analyzed to determine which system provides the most usable translations overall.

1.5 Executive Summary

The following sections summarize the major chapters of the thesis, including overviews of the techniques introduced and reports of the key results. Background information for these sections and their corresponding chapters, including an introduction to statistical machine translation, can be found in Chapter 2.

1.5.1 Online Learning for Machine Translation

Machine translation for post-editing can be cast as an *online* learning task that proceeds as a series of trials. In online learning, each trial consists of three stages: (1) the model makes a prediction, (2) the model receives the "true" answer, and (3) the model updates its parameters. Post-editing workflows fit naturally into this paradigm. In the prediction stage, the translation system produces an initial hypothesis. A human post-editor then edits the hypothesis to produce a "true" translation. Finally, the system uses the new source-target sentence pair to update the translation model. While traditional MT systems are unable to operate in this way, we introduce three extensions that allow systems to incorporate new data from post-editors in real time. These extensions include a translation grammar extraction algorithm that immediately incorporates new training instances, the practice of running an online optimizer during decoding, and an extended feature set that allows the translation model to leverage multiple sources of data. Combining these individual components results in a highly adaptive MT system that immediately learns from human feedback and can avoid making the same mistakes repeatedly.

Online Translation Grammar Adaptation: We begin with the on-demand translation model described in §2.2.6 (Lopez, 2008a; Lopez, 2008b). Rather than using all bilingual training data to build a single, large translation grammar, this approach uses a suffix array to index the data so that grammars can be estimated as needed. When a new sentence needs to be translated, the suffix array is used to rapidly build and score a sentence-specific grammar. Rules in on-demand grammars are generated using a sample $\mathcal{S}$ for each source phrase $\bar{f}$ in the input sentence. The sample, containing phrase pairs $\langle \bar{f}, \bar{e} \rangle$, is used to calculate the following statistics:

- $\mathcal{C}_{\mathcal{S}}(\bar{f}, \bar{e})$: count of instances in $\mathcal{S}$ where $\bar{f}$ aligns to $\bar{e}$ (phrase co-occurrence count).
- $\mathcal{C}_{\mathcal{S}}(\bar{f})$: count of instances in $\mathcal{S}$ where $\bar{f}$ aligns to any target phrase.
- $|\mathcal{S}|$: total number of instances in $\mathcal{S}$, equal to number of occurrences of $\bar{f}$ in training data, up to the sample size limit.

These statistics are used to instantiate and compute a set of feature scores for translation rules $X \rightarrow \bar{f} / \bar{e}$.

To accommodate new bilingual data from post-editing, we also maintain a dynamic lookup table for incremental data (Denkowski et al., 2014a). When a human translator edits a MT hypothesis, the sentence pair resulting from the input sentence and post-edited translation is word-aligned with the same model used for the initial data, a process often called forced alignment (Gao and Vogel, 2008). Aligned phrase pairs are then stored in the lookup table and phrase occurrences are counted on the source side. When an on-demand grammar is extracted, the suffix array sample $\mathcal{S}$ for each $\bar{f}$ is accompanied by an exhaustive lookup $\mathcal{L}$ from the lookup table. Statistics matching those from $\mathcal{S}$ are calculated from $\mathcal{L}$:

- $\mathcal{C}_{\mathcal{L}}(\bar{f}, \bar{e})$: count of instances in $\mathcal{L}$ where $\bar{f}$ aligns to $\bar{e}$.
- $\mathcal{C}_{\mathcal{L}}(\bar{f})$: count of instances in $\mathcal{L}$ where $\bar{f}$ aligns to any target phrase.
- $|\mathcal{L}|$: total number of instances of f in post-editing data (no size limit).

The statistics are aggregated (simple summation) and translation rules are instantiated. In addition to the regular feature set, a "post-edit support" indicator feature marks rules that are consistent with post-editor feedback. This allows an optimizer to learn an additional weight for rules that are consistent with human feedback. As the underlying translation model is hierarchical, it can also learn new non-local reordering patterns from post-editing data.

Online Parameter Optimization: MT systems are traditionally optimized in batch mode at the corpus level. Optimization begins with a fixed translation model and an initial set of feature weights. For a given development corpus of bilingual source-target sentences, the MT system uses the model and initial weights to produce a list of the most likely hypotheses for each source sentence. An optimizer such as minimum error rate training (Och, 2003) is then used to select a new set of feature weights that prefers better scoring hypotheses from each list. Once a new set of feature weights is chosen, the MT system re-translates the development corpus using the new weights and the process continues. Once a stopping point is reached (either completing an iteration that produces no previously unseen hypotheses or reaching a fixed limit on the number of iterations), the current set of feature weights is used as the system's final static weight vector.

We use the margin-infused relaxed algorithm described in §2.3.2 (Crammer et al., 2006a; Chiang et al., 2008; Eidelman, 2012), an online learning algorithm that makes an adjustment to the weight vector after each sentence in the development corpus is translated. However, the confines of batch MT system development require this algorithm to be run in batch mode, similar to MERT. Beginning with a uniform weight vector, MIRA makes a fixed number of passes over the development corpus to iteratively refine the weights. In each pass, each sentence is translated and a parameter update is made. This is problematic for post-editing as our translation model is updated after each sentence, yet the optimizer must ultimately learn a single set of feature weights. In a linear translation model, a single weight cannot adequately scale a feature that becomes more powerful over time.

To address the limitations of batch learning and better fit the online learning paradigm, we continue running the MIRA optimizer during decoding when new input sentences are translated. For each document to be translated, we begin with the set of feature weights resulting from batch MIRA on the simulated post-editing development corpus. As each sentence is translated and post-edited (or simulated with a reference translating), MIRA makes an update to the weight vector just as in optimization. Running MIRA during decoding allows the feature weights as well as the translation grammar to be up to date with all available post-editing data when each sentence is translated. In the only departure from optimization, we increase regularization strength during decoding to prefer smaller weight updates. While we use MIRA in our work, any online learning algorithm can be substituted just as different batch optimizers can be plugged into the standard MT system tuning step.

Extended Post-Editing Feature Set: Our original formulation of online grammar extraction is still limited by its use of the same feature set as the original static model. Simply summing the sufficient statistics from samples of background and post-editing data restricts the interpolation weight for each data source to be identical and uniform across feature scores. All responsibility for addressing this problem is placed on the single post-edit support feature. We address this issue with an extended feature set that presents the decoder with more fine grained information about the likelihood of translation rules in background and post-editing

	Spanish–English				English–Spanish			
	WMT11	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*29.3*	31.6	34.0	30.2	*30.5*	30.9	27.0	26.5
PE Support	*30.1*	32.1	35.7	32.0	*31.4*	**31.7**	28.4	27.8
Extended	*30.7*	32.4	**36.2**	32.1	*31.6*	**31.7**	28.8	28.2
+ Weights	*30.9*	**33.0**	36.1	**32.4**	*31.6*	**31.7**	**29.8**	**28.9**
	Arabic–English				English–Arabic			
	MT08	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*22.2*	26.0	11.2	11.5	*19.1*	23.7	7.8	8.7
PE Support	*22.7*	26.8	14.7	15.8	*19.6*	24.0	8.5	9.4
Extended	*23.1*	27.5	**15.1**	15.8	*20.2*	24.6	9.3	10.3
+ Weights	*23.1*	**27.8**	**15.1**	**16.0**	*20.1*	**24.8**	**9.5**	**10.7**

Table 1.3: BLEU scores for baseline systems, simple and extended online feature sets, and fully adaptive systems with runtime weight updates. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All adaptive systems (PE Support, Extended, and + Weights) show statistically significant improvement over respective baselines ($p < 0.05$ in approximate randomization).

data. When scoring each translation rule $X \to \bar{f}/\bar{e}$ with a background suffix array sample $\mathcal{S}$ and dynamic post-editing lookup $\mathcal{L}$, we compute three instances of each feature h:

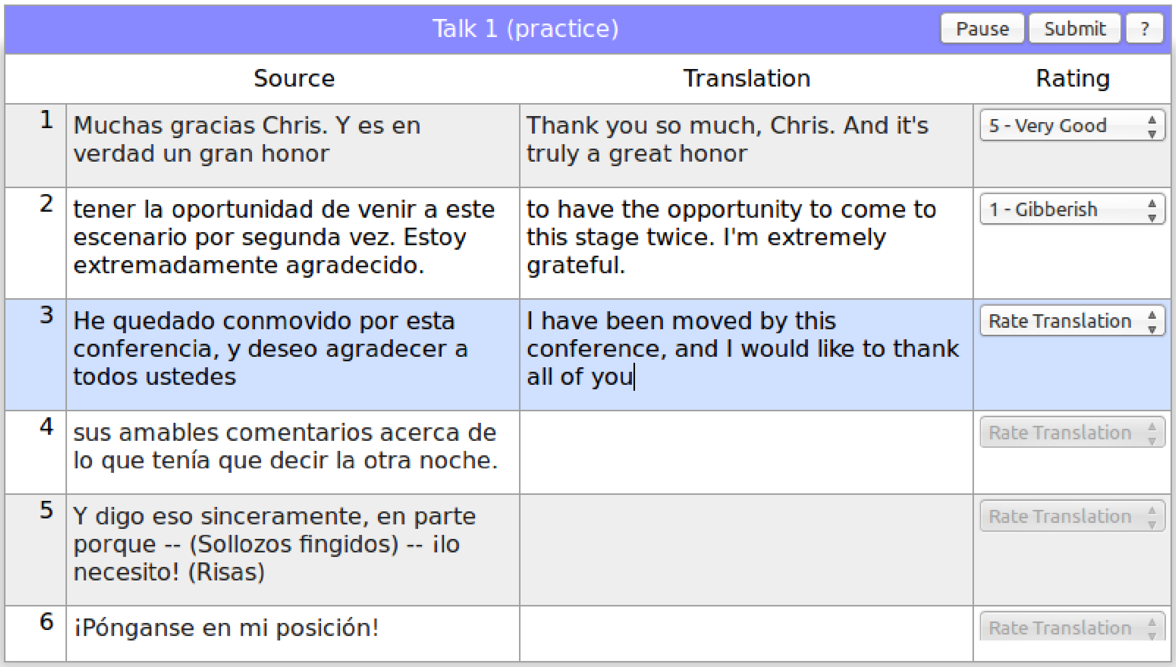
- $h_{\mathcal{S}\cup\mathcal{L}}$: the feature score computed on the union (aggregation) of background and post-editing data, equivalent to the version of h in the online translation model
- $h_{\mathcal{S}}$: the feature score computed only on the background data, equivalent to the version of h in the static baseline model
- $h_{\mathcal{L}}$: the feature score computed only on the post-editing data

Each feature is visible to the decoder and has an independent feature weight, effectively tripling the size of the phrase feature set. This allows the translation system to weigh the contribution of background versus post-editing data on a per-feature basis. In a linear translation model, this is formally equivalent to the following process. For each rule, the weighted score for each feature $w \cdot h$ is computed as follows:

$$w \cdot h = w_{\mathcal{S}\cup\mathcal{L}} \cdot h_{\mathcal{S}\cup\mathcal{L}} + w_{\mathcal{S}} \cdot h_{\mathcal{S}} + w_{\mathcal{L}} \cdot h_{\mathcal{L}} \tag{1.1}$$

First, the score is initialized with the weighted value from the union of background and incremental data (equivalent to the simple online grammar feature). Next, the score is adjusted up or down by the weighted score from the background data. Finally, the score is adjusted by the weighted score from post-editing data. The score for each component is up to date with all incremental data for the current sentence and the weight reflects the optimizer's confidence in each data source at the current point in the document. In a final extension, we replace the single post-edit support feature with features that count the number of rules and words originating from each data source.

Evaluating Adaptive MT Systems: We evaluate our MT system extensions in all simulated post-editing scenarios outlined in §1.4, translating a mixture of language directions and domains that cover a broad range of difficulty levels. We begin with a static baseline system that uses the on-demand translation model and is optimized with MIRA. We then evaluate three adaptive MT systems. The first uses online grammar



	Source	Translation	Rating
1	Muchas gracias Chris. Y es en verdad un gran honor	Thank you so much, Chris. And it's truly a great honor	5 - Very Good
2	tener la oportunidad de venir a este escenario por segunda vez. Estoy extremadamente agradecido.	to have the opportunity to come to this stage twice. I'm extremely grateful.	1 - Gibberish
3	He quedado conmovido por esta conferencia, y deseo agradecer a todos ustedes	I have been moved by this conference, and I would like to thank all of you	Rate Translation
4	sus amables comentarios acerca de lo que tenía que decir la otra noche.		Rate Translation
5	Y digo eso sinceramente, en parte porque -- (Sollozos fingidos) -- ¡lo necesito! (Risas)		Rate Translation
6	¡Pónganse en mi posición!		Rate Translation

Figure 1.2: Screenshot of the TransCenter post-editing and rating interface

extraction with the standard feature set plus a single post-edit support feature. The second uses online grammar extraction with an extended feature set. Both of these systems use a static weight vector. The third system uses online grammar extraction with an extended feature set and updates weights after each sentence is translated. Experiments are conducted using the Moses decoder (Koehn et al., 2007).

Shown in Table 1.3, all online systems outperform all baselines in all cases. Our extended feature set frequently leads to significant gains over the simple online feature set, especially in out-of-domain scenarios. Combining the extended feature set with runtime weight updates leads to significant and in some cases super-additive improvement. Using sentence-specific weights to keep pace with sentence-specific translation grammars allows our system to leverage much more of the potential in post-editing data.

1.5.2 Live Post-Editing Evaluation: Software and Experiments

While translation model adaptation, optimization, and evaluation experiments can all be carried out with simulated data, the ultimate goal of our work is to produce real time adaptive MT systems that can be used by actual human translators. As such, the most important measure of efficacy for our systems is the ultimate impact on human productivity in live translation scenarios. Measuring human productivity requires conducting live post-editing experiments, which in turn require an interface between translators and our MT systems. To address this need, we have developed a lightweight web-based translation editing environment called *TransCenter* in which users translate documents by post-editing MT output served by our adaptive systems (Denkowski and Lavie, 2012b; Denkowski et al., 2014b). As translators work, the adaptive systems learn from their post-edits in real time and TransCenter records all user activity. This forms an end-to-end translation and post-editing pipeline that is used to evaluate our adaptive MT systems.

TransCenter: Post-Editing User Interface: Shown in Figure 1.2, our software uses a simple user interface that follows the two column format that translators are familiar with (Denkowski and Lavie, 2012b). The left

	Sim PE BLEU	HTER	Rating
Baseline	34.50	19.26	4.19
Adaptive	**34.95**	**17.01**	**4.31**

Table 1.4: Aggregate simulated post-editing BLEU scores, HTER scores, and average translator self-ratings (5 point scale) of post-editing effort for translations of TED talks from Spanish into English.

column displays the source sentences while the right column is incrementally populated with translations from one of our MT systems as the user works. For each sentence, the translator is asked to edit the MT output to be grammatically correct and convey the same information as the source sentence. After editing, the final translation is archived and (if the system is adaptive) fed back to the MT system for learning (Denkowski et al., 2014b). The next sentence is then machine translated and post-edited. The user is additionally asked to rate the amount of work required to post-edit each sentence immediately after completing it, using a scale that ranges from 5 (no post-editing required) to 1 (requires total re-translation).

Our software is designed to make the barrier of collecting post-editing data as low as possible. TransCenter includes the following features to provide a smooth user experience:

- The editing interface is accessed via web browser so that users can work from any computer with an Internet connection.
- TransCenter automatically tracks state and communicates with the underlying MT system to support stopping and resuming tasks in the case of interruption.
- An uncluttered interface design allow users to focus on the editing task with minimal distraction.
- Full keyboard navigation allows all translation editing and rating to be completed without using the mouse.
- A Pause button allows users to take breaks if necessary.

Data Collection: In addition to gathering final edited translations and user ratings, TransCenter records all user interaction at a level of detail sufficient to replay the entire post-editing session. This includes number of keystrokes, number of milliseconds each sentence is focused, and a millisecond-timestamped record of each individual keystroke. Our software uses this information to generate reports of the following measures of human effort:

- The edit distance between the original MT output and the post-edited output according to HTER
- The user rating of each sentence's usability for post-editing
- The number of milliseconds each sentence is focused for editing
- The number of distinct keystrokes used to edit each sentence
- The number of atomic edit operations (insertions or deletions) used in editing each sentence
- Two *pause measures*: average pause ratio (APR) and pause to word ratio (PWR) (Lacruz et al., 2012; Lacruz and Shreve, 2014a)

Live Post-Editing Experiments: Connecting TransCenter to our MT systems forms a complete post-editing pipeline that enables us to run live evaluations to measure the effect of our online model adaptation techniques on human productivity. These experiments are conducted in collaboration with Kent State University's Institute for Applied Linguistics[8], an academic institution for training professional translators. We establish an experimental setup wherein observing a pool of human translators can determine which of two MT systems ("A" and "B") is better for post-editing. First, an even number of evaluation documents is selected. Next, translators are assigned to one of two groups ("odd" or "even") based on their (sequentially assigned) user ID. Each user is then asked to translate each document by post-editing the outputs of a MT system. For odd numbered documents, odd numbered translators use MT system A while even numbered users use system B. For even documents, odd translators use system B while even translators use system A. TransCenter does not display any information about which system is being used to translators.

In the first round of live experiments, we compare the static baseline system described in §1.4 to an adaptive system that updates both translation grammar and weights after each sentence.[9] For our evaluation documents, we draw four excerpts from TED talks that have been translated from Spanish to English, totalling 100 sentences. Our translators are five graduate students from the applied linguistics program training to be Spanish–English translators. Each student uses TransCenter to post-edit MT outputs for each document and all user interaction is logged. We also match previously reported results by running our simulated post-editing pipeline on the talk evaluation documents and reporting BLEU scores. Shown in Table 1.4, with this small amount of data, the improvement from the adaptive system is less than half of a point. However, when we evaluate the actual human data, we see a significant improvement in HTER and a slight user preference. This provides evidence that (1) simulated post-editing gains are a good indicator that there will be actual human effort savings, and (2) small gains in simulated scenarios can translate to significant gains in actual post-editing.

1.5.3 Automatic Metrics of Post-Editing Effort: Optimization and Evaluation

Traditionally, machine translation is treated as a final product that humans will use to read content in their native languages and other language technologies such as information retrieval systems will use directly as input. Approaches to both human and automatic evaluation focus on improving the adequacy of MT system output for these purposes. In contrast, post-editing uses MT as an intermediate step to reduce the amount of work required by human translators. Whereas translation models that incorporate post-editing feedback target this task in terms of model estimation, automatic metrics that accurately evaluate the amount of work required to edit translation hypotheses target post- editing in terms of parameter optimization. Pairing online models with automatic post-editing metrics enables end-to-end translation systems specifically targeting human translation.

The Meteor Metric: Meteor is an automatic evaluation metric that scores MT hypotheses by aligning them to reference translations (Denkowski and Lavie, 2011). Alignments are based on several types of flexible matches that go beyond surface forms to identify word and phrase correspondences that would be clear to humans but are missed by standard metrics. Given a translation hypothesis E' and reference translation E, Meteor creates an alignment as follows. First, the search space of possible alignments is constructed by identifying all possible matches between the two sentences according to the following matchers:

- Exact: Match words if their surface forms are identical.
- Stem: Stem words using a language-appropriate Snowball stemmer (Porter, 2001) and match if the stems are identical.

[8] `http://appling.kent.edu/`
[9] This system does not use the extended post-editing feature set as this was a later development.

- Synonym: Match words if they share membership in any synonym set according to the WordNet (Miller and Fellbaum, 2007) database.
- Paraphrase: Match phrases if they are listed as paraphrases in the Meteor paraphrase tables (Denkowski and Lavie, 2011).

All matches are generalized to phrase matches in the form $\langle E'^{i+n}_{i}, E^{j+m}_{j} \rangle$ where i and j are start indices in the hypothesis and reference and n and m are match lengths. Matches *cover* one or more words in each sentence. Once matches are identified, the final alignment is resolved as the largest subset of non-overlapping matches across both sentences.

Given an alignment between hypothesis E' and reference E, the Meteor metric score is calculated as follows. First calculate initial statistics:

- $\langle \mathcal{C}_{\mathtt{f}}(E'), \mathcal{C}_{\mathtt{f}}(E) \rangle$: function words in E' and E. Count any word that appears in the Meteor function word lists estimated from large monolingual data (Denkowski and Lavie, 2011).
- $\langle \mathcal{C}_{\mathtt{c}}(E'), \mathcal{C}_{\mathtt{c}}(E) \rangle$: content words in E' and E. Count any word that does not appear in the function word lists.
- $\langle h_i(\mathcal{C}_{\mathtt{c}}(E')), h_i(C_{\mathtt{f}}(E')), h_i(\mathcal{C}_{\mathtt{c}}(E)), h_i(\mathcal{C}_{\mathtt{f}}(E)) \rangle$: the number of content and function words in E' and E covered by each type of match h_i. (For example, counts of content and function words covered by exact matches in the hypothesis and reference.)
- `Ch`: the minimum number of *chunks* (series of matches that are contiguous and identically ordered in both sentences) that the alignment can be divided into.

Calculate weighted precision and recall using match type weights $w_i \in W$ and content-vs-function word weight (δ):

$$\mathcal{P} = \frac{\sum_i (w_i \times (\delta \times h_i(\mathcal{C}_{\mathtt{c}}(E')) + (1-\delta) \times h_i(\mathcal{C}_{\mathtt{f}}(E'))))}{\delta \times \mathcal{C}_{\mathtt{c}}(E') + (1-\delta) \times \mathcal{C}_{\mathtt{f}}(E')} \tag{1.2}$$

$$\mathcal{R} = \frac{\sum_i (w_i \times (\delta \times h_i(\mathcal{C}_{\mathtt{c}}(E)) + (1-\delta) \times h_i(\mathcal{C}_{\mathtt{f}}(E))))}{\delta \times \mathcal{C}_{\mathtt{c}}(E) + (1-\delta) \times \mathcal{C}_{\mathtt{f}}(E)} \tag{1.3}$$

The harmonic mean of $\mathcal{P}$ and $\mathcal{R}$ parameterized by α (van Rijsbergen, 1979) is then calculated:

$$\mathcal{F}_\alpha = \frac{\mathcal{P} \times \mathcal{R}}{\alpha \times \mathcal{P} + (1-\alpha) \times \mathcal{R}} \tag{1.4}$$

A fragmentation score is calculated using the total number of matched words `M` (average over hypothesis and reference) and number of chunks (`Ch`):

$$\mathtt{M} = \frac{\sum_i (h_i(\mathcal{C}_{\mathtt{c}}(E')) + h_i(\mathcal{C}_{\mathtt{f}}(E')) + h_i(\mathcal{C}_{\mathtt{c}}(E))) + h_i(\mathcal{C}_{\mathtt{f}}(E))}{2} \qquad \mathtt{Frag} = \frac{\mathtt{Ch}}{\mathtt{M}} \tag{1.5}$$

The final Meteor score is calculated with fragmentation parameters β and γ:

$$\mathtt{Meteor}(E', E) = \left(1 - \gamma \times \mathtt{Frag}^{\beta}\right) \times \mathcal{F}_\alpha \tag{1.6}$$

Each of the Meteor scoring statistics can be interpreted as a key predictor of post-editing effort. Precision ($\mathcal{P}$) is an inverse measure of the amount of content in the hypothesis that must be *deleted* to match the reference. Recall ($\mathcal{R}$) inversely measures the amount of content that must be *inserted*. Fragmentation (`Ch`) is a measure of how much *reordering* is required to match the reference. The parameters $W = \langle w_i, ..., w_n \rangle$,

p	HTER	Rating	Keystroke	Time	APR	PWR
HTER	–	-0.84	0.91	0.61	-0.55	0.64
Rating	-0.84	–	-0.82	-0.56	0.46	-0.53
Keystroke	0.91	-0.82	–	0.70	-0.56	0.66
Time	0.61	-0.56	0.70	–	-0.53	0.69
APR	-0.55	0.46	-0.56	-0.53	–	-0.65
PWR	0.64	-0.53	0.66	0.69	-0.65	–

Table 1.5: Spearman's correlation between several automatic measures of editing effort computed by TransCenter

α, β, γ, and δ can be tuned to maximize agreement between Meteor scores and human assessments of translation quality.

Meteor Parameter Optimization: Tuning a version of Meteor to approximate a given evaluation task requires a set of n MT outputs with reference translations plus a set of human-annotated numerical scores $Y = \langle y_1, ..., y_n \rangle$. Meteor scores the MT outputs, producing metric scores $X = \langle x_1, ..., x_n \rangle$. Ideally, X should be strongly correlated with Y, meaning that a high metric score should correspond to a high human score and vice versa. During tuning, sufficient statistics are calculated for each MT output, allowing it to be rapidly re-scored with various parameter settings. We then conduct an exhaustive parametric sweep over feasible parameter values to maximize correlation between X and Y over all MT outputs. Our work uses two different measures of correlation that offer different advantages depending on the task: Pearson's r and Spearman's ρ. The Pearson product-moment correlation coefficient r measures the *linear* correlation between two variables on a scale from 1 to -1 (Pearson, 1895). Spearman's rank correlation coefficient ρ assesses the extent to which two variables can be described using a monotonic function (Spearman, 1904). Removing the linearity constraint generally allows metrics to reach much higher correlation values while sacrificing some interpretability of absolute scores. This is useful in the case of system optimization where the goal is select a parameter set that yields the best possible translations.

Improved Editing Measures for Improved Metrics: The post-editing data collected by TransCenter allows us to explore a range of possible measures of human editing effort. We augment the data collected in our adaptive MT validation experiments with a second similar round of post-editing using a new set of translators from Kent State University's applied linguistics program. Combined, this data consists of 1000 post-edited sentences of TED talks translated from Spanish into English (Denkowski et al., 2014b). Included effort measures are traditional HTER, translator usability ratings, keystroke counts, editing times, and two pause measures: APR and PWR. To examine the relationship between these various measures, we compute correlation between all measures. Shown in Table 1.5, all measures tend to correlate with each other to some degree. Notably, HTER and keystroke have a correlation of 0.91, indicating that for this data, HTER is a very close approximation of actual editing effort. Further, a correlation of -0.84 between HTER and user rating indicates that translators are able to reliably assess the amount of editing they have just completed. Finally, keystroke count stands out as particularly indicative of overall effort; it is highly correlated with both HTER and user rating and has the highest correlation with editing time of any measure.

In a second experiment, we tune a version of Meteor to maximize correlation (Spearman's ρ) with each of these measures. Examining the optimal parameters provides insight into what types of translations require more or less effort to edit according to each measure. Shown in Table 1.6, we focus on the three most promising measures: HTER, Keystroke, and rating. One striking result is the focus on content words (δ) in rating parameters. This indicates that translators do not consider smoothing out grammatical errors to be

Task	α	β	γ	δ	w_{exact}	w_{stem}	w_{syn}	w_{par}
HTER	0.90	0.10	0.55	0.60	1.00	0.00	0.00	0.80
Keystroke	0.65	0.10	0.55	0.65	1.00	0.00	0.00	0.80
Rating	0.45	0.00	1.00	1.00	1.00	0.20	0.00	0.80

Table 1.6: Comparison of Meteor parameters for different measures of editing effort

	HTER	Rating	SPE BLEU
TED-BLEU	20.1	4.16	**27.3**
TED-Meteor	**18.9**	**4.24**	26.6

Table 1.7: Results for live post-editing experiments with task-specific metrics

nearly as significant as correcting mistranslations of important concepts. Also of note is that rating parameters favor precision (α), which is actually contrary to annotators' natural preference for recall observed in evaluations such as WMT (Callison-Burch et al., 2012). Finally, the HTER parameters are far more extreme than those learned from other post-editing data (Denkowski and Lavie, 2011): recall is preferred almost exclusively, the fragmentation penalty is harsher (γ), and stem and synonym matches are disregarded. The results of our metric experiments demonstrate two things. First, parameters are specific to the *data set* as well as the type of edit measure. Second, within a task, several different measure types correlate highly with one another. Together, these observations point to a revised role for automatic metrics in adaptive machine translation. Rather than developing a single "post-editing" version of a metric, we can use the post-editing data that naturally arises from using adaptive systems to tune *task-specific* metrics specifically for use with these systems.

Post-Editing Experiments with Task-Specific Metrics: We incorporate task-specific metrics into our adaptive MT systems as follows. First, we build and deploy an adaptive MT system. This requires no post-editing data, using simulated post-editing and the BLEU metric during optimization and internal evaluation. Once this system is put into production, serving translations to actual humans, post-edited data is naturally created. Once a sufficient amount of data is collected, it can be used to tune a custom version of Meteor that is specific to the MT system and the domain of the data being post-edited. This system is then re-tuned using this version of Meteor and re-deployed. As the system continues to translate and adapt, the task-specific version of Meteor is used as the optimization target; BLEU is entirely removed from the system.

We evaluate this approach with another set of post-editing experiments. We begin by selecting our task-specific metric: the version of Meteor tuned on keystroke data collected in our previous rounds of Spanish–English TED talk post-editing. This metric is then used to re-tune the fully adaptive system used in previous live post-editing experiments and act as the optimization target during decoding. The resulting Meteor-tuned system is evaluated against the BLEU-tuned baseline. To prevent the Meteor-tuned system from having an advantage simply from having access to more data, both systems are tuned on TED talk data. We work with another set of 5 translators from Kent State University and excerpts from 10 talks in the second TED data set (TED2) totaling 200 sentences. Following previous experiments, we collect live post-editing data with TransCenter and also report simulated post-editing BLEU scores. Shown in Table 1.7, the simulated BLEU score actually indicates a drop in performance when Meteor is used for optimization. However, data collected from TransCenter shows that Meteor-driven translations require less mechanical effort to correct and are preferred by post-editors.

Chapter 2

Background

2.1 The Mechanics of Phrase-Based Machine Translation

Faced with the task of translating a French sentence into English, a human translator has the ability to read the original sentence, call on knowledge of source language syntax and semantics to discern the meaning, and write out a grammatically correct sentence in the target language that conveys the same meaning. Years of reading and writing both languages allow the translator to be sensitive to nuances such as idioms, tone, and context when crafting a polished, meaning equivalent translation. Given the same task, a machine translation system with no inherent knowledge of human language cannot replicate this process. However, when provided with large amounts of bilingual text (millions of sentence pairs or more), the system can recognize words and phrases from the input sentence and recall how humans have translated these pieces in the past. Using a collection of statistical models, the system can predict the most likely human translation of the new sentence given what it has seen before. Recognizing larger pieces of an input sentence leads to better translation quality; in the best case, an entire sentence can be recognized and a human quality translation can be recalled while in the worst case, each word must be recalled from a different translation and pieced together. This puts a vital importance on the availability of data that is similar to what needs to be translated. Current statistical machine translation systems use a range of techniques to learn ideal units of translation from such data, match them against unseen source sentences, and piece their translations together in a way that seems reasonable in the target language. In this section, we build up to the widely used phrase-based approach to machine translation. For further reading on the motivation for and theory of statistical machine translation, see Kevin Knight's tutorial on earlier word-based translation (1999) and Chapter 1 of Adam Lopez's dissertation (2008a) that surveys more recent work.

2.1.1 Word Alignment

Early statistical translation models (Brown et al., 1993) are word-based, explaining translation as the following lexical process. For a given source language sentence $F = \langle f_1, ..., f_n \rangle$ with length n, generate a target sentence length m and set of alignment links $A = \langle a_1, ..., a_m \rangle$. Finally, generate a target sentence $E = \langle e_1, ..., e_m \rangle$ where each target word e_i only depends on the source word f_{a_i} that it is aligned to. Current alignment models handle differences in length between source and target sentences by allowing each f_i to generate any number of links, aligning to zero or more words e_i. Target words e_i can similarly align to zero or more words f_i. Differences in word order between source and target languages are accounted for by allowing alignment links to be unordered with respect to the source sentence. For example, Figure 2.1 shows an alignment where f_2 (de) is aligned to both e_1 and e_3, and f_1 (devis), is aligned to e_4.

While modern systems do not translate with word based models directly, the intermediate alignments between source and target words produced by these simple models are still useful as a starting point for

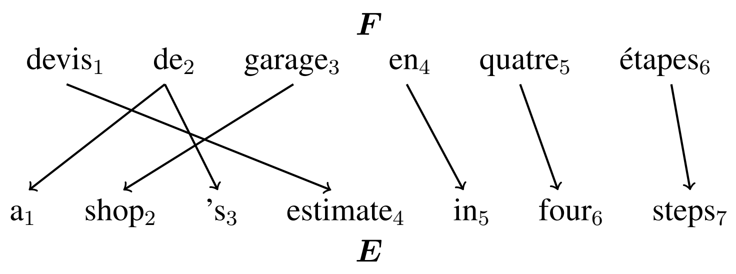

F
devis1 de2 garage3 en4 quatre5 étapes6
a1 shop2 's3 estimate4 in5 four6 steps7
E

Figure 2.1: Visualization of French-to-English word alignment with one-to-many alignments and reordering

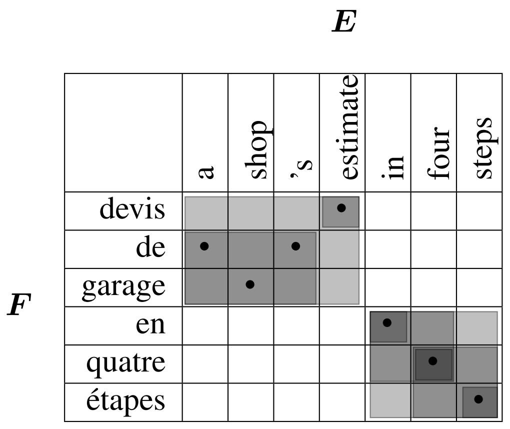


Figure 2.2: Visualization of phrase extraction from an aligned sentence pair. Dots signify alignment links and shaded boxes signify extracted phrases. Phrase length is limited to 3 words on the source side. A total of 9 phrases are extracted.

building more sophisticated models. One shortcoming of these alignments is that since they come from directional models, they are unable to map multiple source words to a single target word, or multiple source words to multiple target words. To account for this, models are typically run in both directions (source-to-target and target-to-source) and the alignments symmetrized (Och and Ney, 2003; Koehn et al., 2005). Symmetrization uses information from both alignments to produce a single, bidirectional alignment that supports one-to-many alignments in either direction.

2.1.2 Bilingual Phrase Extraction

Individually translating each word in a source sentence without context and permuting the result into something meaningful is clearly problematic. Once bilingual text has been aligned at the word level, phrase-based models (Koehn et al., 2003; Och and Ney, 2004; Och et al., 1999) can learn more reliable mappings between source and target languages by grouping together sequences of words into atomic units of translation. For example, rather than relying on the complex alignment process in Figure 2.1 to translate "devis de garage", a phrase-based model can simply learn that the whole phrase translates into "a shop's estimate". These mappings, termed *phrase pairs*, can be extracted automatically from word-aligned text. Given an aligned source-target sentence pair $\langle F, E, A \rangle$, phrase pairs consistent with the alignment can be identified as follows. A phrase pair $\langle f_i^{i+n}, e_j^{j+m} \rangle$ covering the contiguous span of words from i to $i + n$ in the source sentence

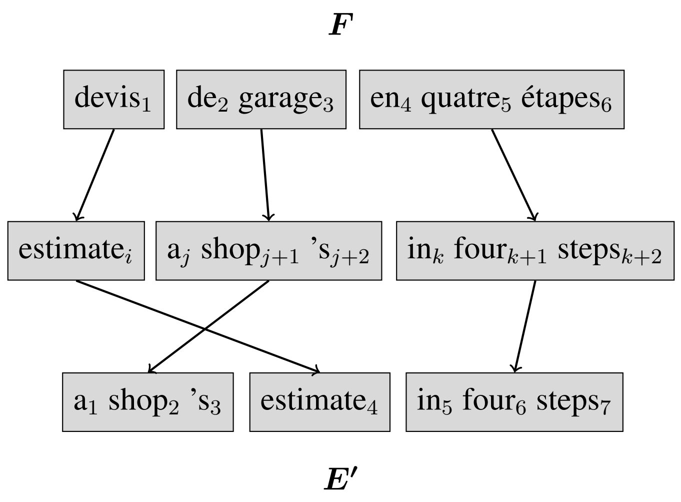

F
devis$_1$
de$_2$ garage$_3$
en$_4$ quatre$_5$ étapes$_6$
estimate$_i$
a$_j$ shop$_{j+1}$'s$_{j+2}$
in$_k$ four$_{k+1}$ steps$_{k+2}$
a$_1$ shop$_2$'s$_3$
estimate$_4$
in$_5$ four$_6$ steps$_7$
E'

Figure 2.3: Visualization of phrase-based segmentation, translation, and reordering with 3 phrase pairs

and from j to $j+m$ in the target sentence is extracted if (1) at least one word in f_i^{i+n} is aligned to a word in e_j^{j+m} and (2) no word in f_i^{i+n} is aligned to any word outside e_j^{j+m} and vice versa. Formally, the bilingual phrase pairs for a given sentence pair are defined:

$$\begin{aligned}\texttt{BPP}(F,E,A) = \Big\{ \langle f_i^{i+n}, e_j^{j+m} \rangle \;\Big|\; & \forall \langle i',j' \rangle \in A : i \le i' \le i+n \iff j \le j' \le j+m \\ & \wedge \exists \langle i',j' \rangle \in A : i \le i' \le i+n \wedge j \le j' \le j+m \Big\}\end{aligned} \tag{2.1}$$

An example of phrase extraction is visualized in Figure 2.2. Note that many overlapping phrases can be extracted from the same sentence pair.

Once phrases are learned, the task of translating a new source sentence F consists of decomposing it into a series of phrases, rewriting each phrase with its target language equivalent, and permuting the order of phrases on the target side to produce the final sentence E' (called a translation *hypothesis*). Translating at the phrase level has the key advantages of context and encapsulation. While individual words can have many translations, longer phrases are generally less ambiguous. While a word-based model cannot distinguish between the French preposition “en” and the English language code abbreviation “en”, a phrase-based model can match the longer phrase “en quatre étapes”, using additional context to resolve translation ambiguity. Other phrases such as “devis de garage” that would require complicated word mapping and reordering in word-based translation can be captured in a single phrase pair. Other complex natural language phenomena such as idiomatic phrases and morphological inflection can also be encapsulated in phrase pairs, allowing for a single phrase rewrite operation to generate a human quality translation for difficult content. However, this also underscores the importance of having seen at least one instance of a given language construction in the training text.

2.1.3 Phrase Reordering

To account for the permutation of translated phrases (the final step in Figure 2.3), the model also learns a set of *reordering patterns* for each phrase pair (Koehn et al., 2005). Patterns are based on the formalism that each target sentence (E in the training data or E' during translation) is generated by translating one source phrase at a time to build a translation from left to right on the target side. At any point in translation, any source phrase may be translated next using one of the following reorderings:

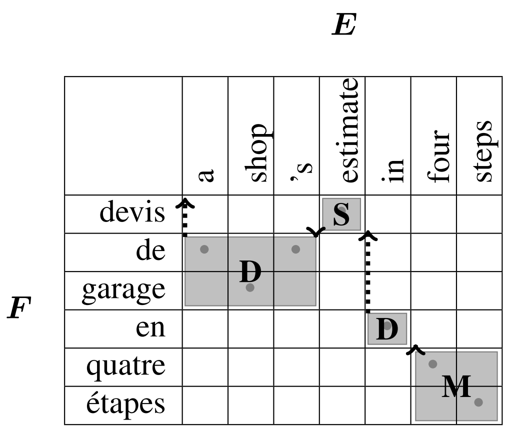


Figure 2.4: Visualization of learning monotone, swap, and discontinuous reordering patterns for four of the many phrases extracted from an aligned sentence pair. Arrows connect phrase boundaries to single aligned words. Alternatively, this can be seen as a visualization of translating the source sentence with four phrase pairs using the following reordering operations: discontinuous, swap, discontinuous, monotone.

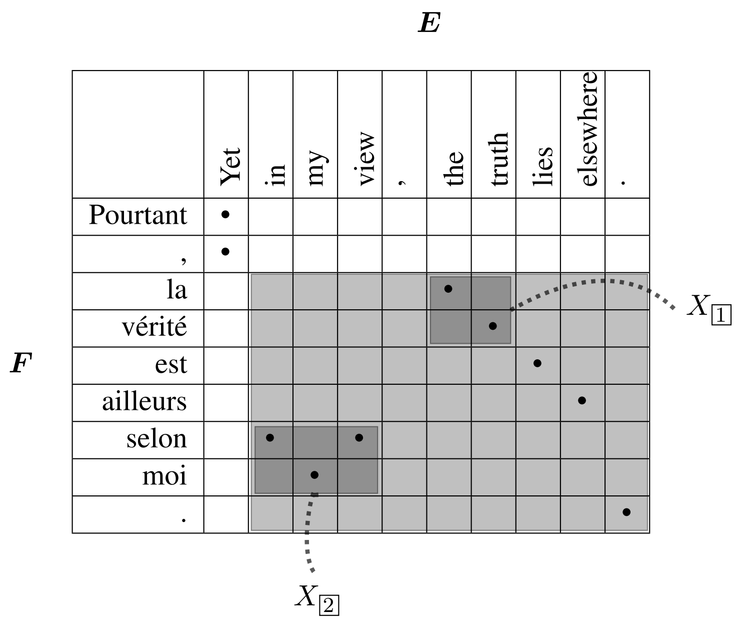
- Monotone: this source phrase occurs immediately after the previously translated phrase (translate in order).
- Swap: this source phrase occurs immediately before the previously translated phrase (translate in reverse order).
- Discontinuous: this source phrase is not adjacent to the previously translated phrase (jump to somewhere else in the sentence and start translating).

To learn possible reorderings, this translation process can be simulated on the aligned training text from which phrases are extracted. For each phrase pair, the model follows the alignment of the target word immediately preceding the phrase (the word that would have been most recently translated when generating a sentence left-to-right). Based on the position of the corresponding source word, one of the three patterns is identified:

- Monotone: the source word occurs immediately before the phrase.
- Swap: the source word occurs immediately after the phrase.
- Discontinuous: the source word is not adjacent to the phrase.

The pattern is then added to the list of possible reordering that can be applied to the phrase pair. An implied alignment before the first word of each sentence is used to anchor the first phrase.

Phrase-level reordering patterns allow the model to learn language-specific word order permutations. Figure 2.4 shows the reordering patterns learned from one possible phrase segmentation of an example sentence. Here the model learns that when translating from French to English, the monotone order of prepositions and nouns is preserved in prepositional phrases while the order of some words within noun phrases can be swapped.



$$X \longrightarrow X_{\boxed{1}} \text{ est ailleurs } X_{\boxed{2}} \text{ . } \big/ \; X_{\boxed{2}} \text{ , } X_{\boxed{1}} \text{ lies elsewhere .}$$

Figure 2.5: Visualization of a hierarchical phrase pair extracted from an aligned sentence pair. The linked non-terminals in the resulting SCFG rule encode reordering between the source and target.

2.1.4 Hierarchical Phrase-Based Translation

While phrase-based models excel at translating series of short, self-contained phrases, longer sentences pose significant challenges. Language phenomena such as long distance reordering and word agreement require context beyond what can be easily captured by short phrases and simple reordering patterns. To correctly translate long, complex sentences, phrase-based systems must make sequences of independent and unintuitive translation and reordering decisions reminiscent of word-based models. As an alternative to using a simple reordering model, the *hierarchical* phrase-based formalism (Chiang, 2007) enables phrases to contain other phrases, allowing long distance context to be encapsulated in the same way as local context. With these generalized phrases, larger portions of text can be grouped together and reordered within the context of a single phrase pair. Given an aligned source-target sentence pair $\langle F, E, A \rangle$, hierarchical phrase pairs can be extracted as follows. First, identify initial phrase pairs that meet the criteria in Equation 2.1. Next, identify phrases that contain other phrases and replace the source and target words covered by each sub-phrase with a special indexed symbol $X_{\boxed{i}}$. These symbols indicate where other phrase pairs can be plugged in. To keep the number of extracted rules manageable, the following additional constraints are imposed: (1) phrases and sub-phrases must be *tight*, meaning that boundary words must be aligned, (2) there must be at least one word between any two $X_{\boxed{i}}$ symbols in the source phrase, (3) there must be at least one word in the source phrase, and (4) there may be at most two $X_{\boxed{i}}$ symbols in any phrase. An instance of hierarchical phrase extraction is visualized in Figure 2.5. Note that this is just one of many hierarchical phrase pairs could be extracted from the example sentence pair.

Under this model, phrase pairs can also be expressed as rules in a synchronous context-free grammar (SCFG) where all source and target phrases are given the same label X. Formally, any translation *rule*

$\boldsymbol{F}$: Pourtant , la vérité est ailleurs selon moi .

$\boldsymbol{G}$:

$S \longrightarrow S_{\boxed{1}}\ X_{\boxed{2}}\ \big/\ S_{\boxed{1}}\ X_{\boxed{2}}$

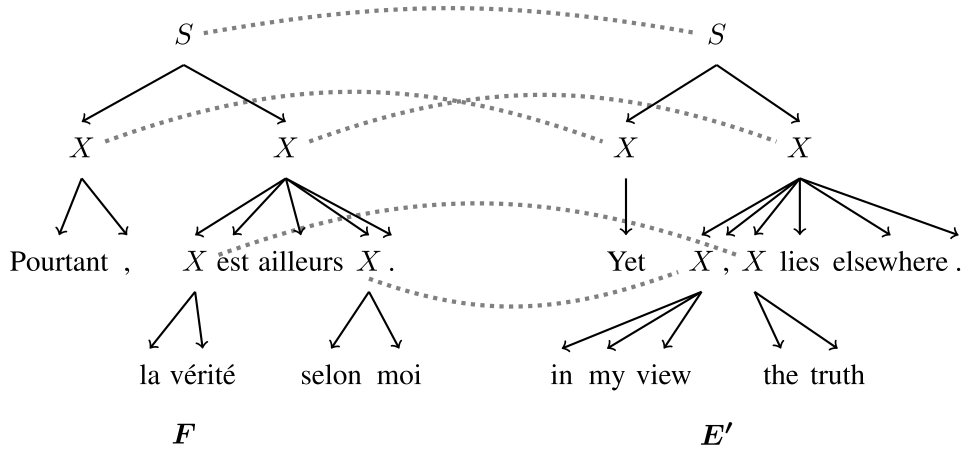
$S \longrightarrow X_{\boxed{1}}\ \big/\ X_{\boxed{2}}$

$X \longrightarrow X_{\boxed{1}}$ est ailleurs $X_{\boxed{2}}$. $\big/$ $X_{\boxed{2}}$, $X_{\boxed{1}}$ lies elsewhere .

$X \longrightarrow$ Pourtant , $\big/$ Yet

$X \longrightarrow$ la vérité $\big/$ the truth

$X \longrightarrow$ selon moi $\big/$ in my view



$\boldsymbol{S}_{\boxed{1}}\ \big/\ \boldsymbol{S}_{\boxed{1}}$

$\Longrightarrow\ \boldsymbol{S}_{\boxed{2}}\ \boldsymbol{X}_{\boxed{3}}\ \big/\ \boldsymbol{S}_{\boxed{2}}\ \boldsymbol{X}_{\boxed{3}}$

$\Longrightarrow\ \boldsymbol{X}_{\boxed{4}}\ X_{\boxed{3}}\ \big/\ \boldsymbol{X}_{\boxed{4}}\ X_{\boxed{3}}$

$\Longrightarrow$ **Pourtant ,** $X_{\boxed{3}}$ $\big/$ **Yet** $X_{\boxed{3}}$

$\Longrightarrow$ Pourtant , $\boldsymbol{X}_{\boxed{5}}$ **est ailleurs** $\boldsymbol{X}_{\boxed{6}}$ **.** $\big/$ Yet $\boldsymbol{X}_{\boxed{6}}$ **,** $\boldsymbol{X}_{\boxed{5}}$ **lies elsewhere .**

$\Longrightarrow$ Pourtant , **la vérité** est ailleurs $X_{\boxed{6}}$. $\big/$ Yet $X_{\boxed{6}}$, **the truth** lies elsewhere .

$\Longrightarrow$ Pourtant , la vérité est ailleurs **selon moi** . $\big/$ Yet **in my view** , the truth lies elsewhere .

Figure 2.6: Example of translation as parsing with a synchronous context-free grammar. Top: F is a sample source language sentence and G is a sample translation grammar. Center: visualization of the trees generated from parsing F with G, also building a target hypothesis E'. Dotted lines indicate that non-terminals share indices. Bottom: synchronous derivation of F and E' under G.

extracted from data can be written:

$$X \longrightarrow \bar{f} \;/\; \bar{e} \tag{2.2}$$

Here $\bar{f}$ denotes a source-language phrase and $\bar{e}$ denotes a target-language phrase. Phrases *must* contain terminals (words) and *may* contain linked non-terminals $X_{\boxed{i}}$ (see example rule in Figure 2.5). The task of translation is now equivalent to parsing the source sentence with the translation grammar, simultaneously building up a target-language derivation and ultimate translation. To allow building full derivations, a single goal non-terminal S is added to the translation grammar. To maintain the benefits of the phrase-based approach (dividing input sentences into individually-translatable chunks), two *glue rules* are added that string together series of phrases:

$$\begin{aligned} S &\longrightarrow S_{\boxed{1}}\, X_{\boxed{2}} \;/\; S_{\boxed{1}}\, X_{\boxed{2}} \\ S &\longrightarrow X_{\boxed{1}} \;/\; X_{\boxed{2}} \end{aligned} \tag{2.3}$$

Figure 2.6 shows an example of translation as parsing with a synchronous context-free grammar.

In addition to providing a powerful generalization of the phrase-based formalism, the hierarchical approach can be considered an unsupervised version of syntactic machine translation. In syntactic translation, models learn correspondences between source and target language *structure* from bilingual text that has been annotated with parse trees (sometimes called concrete syntax trees) on the source (Yamada and Knight, 2001; Liu et al., 2006), the target (Galley et al., 2004), or both (Lavie et al., 2008; Liu et al., 2009) sides. This additional information allows syntactic models to learn SCFG rules by dividing source and target parse trees into corresponding chunks based on word-level alignments, then translate new sentences by parsing them with the resulting translation grammar. While the hierarchical model does not have access to parse information, it can learn similar translation rules based solely on word alignments. The end result is a model that incorporates strengths of syntactic approaches while retaining the flexibility of phrase-based translation.

In practice, the hierarchical model tends to outperform the simple phrase-based model on longer distance reordering while sacrificing some reliability on short distance reordering. While the hierarchical model possesses more powerful reordering capabilities, it must back off to using series of glue rules when the input text does not match any hierarchical rules. In this case, it effectively operates as a phrase-based model without the benefit of a reordering model. Depending on the reordering demands of a given language pair, one model or the other may be preferable.

2.1.5 Generalized Phrase-Based Translation

Much of the material covered in the following sections can be applied to the more general class of statistical translation models that includes both phrase-based and hierarchical formalisms. Unless otherwise specified, model components and algorithms described are common to both approaches (traditional phrase-based with a reordering model or hierarchical phrase-based without a reordering model). As such, we will use the following general terms when describing these models:

- A **translation rule** ($X \rightarrow \bar{f}/\bar{e}$) refers to either a phrase pair (terminals only) under the phrase-based formalism or a SCFG rule (terminals and possible non-terminals) under the hierarchical formalism.

- A **translation grammar** refers to a list of translation rules, phrase pairs under the phrase-based formalism or SCFG rules under the hierarchical formalism.

Both models use the translation grammar to build a derivation D that maps the source sentence to a newly generated target sentence. In phrase-based translation, D is a list of X rules that map source phrases to target phrases and the reordering patterns applied to those phrases on the target side (Figure 2.3). In hierarchical translation, D is a standard SCFG derivation (Figure 2.6).

2.2 Translation Model Parameterization

In the previous section, we introduced phrase-based translation models and described the process for extracting rules from bilingual text and using them to translate unseen sentences. The performance of these models is highly dependent on the amount of training data available, with models for high-traffic language pairs such as Spanish–English and Arabic–English typically being learned from millions of bilingual sentence pairs. Given the natural ambiguity of human language and the need to piece together translations from such large numbers of sources to translate new content, current machine translation systems employ several statistical models to predict the single *most likely* translation of a source sentence given all data the system has seen previously. This amounts to scoring and ranking the often exponential number of possible translation candidates for a single sentence. This process is referred to as *decoding* and the programs that conduct this process *decoders*. In this section, we describe decoding (inference) and the process of estimating the prerequisite translation models from bilingual text (learning).

2.2.1 Linear Translation Models

The translation models described in §2.1 can predict exponentially many translations for each source sentence. The model needs a way to score each possible translation so that it can select the single most likely candidate. The dominant approach, which we use throughout our work, is a translation model parametrization by Och and Ney (2002; 2003). A translation *hypothesis* consists of F, the input source-language sentence, D, the derivation (collection of rules) that maps F to some target-language sentence E', and E' itself, the translation we are ultimately interested in. We then introduce arbitrary *feature functions* $h_i \in H$ into our model that assign real-number values to hypotheses. Further described in the following sections, these functions typically measure how reliably F translates into E' or how well-formed of a sentence E' is in the target language. For each h_i, a corresponding *weight* $w_i \in W$ controls the relative contribution of the feature to the final score, allowing the model to trust some features more than others. Setting these weights (collectively called a weight vector) to maximize system performance is discussed in §2.3. By calculating the inner product of feature scores and weights for a given translation, one obtains a final score that can be compared against scores of other translations. Formally, the score of a translation hypothesis $\langle F, E', D\rangle$ can be written:

$$S(F, E', D) = \sum_{i=1}^{|H|} w_i h_i(F, E', D) \tag{2.4}$$

This leads to the translation decision rule to select $\hat{E}'$, the target language sentence with the highest score under the model:

$$\hat{E}'(F) = \arg\max_{\langle E', D\rangle} \sum_{i=1}^{|H|} w_i h_i(F, E', D) \tag{2.5}$$

The decision rule, used directly by our model, is a straightforward formulation of hypothesis score as the inner product of a feature score vector and a and feature weight vector. As observed by Clark (2015), when a translation model uses only this decision rule with arbitrary feature functions and weights, the model is linear rather than log-linear. The model is highly extensible and facilitates learning weights so as to directly maximize translation quality on held-out data.

2.2.2 Rule-Local Features

To score translation hypotheses, we add real-valued *local* features h_i to each rule in the translation grammar, making it a weighted grammar. The *global* value of each feature function (used in Equations 2.4 and 2.5) is the sum of the local features used in the derivation:

$$h_i(D) = \sum_{X \to \bar{f}/\bar{e} \in D} h_i\left(X \to \bar{f}/\bar{e}\right) \tag{2.6}$$

By assuming rules have the same feature values independent of context, efficient inference is possible with dynamic programming. While the scores $h_i(X \to \bar{f}/\bar{e})$ assigned to each rule can be arbitrary, they generally reflect how consistent a translation rule is with bilingual training data or provide other information about the current derivation during model search.

Phrase Features: Given a rule $X \to \bar{f}/\bar{e}$, these features encode the empirical relative frequency of a given source phrase $\bar{f}$ being translated as a target phrase $\bar{e}$ according to the bilingual training data. Here the training data is the set of all rule instances extracted from all sentences in the training text. Counting rules that share source, target, and both sides leads to the following statistics:

- $C(\bar{f}, \bar{e})$: the count of times the rule with source $\bar{f}$ and target $\bar{e}$ is extracted.
- $C(\bar{f})$: the count of times any rule with source $\bar{f}$ is extracted with any target.
- $C(\bar{e})$: the count of times any rule with target $\bar{e}$ is extracted with any source.

Given these statistics, two standard translation probabilities are calculated:

$$P(\bar{e}|\bar{f}) = \frac{C(\bar{f}, \bar{e})}{C(\bar{f})} \qquad P(\bar{f}|\bar{e}) = \frac{C(\bar{f}, \bar{e})}{C(\bar{e})} \tag{2.7}$$

Although feature scores are used in the context a linear model rather than a log-linear model, the log transformation typically yields better performance. As such, the log-transformed versions of these probabilities are used as features in the model:

$$\texttt{P(e|f)} = \log P(\bar{e}|\bar{f}) \qquad \texttt{P(f|e)} = \log P(\bar{f}|\bar{e}) \tag{2.8}$$

Lexical Features: Since individual words generally occur far more frequently than whole phrases, word-level translation scores can be effectively estimated from much larger data, leading to more reliable estimates. Adding these *lexical* scores to the linear translation model can be seen as smoothing the less stable phrase-based translation scores with word-based translation scores. Given a rule $X \to \bar{f}/\bar{e}$, lexical features encode the probability of the words $e \in \bar{e}$ being *individually* mapped to the words $f \in \bar{f}$. Here the training data is the set of all alignment links from all word alignments in the bilingual training text. Counting instances of aligned words leads to the following statistics:

- $C(f, e)$: the count of times source word f is aligned to target word e. When counting links, one-to-many alignments are accounted for by adding a fractional count of $\frac{1}{n}$ for any instance where f or e is aligned to n words instead of one.
- $C(f)$: the count of times source word f is aligned to one or more target words.
- $C(e)$: the count of times target word e is aligned to one or more source words.

Word-level lexical probabilities are calculated:

$$P(e|f) = \frac{C(f, e)}{C(f)} \qquad P(f|e) = \frac{C(f, e)}{C(e)} \tag{2.9}$$

These scores are then used to calculate phrase-level lexical scores. When aligning words within phrases, we use an approximation wherein each source word $f \in \bar{f}$ is aligned to the target word $e \in \bar{e}$ with the highest word-level score. Formally, the two lexical scores are calculated:

$$\text{lex}(\bar{e}|\bar{f}) = \prod_{e \in \bar{e}} \arg\max_{f \in \bar{f}} P(e|f) \qquad \text{lex}(\bar{f}|\bar{e}) = \prod_{f \in \bar{f}} \arg\max_{e \in \bar{e}} P(f|e) \tag{2.10}$$

As with phrase translation probabilities, log-transformed versions of these scores are used in the model:

$$\texttt{Lex(e|f)} = \log \text{lex}(\bar{e}|\bar{f}) \qquad \texttt{Lex(f|e)} = \log \text{lex}(\bar{f}|\bar{e}) \tag{2.11}$$

2.2.3 Reordering Features (Phrase-Based Model)

When decoding with a standard phrase-based model, the derivation consists of the list of rules $X \rightarrow \bar{f}/\bar{e}$ applied, the order they are applied in, and the spans of source text they translate. This allows each applied rule to be scored by a reordering model to determine how consistent the reordering operations used by the decoder are with those observed in training text. The model encodes the relative frequency of each reordering pattern (described in §2.1.3) for each phrase pair extracted from the aligned training text. These features measure the probability of the extracted phrase $\langle \bar{f}, \bar{e} \rangle$ being monotone (M), swap (S), or discontinuous (D) with respect to the previous phrase[1]:

$$\texttt{M(f,e)} = P(M|\bar{f}, \bar{e}) \qquad \texttt{S(f,e)} = P(S|\bar{f}, \bar{e}) \qquad \texttt{D(f,e)} = P(D|\bar{f}, \bar{e}) \tag{2.12}$$

These scores can also be computed in the opposite direction, considering the next used phrase rather than the previous. In this case, the same patterns are used, but the next translated word is considered rather than the previous. Additionally, a single distance-based reordering feature is used to count the total reordering distance $|D|$ for each discontinuous phrase:

$$\texttt{Dist(f,e)} = \begin{cases} |D| & \langle \bar{f}, \bar{e} \rangle \text{ is discontinuous} \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

Phrase-based models typically employ both forward and backward reordering probabilities as well as the simple distance-based score, totalling 7 reordering features.

2.2.4 SCFG Features (Hierarchical Model)

When decoding with a hierarchical model, the derivation D is the list of SCFG rules $X \rightarrow \bar{f}/\bar{e}$ applied. Whereas the phrase-based model uses reordering features to reward likely phrase orderings, this model uses a set of rule features to reward likely bilingual derivations. The following *indicator* features are used to score each rule in D, returning 1 if the rule meets the criteria, otherwise 0:

- `Arity0(f,e)`: this rule contains zero non-terminals $X_{\boxed{i}}$.
- `Arity1(f,e)`: this rule contains one non-terminal $X_{\boxed{i}}$.
- `Arity2(f,e)`: this rule contains two non-terminals $X_{\boxed{i}}$.
- `Glue(f,e)`: this rule is a glue rule. (Equation 2.3).

This allows the decoder to, for example, prefer translating phrases with rules that encode reordering patterns rather than translating phrases separately and chaining them together with glue rules.

[1]Here the use of M, S, and D is specific to this section, not to be confused with the start symbol S or derivation D used in other sections.

2.2.5 Monolingual Features

Language Model Features: In addition to feature scores assigned to SCFG rules that encode the likelihood of source phrases translating into target phrases, a machine translation systems employs a *language model* that assigns scores to the target language sentence $E' = \langle e_1 ... e_{|E'|} \rangle$. Language model scores reflect $P(E')$, the likelihood of sentence E' occurring given the monolingual training text. Linguistically, language models can be seen as a measure of grammaticality and fluency, how well formed the translation hypothesis is in the target language. Standard language models use an N-gram approximation where the probability of a word e_i is conditioned on the previous N-1 words, typically 3 or 4. Since the model matches E against the training data, this approximation greatly reduces sparsity; entire sentences that the translation model generates are unlikely to appear in training data, but short sequences of words are more likely. Formally, the probability under an N-gram language model and the corresponding log-transformed feature in our system are given:

$$P_N(E') = \prod_{i=1}^{|E'|} P_N(e_i | e_1^{i-1}) = \prod_{i=1}^{|E'|} P_N(e_i | e_{i-N}^{i-1}) \qquad \texttt{LM(E)} = \log P_N(E') \tag{2.14}$$

N-gram probabilities for language models use smoothed maximum likelihood estimates on the training data, which consists of all N-gram instances that occur in the monolingual text. When an N-gram is not found, rather than assigning zero probability, the model can "back off" to the probability for a shorter context, starting with N-2, down to 0. If the current word is not in the vocabulary of the model, a single probability for out-of-vocabulary (OOV) words is applied. To fine-tune the impact of OOV words, an additional count-based feature is added to track the number of OOV words in E':

$$\texttt{OOV(E)} = \sum_{e \in E'} \begin{cases} 0 & e \text{ in language model} \\ 1 & \text{otherwise} \end{cases} \tag{2.15}$$

As monolingual data is generally far more plentiful than bilingual data, language models are estimated from orders of magnitude more data than translation models, making language model scores powerful discriminative features. It is important to note that translation (phrase and lexical) features and language model features operate independently within the linear model, each weighing in on the quality of a translation hypothesis. Hypotheses with higher $P(E'|F)$ frequently have higher $P(E')$ but this is not always the case, especially when translating out-of-domain text where bilingual and monolingual training data may be mismatched.

Source Out-of-Vocabulary Feature: While the language model counts the number of OOVs with respect to the monolingual (language model training) text, a *pass through* feature counts the number of OOVs with respect to the bilingual (translation grammar training) text. As the model cannot translate these words, they must be "passed through" verbatim to the translation hypothesis. It is possible that some OOVs, such as proper names, will be in the language model, so this count is maintained as a separate feature

$$\texttt{PassThrough(F)} = \sum_{f \in F} \begin{cases} 0 & f \text{ can be translated} \\ 1 & \text{otherwise} \end{cases} \tag{2.16}$$

Word Count Feature: This feature is simply the number of words in the translation hypothesis E'. It directly counterbalances the bias toward shorter sentences favored by the language model. Multiplying language model probabilities for additional words continues to lower the final score, leading the language model to naturally prefer shorter sentences even when the individual N-grams receive low scores. A positively weighted word count feature can reward longer translations, leading to a better balance between translation quality and length.

$$\texttt{WordCount(E)} = |E'| \tag{2.17}$$

2.2.6 On-Demand Grammar Extraction with Suffix Arrays

In phrase-based translation, grammar estimation is conducted in two stages. First, rule instances are extracted from the aligned training text. Second, the rule instances are combined and scored to produce a single grammar that can translate any phrase extracted from the training text. However, this grammar discards any knowledge of the documents or sentences in which various phrases occurred in the training text, relying only on rule-local context for translation. An alternative approach, termed *example-based* machine translation, translates input sentences by matching individual phrases against occurrences in the training text to recall how they were translated in similar context (Veale and Way, 1997; Brown, 1999; Carl, 1999; Cicekli and Güvenir, 2000, *inter-alia*). One important requirement for EBMT is an efficient index of the source text, such as a suffix array, that allows fast phrase lookups at runtime (Brown, 2004).

In an approach that adds the training text awareness of EBMT to phrase-based translation, Callison-Burch et al. (2005) and Lopez (2008a; 2008b) introduce an *on-demand* phrase-based model. Rather than building a single large translation model, the aligned training text is indexed with a suffix array as in EBMT (Manber and Myers, 1993). When an input sentence needs to be translated, a grammar extraction program finds all possible decompositions of the source sentence into phrases and searches for these phrases in the training text. This results in a list of all instances of each source phrase along with the target-language phrases the source phrase is aligned to. This data can be used to score a sentence-level grammar that translates the input sentence. Instead of using all occurrences of each source phrase in the data (potentially millions for common phrases), a smaller *sample* can be used to estimate feature scores. Lopez (2008a) finds that a sample size of 100 performs comparably to using the entire data. This allows for both the rapid generation of grammars on an as-needed basis and the inclusion of a powerful suffix array-backed feature set, discussed below.

2.2.7 Suffix Array Phrase Features

A source-side suffix array provides valuable information that can be used to estimate a more powerful set of phrase features. For each source phrase $\bar{f}$, the suffix array returns a sample $\mathcal{S}$ that consists of instances $\langle \bar{f}, \bar{e}' \rangle$ where $\bar{e}'$ is the target-language phrase that $\bar{f}$ is aligned to in the given instance. In the case that $\bar{f}$ is unaligned, $\bar{e}'$ is empty and no rule can be instantiated. A single $\mathcal{S}$ is used to score all rules $X \rightarrow \bar{f}/\bar{e}'$ that can be instantiated over $\bar{f}$ and $\bar{e}'$. Feature scores are calculated using the following statistics:

- $C_{\mathcal{S}}(\bar{f}, \bar{e})$: the count of instances in $\mathcal{S}$ where $\bar{f}$ is aligned to $\bar{e}$. Also called the co-occurrence count, the log-transformed version of this statistic can be used directly as a feature score:

$$\texttt{Count(f,e)} = \log C_{\mathcal{S}}(\bar{f}, \bar{e}) \tag{2.18}$$

- $C_{\mathcal{S}}(\bar{f})$: the count of instances in $\mathcal{S}$ where $\bar{f}$ is aligned to any target phrase.

- $|\mathcal{S}|$: the total number of instances in $\mathcal{S}$, equal to the number of occurrences of $\bar{f}$ in the training data, up to the sample size limit. The log-transformed version of this statistic can be used directly as a feature score:

$$\texttt{SampleCount(f)} = \log |\mathcal{S}| \tag{2.19}$$

In addition to the above statistics, a suffix array makes possible two *singleton indicator* features that return a value of one if their conditions are met, otherwise zero. These features are used to count rare translations where only one instance of $\bar{f}$ or $\langle \bar{f}, \bar{e} \rangle$ exist in the training data:

$$\texttt{Singleton(f)} = \begin{cases} 1 & C_{\mathcal{S}}(\bar{f}) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \texttt{Singleton(f,e)} = \begin{cases} 1 & C_{\mathcal{S}}(\bar{f}, \bar{e}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

Finally, the suffix array index of all instances of $\bar{f}$ in the source side of the training data allows for the calculation of a more powerful phrase translation score. Termed the *coherent* translation score, this variant conditions on the frequency of $\bar{f}$ in the data rather than frequency of $\bar{f}$ being extracted as part of a phrase pair. The formula and subsequent log-transformed feature function are given:

$$\text{coherent } P(\bar{e}|\bar{f}) = \frac{C_{\mathcal{S}}(\bar{f}, \bar{e})}{|\mathcal{S}|} \qquad \texttt{CoherentP(e|f)} = \log \text{coherent } P(\bar{e}|\bar{f}) \tag{2.21}$$

The use of $|\mathcal{S}|$ instead of $C_{\mathcal{S}}(\bar{f})$ increases the value of the denominator in cases where the source phrase is frequently unaligned, preventing rule extraction. This reduces the feature value of rules for which the source side tends not to align well.

2.3 Translation System Optimization

One of the key advantages of the linear translation model discussed in §2.2.1 is the ability to add any real-valued feature function and assign a weight to control its contribution to scoring hypotheses. This theoretically allows any knowledge source to be adapted to provide information to a translation model, relying on the optimizer to decide how useful it is. Learning ideal weights for these features is an important problem. Translation system optimization is generally formulated as choosing the weight vector that, when used with a fixed set of features, is most likely to result in good translations of future text. This is complicated by (1) the lack of a clear definition of translation goodness, (2) large search spaces, and (3) the difficulty many algorithms have with finding optimal weights for large numbers of often correlated feature functions. To address the first point, automatic evaluation metrics are introduced that compare translation hypotheses against pre-generated human translations and generate a similarity score (frequently called distance) in a well-defined way. To address the second, newer optimization algorithms use different objective functions intended to scale to much larger feature sets. In this section, we describe two popular translation system optimization techniques and the metrics they use to score translations.

2.3.1 Batch Learning: Minimum Error Rate Training

One widely used method for learning feature weights is minimum error rate training (MERT) (Och, 2003), which directly optimizes system performance on a given development data set (also called a tuning set). The intuition here is that the system parameters that lead to the best translations for known data are likely to lead to good translations for unknown data. Translation quality is measured by an automatic metric G that returns a similarity score between a system's output E' and a human reference translation E at the corpus level. MERT searches for the weight vector $\hat{W}$ that leads the translation model to prefer the E' closest to E, maximizing the metric score. Given a bilingual development corpus C that consists of sentence pairs $\langle F, E \rangle$, MERT's optimization function is given:

$$\hat{W} = \arg\max_{W} \sum_{\langle F,E \rangle \in C} G\left(\hat{E}'(F), E\right) \tag{2.22}$$

Using Equation 2.5, we expand the translation model's decision rule $\hat{E}'(F)$ to show the direct impact of feature weights on translation selection and consequently score:

$$\hat{W} = \arg\max_{W} \sum_{\langle F,E \rangle \in C} G \left(\arg\max_{\langle E',D \rangle} \sum_{i=1}^{|H|} w_i h_i(F, E', D), E \right) \tag{2.23}$$

The search for $\hat{W}$ proceeds as an iterative line search. First, the translation system uses an initial set of weights (uniform, random, or set based on some amount of prior knowledge) to translate the source sentences in C. Rather than producing the single most likely translation under the model, the system outputs the K most likely translations, either as a list or packed into a lattice or hypergraph (Macherey et al., 2008). Candidates in the K-best list are annotated with feature scores, allowing them to be re-scored with different weight vectors. A weight vector W is scored by calculating the metric score for the single candidate in each K-best list preferred by the model under W and aggregating the results. MERT then conducts a sequence of individual searches, finding the best-scoring value for each weight $w \in W$ while holding the values of all other weights fixed. Once MERT has optimized the entire weight set, the system generates a set of K-best lists with the new weight vector and aggregates them to lists from previous iterations. This allows the line search to view increasingly large portions of the space of possible translations under the model. In addition to searching from the best W in the previous iteration, MERT also searches several random weight vectors in each iteration to reduce the chance of getting stuck in a local optimum. MERT concludes when no new translations are discovered under the current weight vector W, selecting the final vector $\hat{W}$ that results in the best known score given the visible space of translations.

While MERT is still the de facto optimization algorithm for translation systems, its design leads to a few natural drawbacks. Only looking at the metric score of the top-best translation for each sentence in a development set can lead MERT to prefer local optima that do not generalize well to other data sets; the optimizer may select weights that are overly specific to C at the expensive of performance on other data, *overfitting* the tuning set. Additionally, the line search method does not scale well to larger feature sets, especially when features are correlated; searching one parameter at a time can miss complex interactions between features. For sets of more than a few dozen features, line search actually becomes intractible.

2.3.2 Online Learning: Margin Infused Relaxed Algorithm

Batch learning proceeds as a series of iterations wherein the entire development corpus is translated and parameters are selected to maximize a corpus-level objective function. Alternatively, *online* learning involves processing one training example at a time, that is translating a single sentence and making a slight parameter update to prefer better-scoring translation hypotheses for that sentence. Beginning with a uniform weight vector (all zeroes) and using an online learning algorithm to make several *passes* over the development corpus, learning from one sentence at a time, a stable weight vector can be learned.

The most widely used online learning algorithm for optimizing translation systems is the margin infused relaxed algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2006b; Watanabe et al., 2007; Chiang et al., 2008). Beginning with an input weight vector (typically uniform), MIRA makes a number of passes over the development corpus processing one sentence at a time. For each sentence, the decoder produces a list of the K best translation hypotheses E', which are then scored against a reference translation E using an automatic metric. From this list, two hypotheses are extracted. E^+, termed "hope", is a hypothesis with high model score and high metric score. E^-, termed "fear", is a hypothesis with high model score but low metric score. The $\langle E^+, E^- \rangle$ pair represents two data points that the model should (but doesn't currently) assign substantially different scores to. MIRA updates the weight vector so that the model prefers E^+ over the E^- by a *margin* proportional to their metric score difference. The algorithm then proceeds to the next sentence. By processing many hope-fear pairs, MIRA gradually transforms the initial weight vector into one that reliably prefers hypotheses with high metric scores over hypotheses with low metric scores.

Formally, a single MIRA update generates the next weight vector $w' \in W'$ using the current weight vector $w \in W$, current source sentence F, its hope and fear hypotheses $\langle E^+, E^- \rangle$, and their derivations $\langle D^+, D^- \rangle$:

$$\hat{W'} = \arg\min_{W'} \frac{1}{2} ||W' - W||^2 + C\xi_i \tag{2.24}$$

subject to

$$\sum_{i=1}^{|H|} w'_i h_i(F, E^+, D^+) - \sum_{i=1}^{|H|} w'_i h_i(F, E^-, D^-) \geq \text{cost}(E^+, E^-) - \xi_i \tag{2.25}$$

Here cost is determined with respect to difference in automatic metric score, ξ_i is a *slack* variable added to ensure E^+ and E^- are separable, and C is a constant *regularization parameter* that controls how far weights can be updated in a single step.

The above formulation of MIRA overcomes two significant weaknesses of MERT. First, it is capable of handling feature sets that are very large (up to hundreds of thousands of features) and contain many highly-correlated features. Second, it is significantly more stable, not relying on random search restarts to avoid local optima. Perhaps the greatest advantage of MIRA is its ability to update weights on a sentence-by-sentence basis, facilitating the rapid translation model adaptation that is growing in popularity as more users interact directly with MT systems in real time.

2.3.3 Evaluation Metrics

Ideally, a translation's quality (human or automatic) should be measured by its usefulness to humans, either for information assimilation or as a starting point for post-editing. In practice, human evaluation is often infeasible as evaluations need to be carried out rapidly and repeatedly. Automatic evaluation metrics simulate human judgments by comparing a translation hypothesis against a pre-existing reference translation and return a numerical similarity score, generally between zero and one. The role of metrics in system development is twofold. First, metrics are used to provide the thousands of individual evaluations required in algorithms like MERT and MIRA. Second, metrics are used to score the output of optimized systems on held out data to determine which system or system configuration performs best. This section describes three metrics frequently used for optimization and evaluation: BLEU, TER, and Meteor.

BLEU: Based on the idea that good translations should contain words and phrases from references, the bilingual evaluation understudy (BLEU) metric (Papineni et al., 2002) scores hypotheses according to surface form N-gram precision. For every n-gram length up to N, ($N = 4$ in the widely used BLEU$_4$ variant), an individual precision score $\mathcal{P}_n$ is calculated as the percentage of n-grams in the hypothesis also found in the reference. Precision scores are combined using a geometric mean and scaled by a brevity penalty intended to down-weight hypotheses that achieve good precision but are too short to achieve good recall. The penalty ($\mathcal{B}$) is based on the length of the translation hypothesis E' and reference E. The formula for BLEU score is given:

$$\mathcal{B}(E', E) = \begin{cases} 1 & \text{if } |E'| > |E| \\ e^{\frac{1-|E|}{|E'|}} & \text{if } |E'| \leq |E| \end{cases} \qquad \texttt{BLEU}_N(E', E) = \mathcal{B} \times \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log \mathcal{P}_n\right) \tag{2.26}$$

To account for translation variation, multiple reference translations can be used to score a single hypothesis. N-grams from the hypothesis can match any reference, but the same N-gram cannot be matched more times than it occurs in any one reference. The reference length closest to the hypothesis length is used to calculate the brevity penalty $\mathcal{B}$. BLEU scores range from 0 to 1 and are often reported as percentages (e.g., 27.3 BLEU "points" for a score of 0.273).

While BLEU is still the dominant metric for optimization and evaluation, it is frequently criticized for being insensitive to important translation differences. Callison-Burch et al. (2006) show that improvement in BLEU score is neither necessary nor sufficient for improvement in translation quality as assessed by humans. BLEU ignores linguistic phenomena such as synonymy and paraphrasing, penalizing translations that use different vocabulary and phrasing to express the same meaning as reference translations. As N-grams can be matched anywhere in a hypothesis, BLEU is also insensitive to non-local ordering, unable to discriminate between globally coherent sentences and scrambled sentences. The end result is that many diverse sentences in K-best lists appear identical to BLEU, leading to unreliable feedback for optimization algorithms such as MERT.

TER: The translation edit rate (TER) metric (Snover et al., 2006) is based on the idea that good translations should require minimal effort to correct. TER defines four basic operations that can be used to edit hypotheses: single word insertion, single word deletion, single word substitution, and block shift, wherein a contiguous span of words is moved as a single unit. TER is defined as the *minimum* number of equally-weighted edit operations $C_{\texttt{edit}}$ required to transform a translation hypothesis E' into a reference E, normalized by the length of the reference. In the case of multiple references, TER is defined as the minimum edit distance over all references, normalized by the *average* reference length. Formally:

$$\texttt{TER}(E', E) = \frac{C_{\texttt{edit}}(E', E)}{|E|} \tag{2.27}$$

TER scores range from 0 to infinity, though in practice they are capped at 1. While much less widely used than BLEU for optimization and evaluation, TER plays a key role as an approximation of human post-editing effort, discussed in detail in §2.4.3. As TER is an *error* measure, *lower* scores are better.

Meteor: Engineered to address weaknesses of previous metrics, Meteor (Banerjee and Lavie, 2005; Lavie and Denkowski, 2009) is based on the idea that good translations should *align* well to references, just as source sentences align to equivalent target sentences. The Meteor aligner introduces flexible word matching to account for translation variation. When evaluating a hypothesis E' against a reference E, Meteor creates a word alignment based on exact (surface form), stem, and synonym matches. The total number of word matches h in the hypothesis and reference are used to calculate precision and recall:

$$\mathcal{P} = \frac{h(E')}{|E'|} \qquad\qquad \mathcal{R} = \frac{h(E)}{|E|} \tag{2.28}$$

$\mathcal{P}$ and $\mathcal{R}$ are combined in a weighted harmonic mean that scores word choice:

$$\mathcal{F}_\alpha = \frac{\mathcal{P} \times \mathcal{R}}{\alpha \times \mathcal{P} + (1-\alpha) \times \mathcal{R}} \tag{2.29}$$

The number of chunks, ($\texttt{Ch}$) is calculated as the minimum number of contiguously aligned word spans the alignment can be divided into. A fragmentation score ($\texttt{Frag}$) assesses word order and global coherence:

$$\texttt{Frag} = \frac{\texttt{Ch}}{h(E')} \tag{2.30}$$

A final sentence-level score is calculated:

$$\texttt{Meteor}(E', E) = \left(1 - \gamma \times \texttt{Frag}^{\beta}\right) \times \mathcal{F}_\alpha \tag{2.31}$$

Three tunable parameters (α, β, and γ) allow adjusting the relative importance of precision, recall, and reordering. Agarwal and Lavie (2008) tune these parameters to maximize correlation with human judgments of translation quality, leading to greatly improved accuracy over BLEU and TER. Despite higher correlation with human assessments, Meteor is typically not used to tune translation systems.

2.4 Human and Machine Translation

While automatic translation is sometimes sufficient for conveying information across language barriers, many scenarios still require high quality human translation. Governments and international organizations such as the United Nations require accurate translations of content dealing with complex geopolitical issues. Global businesses require polished localization that maintains consistency across dozens of languages. Community-driven projects such as Wikipedia[2] (Wikipedia, 2001) and TED[3] (TED Conferences, 1984) rely on volunteer translators to bring information and resources to diverse language communities. As the amount of data requiring translation has continued to increase, the idea of using machine translation to improve the speed of human translation has gained interest in both the MT and human translation communities. The availability of reliable MT that can serve as a starting point for human translation can potentially save countless hours for both professional and volunteer translators worldwide. Venues such as the AMTA Workshops on Post-Editing Technology and Practice (O'Brien et al., 2012; O'Brien et al., 2014) and Workshop on Interactive and Adaptive Machine Translation (Casacuberta et al., 2014) showcase a variety of innovative approaches to tighter integration of MT with human translation workflows and analyses of the impact of MT on professional translation.

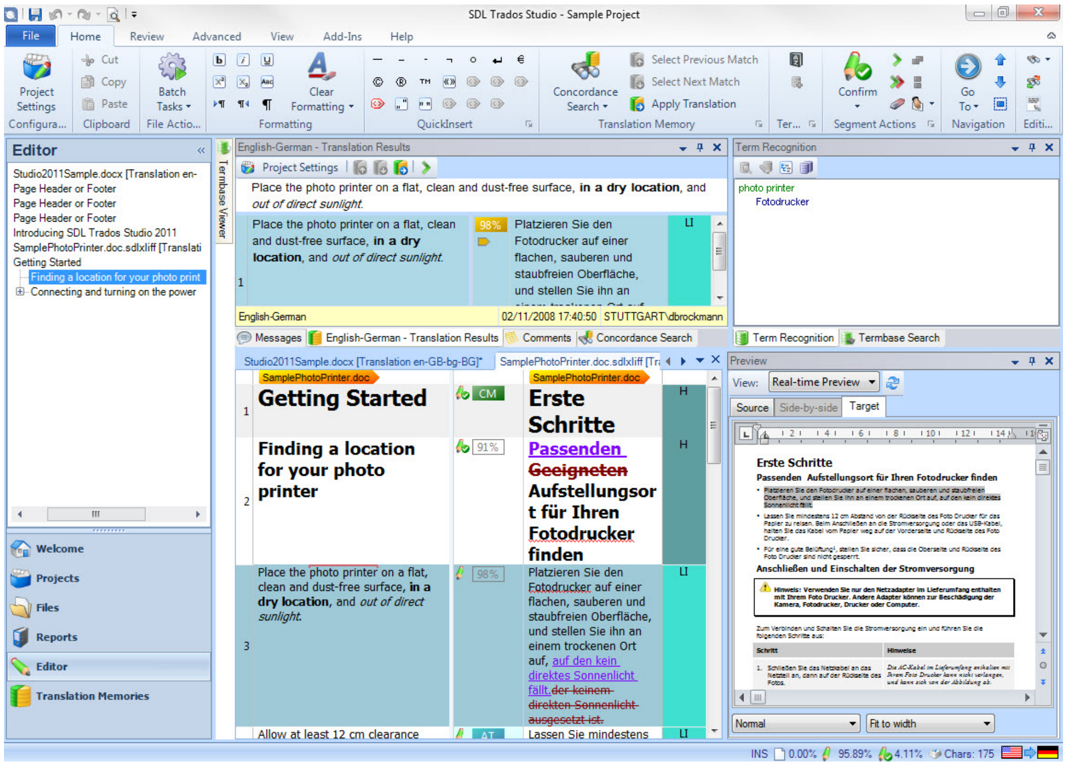
Professional translation projects typically follow the three stage process of translation, editing, and proofreading to ensure high quality results. Most approaches to computer-aided translation (CAT) target the first stage, using real-time MT systems to provide sentence-level translations for humans to post-edit. This section first introduces the professional translation industry, then describes initial work that examines the effects on translation quality and efficiency when traditional statistical MT systems are added to human workflows.

2.4.1 The Professional Translation Industry

While community translation projects rely largely on the efforts of volunteer translators, the global translation industry is primarily driven by the needs of commercial enterprises and international organizations. Companies want to extend their products and services to new markets in diverse countries and cultures. Government and non-profit organizations need to publish information in many languages to maximize accessibility. The vast quantity of content that must be translated to meet these goals has led to the growth of an extensive international language services industry. This industry consists of freelance translators, companies that provide commercial translation services, and vendors that produce globalization software. In a report by Common Sense Advisory, Inc., the global market for outsourced translation services is estimated to be US$37.19 billion in 2014 and rapidly growing (DePalma et al., 2014). As the demand for human quality translation continues to exceed the capacity of the language services industry, there is a push to develop new technologies that better assist human translators, enabling them to work more efficiently. While the demand for such technologies exists primarily in the commercial world, their development can lead to similar improvements in community translation projects.

Translation Workflow: When a company or organization employs a language service provider (LSP) to

[2] http://en.wikipedia.org/wiki/Wikipedia:Translate_us
[3] www.ted.com/participate/translate



Studio2011Sample.docx [Translation en-
Page Header or Footer
Page Header or Footer
Page Header or Footer
Introducing SDL Trados Studio 2011
SamplePhotoPrinter.doc.sdlxliff [Translati
Getting Started
Finding a location for your photo print
Connecting and turning on the power

Place the photo printer on a flat, clean and dust-free surface, **in a dry location**, and *out of direct sunlight*.

1 Place the photo printer on a flat, clean and dust-free surface, **in a dry location**, and *out of direct sunlight*. 98% Platzieren Sie den Fotodrucker auf einer flachen, sauberen und staubfreien Oberfläche, und stellen Sie ihn an

English-German 02/11/2008 17:40:50 STUTTGART\dbrockmann

photo printer
Fotodrucker

Studio2011Sample.docx [Translation en-GB-bg-BG]* SamplePhotoPrinter.doc.sdlxliff [Tr

1	Getting Started	CM	Erste Schritte	H
2	Finding a location for your photo printer	91%	Passenden ~~Geeigneten~~ Aufstellungsort für Ihren Fotodrucker finden	H
3	Place the photo printer on a flat, clean and dust-free surface, **in a dry location**, and *out of direct sunlight*.	98%	Platzieren Sie den Fotodrucker auf einer flachen, sauberen und staubfreien Oberfläche, und stellen Sie ihn an einem trockenen Ort auf, auf den kein direktes Sonnenlicht fällt. ~~der keinem direkten Sonnenlicht ausgesetzt ist.~~	LI
	Allow at least 12 cm clearance	AT	Lassen Sie mindestens	LI

Erste Schritte
Passenden Aufstellungsort für Ihren Fotodrucker finden
Anschließen und Einschalten der Stromversorgung

Figure 2.7: The SDL Trados Studio 2014 translation editing environment

translate a source document into a given target language, the workflow typically follows the translation, editing, and proofreading process. First, a bilingual translator opens the source document using software such as the popular SDL Trados Studio[4] (Shown in Figure 2.7). The translator, often a domain expert, then reads the full source document before beginning translation. Translation proceeds sentence by sentence, involving not only producing grammatically correct, meaning preserving target sentences, but also maintaining correct tone and text formatting. This is often done with the assistance of terminology dictionaries and style guides provided by the client requesting the translation. When the first translator finishes, a second, often more experienced translator edits the translation for grammatical correctness and adherence to terminology and style guides. Finally, a senior translator proofreads the document, approving it for return to the client.

The amount of human labor required in this process, multiplied by the continuously growing amount of content requiring commercial translation, has led the language services industry to incorporate technologies to automate various parts of the translation workflow. Workflow management software allows LSPs to better distribute work among translators, including further outsourcing documents to freelance translators. Translation editing software incorporates automatic suggestions from bilingual dictionaries and translation memories (discussed in §2.4.2). Finally, companies and governments are increasingly looking toward machine translation to reduce the burden placed on human translators.

2.4.2 Machine Translation Post-Editing in Human Workflows

Human translators typically employ translation memory (TM) systems that archive previously translated sentences in the same domain. TM systems use "fuzzy matching" algorithms based on edit distance to locate

[4] http://www.sdl.com/products/sdl-trados-studio/

translations of sentences similar to the source sentence being translated. If a sufficiently similar sentence is found, the translation is suggested. While the translation may not be accurate for the current sentence, it is guaranteed to be a human-quality translation. In contrast, MT systems always produce suggestions for every sentence, but make no quality guarantees. He et al. (2010) conduct a user study that integrates machine translation into a TM system used by human translators. For each sentence, translators are presented with both a human translation from the TM and a MT hypothesis and asked to select the translation that was most suitable for post-editing. The authors find that both TM and MT outputs are selected regularly and in some cases, translators are unable to tell the difference. This result shows promise for the idea of using MT systems to improve TM coverage.

Several organizations have conducted evaluations on the productivity benefits of adding MT to their translation editing workflows. Zhechev (2012) describes experiments comparing post-editing to translating from scratch when localizing Autodesk[5] software. Post-editing significantly improves translation efficiency in several language directions. Poulis and Kolovratnik (2012) describe experiments for the European Parliament that add machine translation to existing tools such as translation memories and bilingual dictionaries. Results are mixed, with MT yielding improved results for some language directions. Tatsumi (2010) conducts a large scale study of MT post-editing in real-world scenarios with professional Japanese translators. Results show that suggestions from a MT system tend to require the same amount of post-editing as "good" matches (above 75% similarity) from a TM when translating in the information technology domain. For TM matches, editors mostly edit lexical items while for MT output, editors mostly fix grammatical errors. Finally, Tatsumi et al. (2012) examine the effectiveness of "crowd-sourcing" post-editing (employing non-experts over the Internet). The authors find that when given MT output, larger pools of non-experts can frequently produce "good enough" translations at least as quickly as experts, often for little or no cost to community projects such as localizing websites. Additional studies by Guerberof (2009), Carl et al. (Carl et al., 2011), and Koehn (Koehn, 2012) all find that translators are both more productive and accurate when post-editing MT output than when translating from scratch.

These recent post-editing studies form an overwhelming consensus: while there is still much room for improvement, the introduction of machine translation tends to improve human translation efficiency and quality. Even in post-editing scenarios where assistive technologies such as bilingual dictionaries and translation memories are already present, the addition of MT consistently leads to quantifiable gains. A survey conducted by Common Sense Advisory, Inc. shows that as of 2013, a growing number of businesses are using MT post-diting to both speed up the translation of material that would be localized anyway and enable the translation of new content that was previously not feasible to localize (DePalma and Sargent, 2013).

2.4.3 Analysis of Post-Editing

Measuring the effect of machine translation post-editing on translator productivity requires the ability to quantify the amount of work a human must do to edit a translation. In an application popularized by the DARPA Global Autonomous Language Exploitation (GALE) project (Olive et al., 2011), MT post-editing effort can be estimated using a semi-automatic evaluation metric. In human-targeted translation edit rate (HTER) (Snover et al., 2006), a human minimally edits a translation hypothesis such that it is grammatical and meaning-equivalent with a reference translation. The TER metric is then used to approximate the number of atomic operations required to post-edit the sentence. In this case, the human editor does not need to be bilingual. HTER can be used either as a MT evaluation metric, representing the *minimum distance* between a system's output and any possible translation, or as an approximation for the amount of human effort required for post-editing. Tasks such as the 2012 NAACL Workshop on Statistical Machine Translation

[5]http://www.autodesk.com/

Quality Estimation task use HTER in this way (Callison-Burch et al., 2012).

Other methods for quantifying post-editing effort include work by Koponen et al. (2012) that shows longer post-editing times to be correlated with certain types of errors that translators rate as more difficult to correct. Lacruz et al. (2012) connect longer pauses in post-editing activity to more challenging edits; translators need to spend more time mentally processing difficult cases before mechanically editing the translation. Subsequent work shows a correlation between the total number of pauses taken during post-editing and the cognitive load placed on translators as determined by examination of keystroke logs (Lacruz and Shreve, 2014b). Blain et al. (2011) take a more qualitative approach to understanding post-editing by introducing a measure based on *post-editing actions*. Edits are grouped into linguistically interpretable actions such as NP structure change, verb agreement correction, and multi-word expression correction. A common theme that emerges from this recent work is that while quantifying post-editing effort is an important component of evaluating human and machine translation workflows, it is a difficult problem that remains only partially solved.

Chapter 3

Online Learning for Machine Translation

When a machine translation system outputs hypotheses for human post-editing, every translation error costs time to correct. Ideally, a translation system should be able to learn from correction and avoid making the same mistakes repeatedly. Every time a human translator corrects a machine translation hypothesis, a new bilingual sentence pair is produced. However, the standard translation models described in Chapter 2 are not designed to incorporate this feedback. These models use the *batch* learning paradigm, wherein all training data is used to estimate a model (translation system) and the model is then use to make predictions (translation of new input sentences). Incorporating new data into the model requires pooling it with initial training data, rerunning word alignment, and estimating new translation models. As this process is computationally expensive, it is typically weeks, months, or longer between system re-trainings. This leads to post-editors' spending their time correcting the same translation errors repeatedly.

Alternatively, the task of machine translation for post-editing can be cast as an *online* learning task. In the online learning paradigm, a task proceeds as a series of trials. Each trial consists of the following three stages: (1) the model makes a prediction, (2) the model receives the "true" answer, and (3) the model updates its parameters. Post-editing workflows fit naturally into this paradigm. In the prediction stage, the translation system produces an initial hypothesis. A human post-editor then edits the hypothesis to produce the "true" or "gold standard" translation. Finally, the system uses the new source-target sentence pair to update the translation model. While the process of post-editing naturally produces the bilingual sentence pairs in step 2, traditional translation models are not equipped to incorporate this data. In this chapter, we describe three extensions to traditional translation systems. We begin by presenting an online method for translation grammar estimation that immediately incorporates new training instances (§3.2). We then discuss running an online optimizer during decoding to keep feature weights synchronized with the online grammar (§3.3). We finally describe an extended feature set that allows the online translation model to better leverage post-editing data (§3.4). Combining these individual components results in a highly adaptive MT system that immediately learns from human feedback and can avoid making the same mistakes repeatedly.

3.1 Related Work

Rapidly updating machine translation systems based on user feedback is an idea that predates phrase-based MT. NISHIDA et al. (1988) use post-editing data to correct errors in a *transfer-based* MT system. This system operates by parsing source sentences and performing a series of lexical (word-based) and syntactic (structural) transformations that ultimately produce target language sentences. By running post-edited output through a reverse (target to source) translation system, the authors aim to identify and correct transfer components that produce errors. Brown (1996) describes an *example-based* MT system that supports incremental model updates. This system, which can be seen as a precursor to on-demand phrase-based translation models, indexes the bilingual training text rather than pre-computing a translation grammar. At runtime, spans of input text are matched against contextually similar occurrences in the training text and translated accordingly. New bilingual sentences, such as those created by post-editing, can be added to the indexed training text at runtime.

More recent work has focused on adapting statistical phrase-based MT systems to immediately leverage new data. Approaches generally fall into the categories of adding new data to translation grammars and using incremental data to adjust feature weights. In the first case, Nepveu et al. (2004) use cache-based grammars and language models that incorporate incrementally translated data from the current document in a computer-aided translation scenario. Bertoldi et al. (2013) show that similar cache-based models lead to improvements in translator productivity when post-editing with an adaptive MT system. Ortiz-Martínez et al. (2010) describe a method for handling new bilingual sentence pairs generated during translation. In addition to feature scores, sufficient statistics are kept for phrase pairs in the translation grammar. When a new sentence pair is available, it is aligned with an iterative word alignment model and new phrase instances are extracted. These instances are used to add new rules to the grammar and update the appropriate sufficient statistics that are in turn used to recompute feature scores. Hardt and Elming (2010) demonstrate the effectiveness of maintaining a distinction between background data (the initial data used to build systems) and post-edit data in an online translation system. The authors also show the feasibility of using pre-existing human reference translations to conduct simulated post-editing experiments, proceeding as follows. A system with both a standard and post-edit-specific grammar translates sentences in a data set. After each sentence is translated, it is aligned to a reference translation as a substitute for post-editing and phrases extracted from the new sentence pair are added to the post-edit-specific grammar similarly to Ortiz-Martínez et al. (2010). While not expressly targeting the application of post-editing, Levenberg et al. (2010) describe a method for incorporating new bilingual data into on-demand grammar extraction as it is available. Introducing a version of the suffix array data structure that can be dynamically updated, the authors are able to add to the data pool from which sentence-level translation grammars are sampled. Sanchis-Trilles (2012) proposes a strategy for online language model adaptation wherein several smaller domain-specific models are built and their scores interpolated for each sentence translated. Interpolation weights depend on the domain of the sentence being translated, allowing the decoder to trust more relevant monolingual data for each sentence.

Focusing on incrementally updating feature weights with post-editing data, Martínez-Gómez et al. (2012) and López-Salcedo et al. (2012) show improvement under some conditions when using techniques including passive-aggressive algorithms, perceptron, and discriminative ridge regression to adapt weights for MT systems initially tuned using MERT. This work also uses reference translations to simulate post-editing. Saluja et al. (2012) introduce a support vector machine-based algorithm capable of learning from binary-labeled examples. This learning algorithm is used to incrementally adjust feature weights given user feedback on whether a translation is "good" or "bad". As with other online learning algorithms, this strategy can be used during both optimization and decoding.

In a different approach to adaptive MT, Simard and Foster (2013) present a post-editing *pipeline* wherein a second stage automatic post-editor (APE) system learns to replicate the corrections made to initial MT output by human translators. As incremental data accumulates, the APE (itself a statistical phrase-based system) attempts to "correct" the MT output before it is shown to humans.

3.2 Online Translation Grammar Adaptation

In this section, we introduce an online version of a rich hierarchical phrase-based translation model that immediately incorporates human feedback by learning new translation rules from post-edited output (Denkowski et al., 2014a). This model is a straightforward extension of the suffix array-backed on-demand model described in §2.2.6 (Lopez, 2008a). Our model offers three key advantages over previous work on online

Feature	On-Demand	Online
`CoherentP(e|f)`	$\log\left(\frac{\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})}{\lvert\mathcal{S}\rvert}\right)$	$\log\left(\frac{\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})+\mathcal{C}_{\mathcal{L}}(\bar{f},\bar{e})}{\lvert\mathcal{S}\rvert+\lvert\mathcal{L}\rvert}\right)$
`SampleCount(f)`	$\log(\lvert\mathcal{S}\rvert)$	$\log(\lvert\mathcal{S}\rvert+\lvert\mathcal{L}\rvert)$
`Count(f,e)`	$\log\left(\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})\right)$	$\log\left(\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})+\mathcal{C}_{\mathcal{L}}(\bar{f},\bar{e})\right)$
`Singleton(f)`	$\mathcal{C}_{\mathcal{S}}(\bar{f})=1$	$\mathcal{C}_{\mathcal{S}}(\bar{f})+\mathcal{C}_{\mathcal{L}}(\bar{f})=1$
`Singleton(f,e)`	$\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})=1$	$\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})\ +\ \mathcal{C}_{\mathcal{L}}(\bar{f},\bar{e})=1$
`PostEditSupport(f,e)`	0	$\mathcal{C}_{\mathcal{L}}(\bar{f},\bar{e})>0$

Table 3.1: Phrase feature definitions for on-demand and online translation models.

translation models. First, by moving from a traditional phrase-based model to a hierarchical model, rules learned from post-edited data can encode non-local *reordering* in addition to new translation choices. Second, our model maintains all sufficient statistics required to use the powerful suffix array-backed features described in §2.2.6. These features are shown by Lopez (2008b) to significantly outperform the standard feature set used in prior models. Third, by using existing bilingual text to *simulated* post-editing, we can learn appropriate weights for our online feature set using standard optimization algorithms such as minimum error rate training.

3.2.1 Grammar Extraction

The starting point for our model is the on-demand translation model described in §2.2.6 (Lopez, 2008a; Lopez, 2008b). Rather than using all bilingual training data to build a single, large translation grammar, this approach uses a suffix array to index the data so that grammars can be estimated as needed. When a new sentence needs to be translated, the suffix array is used to rapidly build and score a sentence-specific grammar. Rules in on-demand grammars are generated using a sample $\mathcal{S}$ for each source phrase $\bar{f}$ in the input sentence. The sample, containing phrase pairs $\langle\bar{f},\bar{e}\rangle$, is used to calculate the following statistics:

- $\mathcal{C}_{\mathcal{S}}(\bar{f},\bar{e})$: count of instances in $\mathcal{S}$ where $\bar{f}$ aligns to $\bar{e}$ (phrase co-occurrence count).
- $\mathcal{C}_{\mathcal{S}}(\bar{f})$: count of instances in $\mathcal{S}$ where $\bar{f}$ aligns to any target phrase.
- $\lvert\mathcal{S}\rvert$: total number of instances in $\mathcal{S}$, equal to number of occurrences of $\bar{f}$ in training data, up to the sample size limit.[1]

These statistics are used to instantiate translation rules $X\rightarrow\bar{f}/\bar{e}$ and calculate scores for the phrase feature set shown in the "on-demand" column of Table 3.1, including the powerful *coherent* translation score.

[1]In practice, a maximum sample size of 300 is used, as this is the point where performance typically levels off. The limit is frequently enforced for short, common phrases that appear up to millions of times in the training data. Rarer phrases may only occur once or a few times in the training data and thus in S.

Although grammars are not sampled until needed, the suffix array is pre-indexed and does not facilitate adding new data.

To accommodate new bilingual data from post-editing, our system maintains a dynamic lookup table for incremental data in addition to the static suffix array for background data. As this lookup table handles a relatively small amount of data compared to the suffix array, it can be implemented with simple data structures such as hash tables.[2] When a human translator edits a MT hypothesis, the sentence pair resulting from the input sentence and post-edited translation is word-aligned with the same model used for the initial data. This process (often called forced alignment) is the only approximation our grammar extraction algorithm makes with respect to the original.[3] Aligned phrase pairs are then stored in the lookup table and phrase occurrences are counted on the source side. When a new grammar is extracted, our model uses all training instances extracted from previously post-edited sentences to learn translation rules. The suffix array sample $\mathcal{S}$ for each $\bar{f}$ is accompanied by an exhaustive lookup $\mathcal{L}$ from the lookup table. Statistics matching those from $\mathcal{S}$ are calculated from $\mathcal{L}$:

- $\mathcal{C}_{\mathcal{L}}(\bar{f}, \bar{e})$: count of instances in $\mathcal{L}$ where $\bar{f}$ aligns to $\bar{e}$.
- $\mathcal{C}_{\mathcal{L}}(\bar{f})$: count of instances in $\mathcal{L}$ where $\bar{f}$ aligns to any target phrase.
- $|\mathcal{L}|$: total number of instances of f in post-editing data (no size limit).

Combined statistics from $\mathcal{S}$ and $\mathcal{L}$ are used to calculate the "online" feature set defined in Table 3.1. An additional indicator feature `PostEditSupport(f,e)` marks rules that are consistent with post-editor feedback. As this model is hierarchical, phrase pairs in $\mathcal{L}$ can contain other phrase pairs, allowing the model to learn new non-local language phenomena from post-editor feedback.

Like work by Levenberg et al. (2010), this learning process can be seen as influencing the *distribution* from which on-demand grammars are sampled over time. The resulting translation grammars are identical (subject to word alignments and sampling strategy) to the infeasible process of adding each instance to the initial training data, re-aligning, and recompiling the suffix array after every sentence is translated. Our model has the added advantage of tracking which data comes from post-edited data. As this data is likely to be highly relevant to subsequent input sentences, we require all relevant phrase pairs from the lookup table to be included when sampling new grammars. This can be seen as *biasing* the statistical model to prefer data that is more likely to yield relevant translation rules. Additionally, the `PostEditSupport(f,e)` feature allows an optimizer to learn an additional weight for all rules that are consistent with human feedback. In addition to marking new rules, this feature facilitates *disambiguation* of existing rules. For example, if a source phrase has 10 possible translations with similar scores and only one is acceptable in the current context, the model will struggle to select the correct choice. However, if that choice is marked as supported by editor feedback and the model has learned to trust this feature, it has a greater chance of producing a correct translation.

3.2.2 Grammar Extraction Evaluation

We evaluate our online grammar extraction algorithm in all simulated post-editing scenarios outlined in §1.4, translating a mixture of language directions and domains that cover a broad range of difficulty levels.

[2]Our implementation pre-computes all sufficient statistics from each aligned sentence pair and stores them in a series of phrase-indexed hash tables. This structure allows fast lookups and aggregation at the expense of less compact storage, which is not an issue when indexing hundreds or thousands of sentences versus millions. Our later implementation using multiple phrase tables also stores incremental data in a suffix array for more efficient scaling.

[3]In standard unsupervised word alignment, the aligner typically makes multiple passes over the corpus, first aligning the entire corpus with the current set of parameters, then updating parameters based on the alignments. This process converges on both a set of alignments and a set of model parameters. In forced alignment, the model makes a single alignment pass over the data using existing parameters and makes no parameter updates.

	Spanish–English				English–Spanish			
	WMT11	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline (MERT)	*29.3*	31.8	33.3	30.2	*30.3*	30.2	27.7	26.6
Baseline (MIRA)	*28.7*	31.6	33.0	30.1	*30.3*	30.4	27.9	27.1
Online (MIRA)	*29.0*	**32.3**	**34.1**	**31.2**	*30.5*	**30.5**	**28.6**	**28.2**
	Arabic–English				English–Arabic			
	MT08	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline (MERT)	*21.6*	25.7	10.4	10.3	*18.9*	23.6	7.6	**8.2**
Baseline (MIRA)	*21.4*	25.7	10.4	10.4	*18.1*	23.2	6.7	7.1
Online (MIRA)	*21.4*	**26.0**	**11.2**	**11.4**	*18.6*	**24.0**	**7.7**	8.0

Table 3.2: BLEU scores for simulated post-editing experiments using static and online translation grammars. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All online results show statistically significant improvement over MIRA baselines on evaluation sets ($p < 0.05$ in approximate randomization).

	News		TED Talks			News		TED Talks	
	New	Supp	New	Supp		New	Supp	New	Supp
Spanish–English	15%	19%	14%	18%	English–Spanish	12%	16%	9%	13%
Arabic–English	9%	12%	23%	28%	English–Arabic	5%	8%	17%	20%

Table 3.3: Percentages of new rules and post-edit supported rules (both old and new rules for which the `PostEditSupport(f,e)` feature fires) in online grammars by domain.

We begin with two static baseline systems that use the on-demand translation model. The MERT-tuned system serves as a strong "best practice" baseline that generally yields translations with high BLEU scores. The MIRA-tuned system serves as an online learning baseline that, while slightly behind the MERT system in some cases, is more directly comparable to our adaptive MT systems. Our experimental system uses the online translation model and is optimized using MIRA with simulated post-editing as described in §1.4.2 to learn reliable feature weights. Experiments are conducted using the `cdec` decoder (Dyer et al., 2010).

Shown in Table 3.2, our online approach yields significant improvement over the MIRA baseline in all cases and over the MERT baseline in nearly all cases. Gains are larger for TED talks where translator feedback can bridge the gap between domains. Table 3.3 shows the aggregate percentages of rules in online grammars that are entirely new (extracted from post-edit instances only) or post-edit supported (superset of new rules). While percentages vary by language and data set, the overall trend is a combination of *learning* new vocabulary and reordering and *disambiguating* existing translation choices. On average, online grammar extraction requires 10% additional CPU time per grammar and negligible memory, keeping real-time learning and translation viable for live post-editing scenarios.

3.3 Online Parameter Optimization

MT systems are traditionally optimized in batch mode at the corpus level. Optimization begins with a fixed translation model and an initial set of feature weights. These weights are either uniform or pre-set to encode common wisdom about which features tend to get higher or lower weights. For a given development corpus of bilingual source-target sentences, the MT system uses the model and initial weights to produce a list of the most likely hypotheses for each source sentence. An optimizer such as minimum error rate training

(Och, 2003) is then used to select a new set of feature weights that prefers better scoring hypotheses from each list. For MERT, "better scoring" simply means higher metric score, most often BLEU. Once a new set of feature weights is chosen, the MT system re-translates the development corpus using the new weights and the process continues. Once a stopping point is reached (either completing an iteration that produces no previously unseen hypotheses or reaching a fixed limit on the number of iterations), the current set of feature weights is used as the system's final weight vector. The MT system then uses the static translation model and weight vector to translate new input data until the system is retrained.

In our work, we use the margin-infused relaxed algorithm described in §2.3.2 (Crammer et al., 2006a; Chiang et al., 2008; Eidelman, 2012), an online learning algorithm that makes an adjustment to the weight vector after each sentence in the development corpus is translated. However, the confines of batch MT system development require this algorithm to be run in batch mode, similar to MERT. Beginning with a uniform weight vector, MIRA makes a fixed number of passes over the development corpus. In each pass, each sentence is translated and a parameter update is made. The final weight vector from each pass is used as the initial vector for the next pass. The weight vector at the end of the last pass is used as the final static weight vector for the system. Under these conditions, MERT and MIRA are interchangeable in system building pipelines.

When applied to simulated post-editing, batch optimization can be useful, but is still limiting. Our simulated post-editing optimization (with either MERT or MIRA) proceeds as follows. Prior to optimization, one set of grammars is extracted, one per sentence in the development corpus. After each grammar is extracted, the source sentence is aligned to the reference translation, forming a simulated post-editing data point, and incorporated into the model. As all post-edit information is pre-encoded in the grammars and reference translations, any batch optimizer can be run normally, learning a weight for each feature. Learning a single weight for a feature that becomes more powerful as more sentences are translated limits the effectiveness of standard linear translation models, though our experiments in §3.2 show that an optimizer is capable of finding weights that lead to significant improvement in translation quality. These weights can be seen as an average of a feature's effectiveness over time. For the first sentence of a document, grammar rules and feature scores are identical to the static model's. As more sentences are translated, new rules are learned and feature scores are more accurate for the current context.

To address the limitations of batch learning and better fit the post-editing paradigm, we continue running the MIRA optimizer as new input sentences are translated. For each document to be translated, we begin with the set of feature weights resulting from MIRA with simulated post-editing. As each sentence is translated and post-edited (or simulated with a reference translating), MIRA makes an update to the weight vector just as in optimization. This is formally equivalent to treating decoding as an additional optimization pass over additional data. Running MIRA during decoding allows the feature weights as well as the translation grammar to be up to date with all available post-editing data when each sentence is translated. In the only departure from optimization, we increase regularization strength during decoding to prefer smaller weight updates.[4] Whereas during optimization, large updates are required to move from uniform weights to final tuned weights, during decoding, small adjustments allow feature weights to keep pace with incremental changes in the translation grammar while remaining relatively stable. While we use MIRA in our work, any online learning algorithm can be substituted just as different batch optimizers can be plugged into the standard MT system tuning step.

3.3.1 Parameter Optimization Evaluation

We evaluate online parameter optimization in all simulated post-editing scenarios outlined in §1.4. We begin by considering the best performing static baseline system (optimized with either MERT or MIRA) for

[4]In our experiments, we use a maximum step size of $C = 0.01$ during tuning and $C = 0.001$ during decoding.

	Spanish–English				English–Spanish			
	WMT11	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Best Baseline	*29.3*	31.8	33.3	30.2	*30.3*	30.4	27.9	27.1
Grammar	*29.0*	32.3	34.1	31.2	*30.5*	30.5	28.6	28.2
Weights	*29.0*	31.6	33.4	30.4	*30.4*	30.4	28.3	27.1
Both	*29.5*	**32.4**	**35.0**	**31.6**	*30.7*	**30.6**	**29.3**	**28.4**
	Arabic–English				English–Arabic			
	MT08	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Best Baseline	*21.6*	25.7	10.4	10.3	*18.9*	23.6	7.6	8.2
Grammar	*21.4*	26.0	11.2	11.4	*18.6*	24.0	7.7	8.0
Weights	*21.4*	25.9	**11.5**	10.6	*18.4*	23.8	7.7	8.6
Both	*21.8*	**26.1**	**11.5**	**12.0**	*18.7*	**24.2**	**8.2**	**9.1**

Table 3.4: BLEU scores for simulated post-editing experiments comparing online grammar extraction and parameter updates against static baselines. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All results for fully adaptive systems (Both) show statistically significant improvement over baselines on evaluation sets ($p < 0.05$ in approximate randomization).

each language direction. We then compare online weight updates during decoding to online grammar extraction as described in the previous section. Finally, we combine online grammar extraction and parameter updates to form a fully adaptive MT system. All online systems are optimized with simulated post-editing. Experiments are conducted using the `cdec` decoder (Dyer et al., 2010).

Shown in Table 3.4, only making online parameter updates tends to under-perform grammar updates and only marginally outperforms static baselines. However, combining grammar and weight updates leads to a system that significantly outperforms the best baseline in all cases.[5] In some cases, the effect of mixing the two is super-additive, leading to improvement larger than the sum of improvements from the individual techniques. While batch-optimized systems are limited to assigning a static weight to each feature, our fully adaptive system uses sentence-specific weights to keep pace with sentence-specific grammars, unlocking much more of the potential in post-editing data. Table 3.5 contains examples from our system's output on TED data that exemplify key improvements in translation quality.

3.4 Extended Post-Editing Feature Set

While simultaneously updating translation grammars and feature weights can significantly leverage post-editing data, our original formulation is still limited by its use of the basic suffix array feature set. For each feature (such as coherent translation probability or lexical score), all information from background and post-editing data must be collapsed into a single score that is multiplied by a single weight. This leads to two related problems. First, simply summing the sufficient statistics from samples of background and post-editing data does not allow for weighing the contribution of one versus the other. When data is aggregated in this way, it is implicitly weighted by size; 10 occurrences of a phrase pair in large background data can dwarf a single occurrence in small post-editing data despite the known tendency of post-edited data to be more relevant during document translation. Even if a rule appears consistently in post-editing data, its score may be very low as it is unable to overcome the influence of background data. Second, different

[5]Statistical significance testing is conducted using bootstrap resampling and approximate randomization over multiple optimizer runs following Clark et al. (2011b).

Baseline	and changing the definition of what the **Zona Cero** is .
Adaptive	and the changing definition of what the **Ground Zero** is .
Reference	and the changing definition of what **Ground Zero** is .
Baseline	was that when we **side by side** comparisons with **coal** , **timber**
Adaptive	was that when we did **side-by-side** comparisons with **wood charcoal** ,
Reference	was when we did **side-by-side** comparisons with **wood charcoal** ,
Baseline	There was a way – **there was one** –
Adaptive	There was a way – **there had to be a way** –
Reference	There was a way – **there had to be a way** –

Table 3.5: Translation examples from baseline and fully adaptive systems of Spanish TED talks into English. Examples illustrate (from top to bottom) learning translations for new vocabulary items, selecting correct translation candidates for the domain, and learning domain-appropriate phrasing.

features require different amounts of data to be estimated reliably. As words occur more frequently than whole phrases, lexical scores stabilize with fewer training sentences than phrase translation probabilities. As such, features may disagree on whether a rule is likely given the background and post-editing data. All responsibility for resolving both of these problems is placed on the post-edit support feature. While this features does typically receive a large positive weight, preferring rules consistent with post-editing, a single score is insufficient for discriminating between rules that have different frequencies in the background and incremental data and several (possibly disagreeing) feature scores.

We address these problems with an extended feature set that presents the decoder with more fine grained information about the likelihood of translation rules in background and post-editing data. When scoring each translation rule $X \to \bar{f}/\bar{e}$ with a background suffix array sample $\mathcal{S}$ and dynamic post-editing lookup $\mathcal{L}$, we compute three instances of each basic suffix array feature h:

- $h_{\mathcal{S}\cup\mathcal{L}}$: the feature score computed on the union (aggregation) of background and post-editing data, equivalent to the version of h in the online translation model
- $h_{\mathcal{S}}$: the feature score computed only on the background data, equivalent to the version of h in the static baseline model
- $h_{\mathcal{L}}$: the feature score computed only on the post-editing data

Each feature is visible to the decoder and has an independent feature weight, effectively tripling the size of the phrase feature set.[6] This allows the translation system to weigh the contribution of background versus post-editing data on a per-feature basis. In a linear translation model, this is formally equivalent to the following process. For each rule, the weighted score for each feature $w \cdot h$ is computed as follows:

$$w \cdot h = w_{\mathcal{S}\cup\mathcal{L}} \cdot h_{\mathcal{S}\cup\mathcal{L}} + w_{\mathcal{S}} \cdot h_{\mathcal{S}} + w_{\mathcal{L}} \cdot h_{\mathcal{L}} \tag{3.1}$$

First, the score is initialized with the weighted value from the union of background and incremental data (equivalent to the simple online grammar feature as described in §3.2). Next, the score is adjusted up or down by the weighted score from the background data. Finally, the score is adjusted by the weighted score from post-editing data. The score for each component is up to date with all incremental data for the current sentence and the weight reflects the optimizer's confidence in each data source at the current point in the

[6]Multiplying a feature set by creating domain-specific copies is originally introduced by Daumé (2007). Clark (2015) successfully applies this approach to MT domain adaptation with static data. Our extension transforms the values of certain feature copies over time using new data.

document. By weighing both data sources and feature scores separately, the translation system can make significantly more informed decisions about which translations are likely given the empirical reliability of available data.

In a final extension, we replace the single post-edit support feature with two additional rule-level features that are in turn computed three times as above:

$$\texttt{RuleCount(f,e)} = 1 \qquad \texttt{WordCount(f,e)} = |\bar{e}| \tag{3.2}$$

These features inform the decoder how many words and rules in the translation hypothesis originate from the background and post-editing data. The rule count feature computed on only the post-editing data is equivalent to the post-edit support feature. The rule count feature computed on background data becomes a "background support" feature while the word counts serve as more fine-grained versions of these two support features. This allows the decoder to make useful distinctions between which phrases are unique to one data source or the other, a task not possible with a single support feature.

3.4.1 Extended Feature Set Evaluation

Whereas previous experiments are conducted with `cdec` (Dyer et al., 2010), our extended feature set is implemented in the Moses toolkit (Koehn et al., 2007). This is primarily due to the recent availability of an efficient implementation of dynamic suffix array-based translation grammars (Germann, 2014) and our own implementation of decoding with multiple grammars. We implement our extended feature set as three distinct suffix array grammars: one built on background data, one on post-editing data, and one on the union of all data (the three conditions described above). At the beginning of each document, the union grammar is identical to the background grammar and the post-editing grammar is empty. For each sentence, the decoder is supplied with the union of rules from all grammars; any rule contained in any grammar appears in the rule table with the scores from all grammars. If a rule is not present in one of the grammars, zeroes are supplied for that portion of the feature set. This is formally equivalent to maintaining a single grammar containing rules extracted from and scored by all data sources as described in Equation 3.1.

We evaluate our extended feature set in all simulated post-editing scenarios outlined in §1.4. We begin by replicating the experiments in the previous sections using the Moses toolkit. Starting with the same bilingual and monolingual training data as previous experiments, we build baseline suffix array systems. Rather than evaluating multiple optimizers, we optimize all systems with the version of MIRA described by Cherry and Foster (2012), which has been shown to match or outperform MERT in a wide range of scenarios. We then reimplement the simple online system with incremental grammar updates and a post-edit support feature. We compare the extended feature set to this system directly without updating feature weights at runtime. We finally add runtime MIRA updates in the most sophisticated version of our adaptive MT system.

Shown in Table 3.6, all online systems outperform all baselines in all cases. Our extended feature set frequently leads to significant gains over the simple online feature set, especially in out-of-domain scenarios. Combining the extended feature set with runtime weight updates leads to the largest absolute scores and relative gains over baselines in any of our experiments.

3.4.2 Analysis of Adaptation

In addition to reporting standard automatic metric scores at the corpus level, we conduct a deeper analysis of the adaptation characteristics of our online systems. We begin by focusing on the latter half of each evaluation set. For each set in our experimental setup, we score the second half of every MT output against the second half of every reference translation and report the scores in Table 3.7. While it might be expected that score differences are more dramatic in the second half of each set due to the larger amount of incremental data available for model adaptation, the relative gains in each case are nearly identical to those measured on

	Spanish–English				English–Spanish			
BLEU	*WMT11*	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*29.3*	31.6	34.0	30.2	*30.5*	30.9	27.0	26.5
PE Support	*30.1*	32.1	35.7	32.0	*31.4*	**31.7**	28.4	27.8
Extended	*30.7*	32.4	**36.2**	32.1	*31.6*	**31.7**	28.8	28.2
+ Weights	*30.9*	**33.0**	36.1	**32.4**	*31.6*	**31.7**	**29.8**	**28.9**
Meteor	*WMT11*	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*34.1*	35.2	35.2	33.7	*56.1*	56.5	55.1	54.7
PE Support	*34.3*	35.2*	35.9	34.5	*56.7*	56.9	56.3	55.6
Extended	*34.5*	**35.3**	**36.2**	34.6	*56.8*	56.9	56.6	**56.0**
+ Weights	*34.4*	35.1*	**36.2**	**34.8**	*56.8*	**57.0**	**56.7**	55.5
TER	*WMT11*	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*51.7*	50.0	44.0	47.2	*52.2*	51.0	53.8	56.1
PE Support	*51.2*	49.5	43.0	45.6	*51.5*	50.5	52.4	54.9
Extended	*50.6*	49.2	**42.5**	**45.4**	*51.3*	50.5	52.0	54.4
+ Weights	*50.2*	**48.5**	42.8	45.9	*50.6*	**50.4**	**48.8**	**51.1**

	Arabic–English				English–Arabic			
BLEU	*MT08*	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*22.2*	26.0	11.2	11.5	*19.1*	23.7	7.8	8.7
PE Support	*22.7*	26.8	14.7	15.8	*19.6*	24.0	8.5	9.4
Extended	*23.1*	27.5	**15.1**	15.8	*20.2*	24.6	9.3	10.3
+ Weights	*23.1*	**27.8**	**15.1**	**16.0**	*20.1*	**24.8**	**9.5**	**10.7**
Meteor	*MT08*	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*30.4*	32.5	21.6	21.6	*36.0*	41.7	21.4	22.4
PE Support	*30.7*	32.8	24.6	**25.0**	*36.5*	42.0	22.7	23.7
Extended	*30.9*	33.2	**24.8**	**25.0**	*37.4*	**42.9**	23.7	24.7
+ Weights	*30.9*	**33.5**	24.6	24.9	*37.1*	42.7	**24.1**	**25.5**
TER	*MT08*	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*59.4*	54.2	73.1	72.9	*64.0*	57.7	75.6	74.2
PE Support	*59.0*	53.7	68.5	67.5	*63.4*	57.4	74.6	73.1
Extended	*58.3*	52.7	69.0	67.8	*62.9*	56.7	74.4	72.8
+ Weights	*58.4*	**53.1**	**66.8**	**65.5**	*62.7*	**56.2**	**73.0**	**71.0**

Table 3.6: Automatic metric scores for baseline systems, simple and extended online feature sets, and fully adaptive systems with runtime weight updates. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate best scores on held-out test sets. All adaptive systems (PE Support, Extended, and + Weights) show statistically significant improvement over respective baselines ($p < 0.05$ in approximate randomization) unless marked with an asterisk.

	Spanish–English				English–Spanish			
	WMT11	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*34.2*	38.7	32.0	31.6	*37.1*	37.7	27.8	27.3
PE Support	*35.3*	39.1	34.4	33.3	*38.0*	**38.7**	29.3	28.7
Extended	*36.0*	39.3	**34.7**	33.3	*38.4*	38.6	29.8	28.7
+ Weights	*36.3*	**39.9**	34.5	**33.4**	*38.3*	38.6	**30.6**	**29.2**
	Arabic–English				English–Arabic			
	MT08	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*19.7*	19.5	12.4	11.0	*16.3*	16.4	8.4	8.3
PE Support	*20.2*	20.6	16.8	15.7	*16.8*	16.6	9.0	9.2
Extended	*20.7*	20.9	17.4	15.9	*17.4*	17.2	10.1	10.4
+ Weights	*20.6*	**21.2**	**17.7**	**16.1**	*17.4*	**17.5**	**10.8**	**10.5**

Table 3.7: BLEU scores for all systems on the latter half of each evaluation set. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All adaptive systems (PE Support, Extended, and + Weights) show statistically significant improvement over respective baselines ($p < 0.05$ in approximate randomization) unless marked with an asterisk.

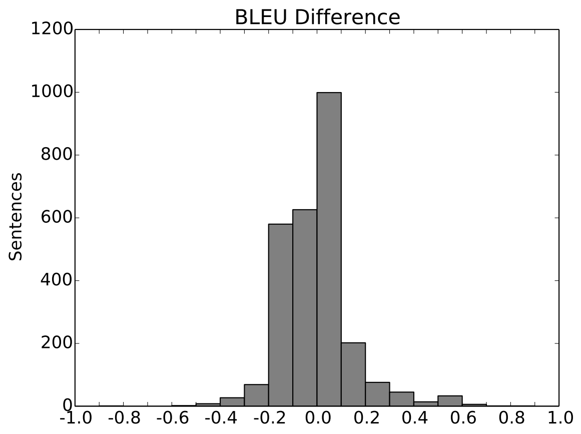
the full sets. These results support the idea that our systems actually begin fitting the target document with relatively little data and continue to yield significantly better translations throughout each document.

To explore the effect of adaptation at a more fine grained level, we use the BLEU and Meteor metrics to compare individual sentences between static and adaptive systems (Papineni et al., 2002; Denkowski and Lavie, 2011). Focusing on the Arabic-English TED talk domain where the differences between systems are most pronounced (TED1), we select the mean-scoring translation run from the baseline and best-performing adaptive system. Each MT output is scored with BLEU and Meteor at the sentence level and the differences are reported in Table 3.8 and Figure 3.1. Despite the fact that adaptive systems have access to more in-domain data, updating the models yields a mix of improved and degraded sentences. While the sum of improvements significantly outweighs the sum of degradations, some sentences are actually better translated by the baseline system. This is not an unusual result in machine translation; making changes to large statistical models that perform a difficult language processing task rarely leads to uniform improvement. To examine this mixed result, we inspect the translations showing the largest improvements and degradations in this data set. Shown in Table 3.9, the first group of sentences shows significant improvement, capturing domain-specific phrasing and terminology. The second group shows only slight degradation, such as dropping a single word. This is consistent with the explanation that learning from post-editing data does not degrade the model but rather changes where errors arise. Large statistical MT systems produce some amount of *noise* in the form of inserted or deleted words or incorrect translations. The errors introduced by online learning are not new to the system, but rather manifested in a different (and smaller) subset of the evaluation sentences. As such, this result can be seen as a reduction and redistribution of noise rather than degradation. Overall, the adaptive system leverages post-editing data to significantly improve translation quality without any material degradation, following a pattern consistent with a generally additive improvement to a statistical MT system.

	BLEU	Meteor
Baseline	11.2	20.4
Adaptive	15.1	23.3

	BLEU	Meteor
Improved (+)	1377	1523
Degraded (-)	687	649

Table 3.8: Differences in BLEU and Meteor for baseline and adaptive systems at the corpus (left) and sentence level (right).



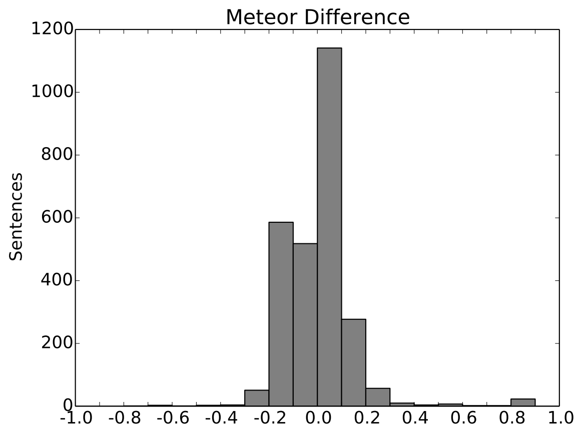


Figure 3.1: Histograms of score differences between baseline and adaptive systems on the Arabic–English TED1 data set according to BLEU and Meteor.

Baseline	An operation by non - sustainable .
Adaptive (+)	It ’s not sustainable .
Reference	It ’s not sustainable .
Baseline	thank you .
Adaptive (+)	thank you very much .
Reference	thank you very much .
Baseline	8,750 and AT - hours of energy per kg from gasoline , propane , Aw - -
Adaptive (+)	8,750 watt - hours of energy from every kilogram of propane or gasoline –
Reference	8,750 watt - hours of energy in every kilogram of propane or gasoline –
Baseline	when you have a million horses ,
Adaptive (-)	when are you million horses ,
Reference	when you have a million horses ,
Baseline	There were two examples here .
Adaptive (-)	There were examples here .
Reference	There were a couple of examples here .
Baseline	to respond to changing circumstances ,
Adaptive (-)	to respond to changing circumstances
Reference	to respond to changing conditions ,

Table 3.9: Translations showing the greatest improvement (top) and degradation (bottom). More sentences in the Arabic–English TED1 data set show improvement than degradation and the magnitude of improvement is significantly larger than the magnitude of degradation.

Chapter 4

Live Post-Editing Evaluation: Software and Experiments

While translation model adaptation, optimization, and evaluation experiments can all be carried out with simulated data, the ultimate goal of our work is to produce real time adaptive MT systems that can be used by actual human translators. As such, the most important measure of efficacy for our systems is the ultimate impact on human productivity in live translation scenarios. Measuring human productivity requires conducting live post-editing experiments, which in turn require an interface between translators and our MT systems. To address this need, we have developed a lightweight web-based translation editing environment called *TransCenter* (a shortened form of "Translation Center") in which users translate documents by post-editing translations served by our adaptive systems. As translators work, TransCenter continues to provide translations for new sentences while relaying already post-edited translations back to the MT system for real time adaptation. Additionally, TransCenter records all user activity for detailed analysis. Pairing TransCenter with our MT systems forms an end-to-end translation and post-editing pipeline that is used to evaluate the adaptive systems described in Chapter 3.

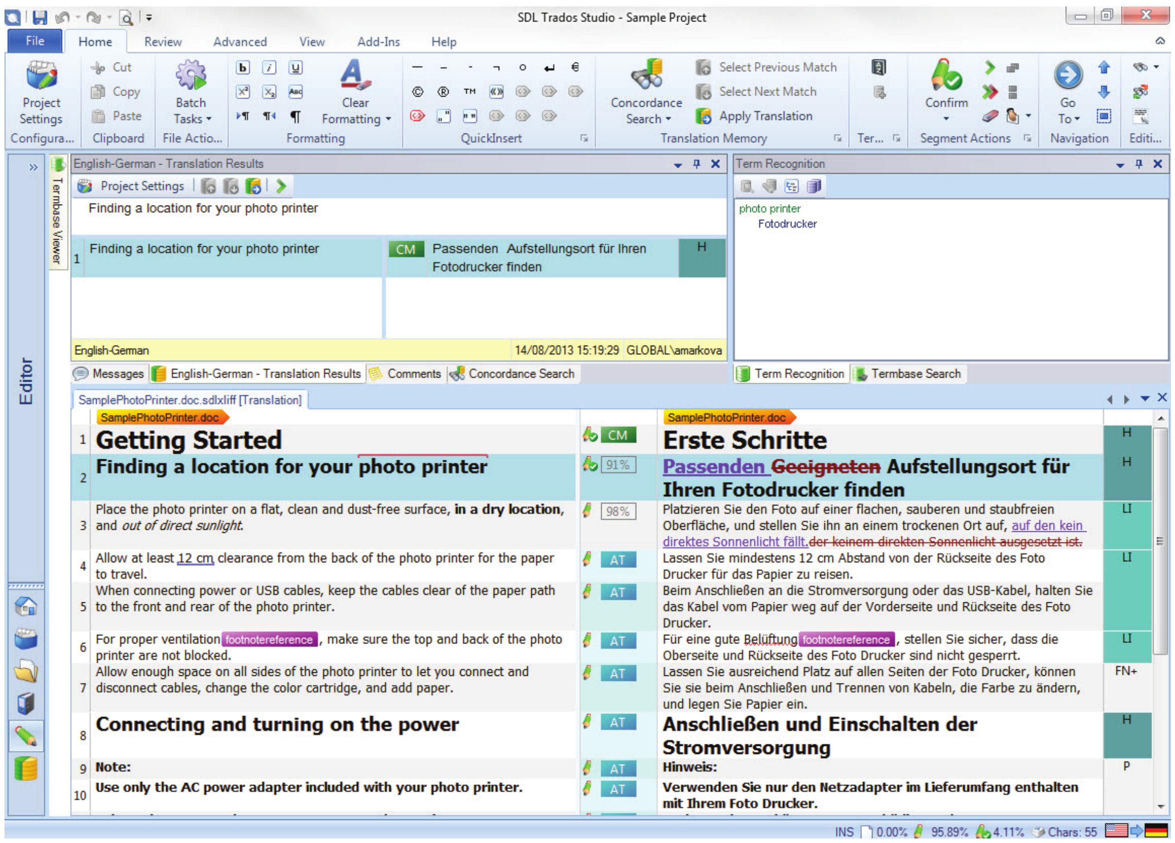
We first describe our TransCenter software (§4.2) then the results of a live post-editing experiment (§4.3). In addition to evaluating the effectiveness of our systems for post-editing, collecting actual user data facilitates a deeper statistical analysis of translation post-editing. We use the collected data to conduct an in-depth analysis of various measures of editing effort and use that information to better optimize our adaptive systems. This work is described in Chapter 5

4.1 Related Work

Software frameworks that bring various assistive technologies into human translation workflows are typically referred to as computer-aided translation (CAT) tools. In general, CAT tools aim to pair technologies such as translation memories and MT with rich text editing that supports widely used file types such as Microsoft Word documents and HTML web pages. The most widely used CAT tool is the commercially developed SDL Trados software suite[1] that provides support for translation memories, terminology dictionaries, and plug-ins for machine translation engines. As shown in Figure 4.1, the Trados interface is built around a two column design (source on the left, target on the right) that has become a standard among CAT tools. Options for other tools such as terminology lookups are organized around the edges of the editing environment. Trados runs on desktop computers and is currently limited to Microsoft Windows.

Recently, the natural language processing research community has developed several open source CAT

[1] `http://www.trados.com/en/`



English-German - Translation Results

Project Settings

Finding a location for your photo printer

1	Finding a location for your photo printer	CM	Passenden Aufstellungsort für Ihren Fotodrucker finden	H

English-German 14/08/2013 15:19:29 GLOBAL\amarkova

Term Recognition

photo printer
Fotodrucker

SamplePhotoPrinter.doc.sdlxliff [Translation]

	SamplePhotoPrinter.doc		SamplePhotoPrinter.doc	
1	**Getting Started**	CM	**Erste Schritte**	H
2	**Finding a location for your photo printer**	91%	**Passenden ~~Geeigneten~~ Aufstellungsort für Ihren Fotodrucker finden**	H
3	Place the photo printer on a flat, clean and dust-free surface, **in a dry location**, and *out of direct sunlight.*	98%	Platzieren Sie den Foto auf einer flachen, sauberen und staubfreien Oberfläche, und stellen Sie ihn an einem trockenen Ort auf, auf den kein direktes Sonnenlicht fällt.~~der keinem direkten Sonnenlicht ausgesetzt ist.~~	LI
4	Allow at least 12 cm clearance from the back of the photo printer for the paper to travel.	AT	Lassen Sie mindestens 12 cm Abstand von der Rückseite des Foto Drucker für das Papier zu reisen.	LI
5	When connecting power or USB cables, keep the cables clear of the paper path to the front and rear of the photo printer.	AT	Beim Anschließen an die Stromversorgung oder das USB-Kabel, halten Sie das Kabel vom Papier weg auf der Vorderseite und Rückseite des Foto Drucker.	
6	For proper ventilation footnotereference, make sure the top and back of the photo printer are not blocked.	AT	Für eine gute Belüftung footnotereference, stellen Sie sicher, dass die Oberseite und Rückseite des Foto Drucker sind nicht gesperrt.	LI
7	Allow enough space on all sides of the photo printer to let you connect and disconnect cables, change the color cartridge, and add paper.	AT	Lassen Sie ausreichend Platz auf allen Seiten der Foto Drucker, können Sie sie beim Anschließen und Trennen von Kabeln, die Farbe zu ändern, und legen Sie Papier ein.	FN+
8	**Connecting and turning on the power**	AT	**Anschließen und Einschalten der Stromversorgung**	H
9	**Note:**	AT	**Hinweis:**	P
10	**Use only the AC power adapter included with your photo printer.**	AT	**Verwenden Sie nur den Netzadapter im Lieferumfang enthalten mit Ihrem Foto Drucker.**	

Figure 4.1: Screenshot of SDL Trados Studio 2014 showing the widely used two column translation editing interface

tools. Two of the largest efforts in this area are the CASMACAT[2] and MateCat[3] projects. CASMACAT (Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation) aims to develop a full-featured open source translator's workbench environment that can be accessed via the web or installed locally (Ortiz-Martínez et al., 2012). Using the common two column format, this workbench focuses specifically on integrating MT into a professional quality translation environment. As the user translates each sentence, an underlying MT system can incrementally suggest the next word or phrase, or translate the remainder of the sentence for the user to post-edit (Alabau et al., 2013). This project also focuses on deeper cognitive analysis of translation editing. Completed work includes experiments showing that a simple, streamlined user interface facilitates faster editing than a complex interface that offers more options (Alabau et al., 2012). The MateCat project also focuses on developing an enterprise grade web-based translation workbench. With an emphasis on machine translation integration and project management, the MateCat tool provides both translations from MT engines and translation memory matches from a large collaborative database (Federico, 2014; Cattelan, 2014). The actual editing interface presented to users follows the established two column format. Both CASMACAT and MateCat are large, multi-institution research projects funded by various government, academic, and commercial entities in the European Union.

In addition to these large projects, several smaller research efforts have innovated on one or more aspects of the traditional CAT environment. Penkale and Way (2012) introduce the SmartMATE online translation

[2]`http://casmacat.eu/`
[3]`https://www.matecat.com`

editing environment that supports translation memories, glossaries, and machine translation and accepts a variety of input file formats. SmartMATE also features the ability to train new MT systems from existing translation memories. Aziz et al. (2012) introduce PET (Post-Editing Tool), a standalone desktop application that provides a simple interface for editing MT output and records data such as editing time, keystrokes, and translator assessments. Moran et al. (2014) extend the open source translation memory tool OmegaT[4] to incorporate accurate sentence-level measurement of translation time. Green (2014) shows the effectiveness of a streamlined translation interface where sentence pairs are arranged vertically rather than horizontally and user interaction is greatly simplified. This tool also supports next phrase prediction and full sentence machine translation that can be controlled using only a keyboard. Additional features such as word-level translations are implemented as hover information when the user mouses over source text. Finally, Schwartz (2014) demonstrates that monolingual domain experts can post-edit machine translations with high reliability. Showing users word-level alignments between source and target improves accuracy despite users' not speaking the source language.

4.2 TransCenter: Post-Editing User Interface

Aiming to be complete translation editing solutions, CAT tools support a wide range of assistive technologies and file formats. By contrast, the goal of our work is to evaluate as accurately as possible the amount of human effort required to correct MT outputs. We intentionally avoid dealing with resources such as translation memories and dictionaries and file formats that contain complex document formatting. While helpful for translators, suggestions from memories and dictionaries complicate analysis. For instance, the availability of good matches from translation memories and dictionaries can lead to very fast editing, masking deficiencies in MT output. Additionally, support for document formatting complicates cognitive effort analysis. If a translator spends several minutes editing a sentence, it is unclear what percentage of that time went toward fixing translation errors versus formatting the target document to match the source style. To maximize the accuracy of collected data, we use only plain text and provide a single machine translation hypothesis per source sentence. While this may lead to additional work for translators, the highly accurate data resulting from translation editing can be used to improve the quality of underlying translation systems. The improved systems can then be plugged in as resources for feature-rich CAT tools to reduce the load placed on human translators. In this sense, our work is largely complementary to work on CAT tools.

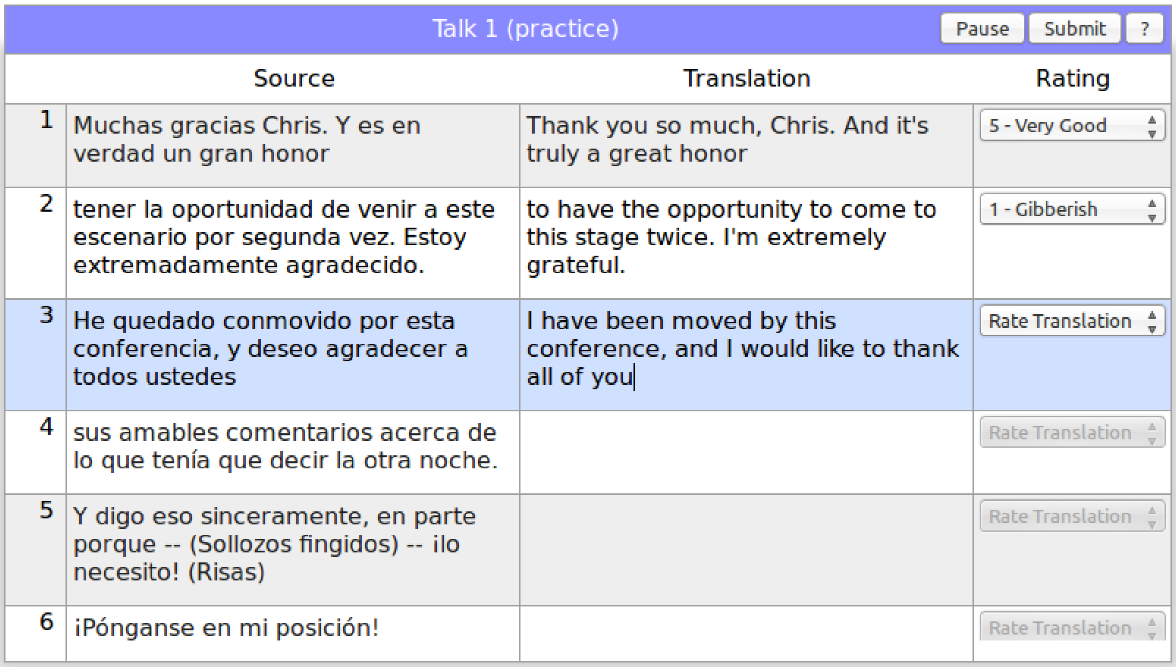
4.2.1 Interface Design

Shown in Figure 4.2, our TransCenter software uses a very simple user interface that follows the two column format that translators are familiar with (Denkowski and Lavie, 2012b). The left column displays the source sentences while the right column, initially empty, is incrementally populated with translations from one of our MT systems as the user works. For each sentence, the translator edits the MT output to be grammatically correct and convey the same information as the source sentence. After editing, the final translation is archived and (if the system is adaptive) fed back to the MT system for learning (Denkowski et al., 2014b). The next sentence is then machine translated and post-edited. The user is additionally asked to rate the amount of work required to post-edit each sentence immediately after completing it, using a scale that ranges from 5 (no post-editing required) to 1 (requires total re-translation).

Our software is designed to make the barrier of collecting post-editing data as low as possible. TransCenter includes the following features to provide a smooth user experience:

- The editing interface is accessed via web browser so that users can work from any computer with an Internet connection.

[4] http://www.omegat.org/



	Source	Translation	Rating
1	Muchas gracias Chris. Y es en verdad un gran honor	Thank you so much, Chris. And it's truly a great honor	5 - Very Good
2	tener la oportunidad de venir a este escenario por segunda vez. Estoy extremadamente agradecido.	to have the opportunity to come to this stage twice. I'm extremely grateful.	1 - Gibberish
3	He quedado conmovido por esta conferencia, y deseo agradecer a todos ustedes	I have been moved by this conference, and I would like to thank all of you	Rate Translation
4	sus amables comentarios acerca de lo que tenía que decir la otra noche.		Rate Translation
5	Y digo eso sinceramente, en parte porque -- (Sollozos fingidos) -- ¡lo necesito! (Risas)		Rate Translation
6	¡Pónganse en mi posición!		Rate Translation

Figure 4.2: Screenshot of the TransCenter post-editing and rating interface

- While users are asked to complete each document in a single session for accurate evaluation, TransCenter automatically tracks state and communicates with the underlying MT system to support stopping and resuming tasks in the case of interruptions such as loss of Internet connectivity.

- An uncluttered interface design and active sentence highlighting allow users to focus on the editing task with minimal distraction.

- Full keyboard navigation allows translation editing and rating, including moving from sentence to sentence, to be completed without using the mouse.

- A Pause button allows users to take breaks if necessary.

4.2.2 Data Collection

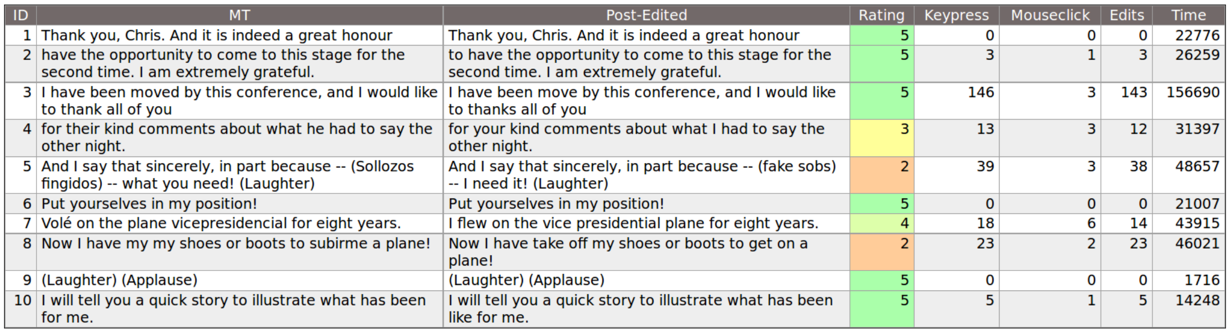
In addition to gathering final edited translations and user ratings, TransCenter records all user interaction at a level of detail sufficient to replay the entire post-editing session. This includes number of keystrokes, number of milliseconds each sentence is focused, and a millisecond-timestamped record of each individual keystroke. Our software uses this information to generate reports of the following measures of human effort:

- The edit distance between the original MT output and the post-edited output according to HTER as described in §2.4.3

- The user rating of each sentence's usability for post-editing

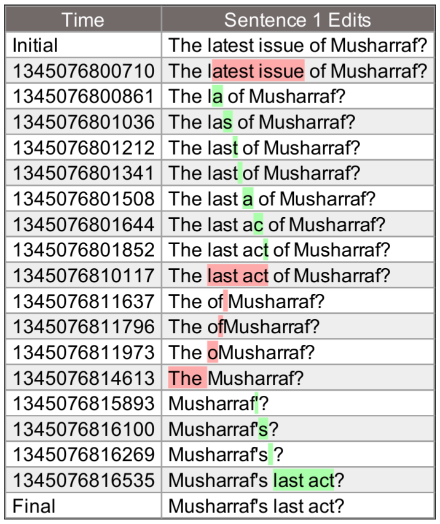
- The number of milliseconds each sentence is focused for editing

- The number of distinct keystrokes used to edit each sentence



ID	MT	Post-Edited	Rating	Keypress	Mouseclick	Edits	Time
1	Thank you, Chris. And it is indeed a great honour	Thank you, Chris. And it is indeed a great honour	5	0	0	0	22776
2	have the opportunity to come to this stage for the second time. I am extremely grateful.	to have the opportunity to come to this stage for the second time. I am extremely grateful.	5	3	1	3	26259
3	I have been moved by this conference, and I would like to thank all of you	I have been move by this conference, and I would like to thanks all of you	5	146	3	143	156690
4	for their kind comments about what he had to say the other night.	for your kind comments about what I had to say the other night.	3	13	3	12	31397
5	And I say that sincerely, in part because -- (Sollozos fingidos) -- what you need! (Laughter)	And I say that sincerely, in part because -- (fake sobs) -- I need it! (Laughter)	2	39	3	38	48657
6	Put yourselves in my position!	Put yourselves in my position!	5	0	0	0	21007
7	Volé on the plane vicepresidencial for eight years.	I flew on the vice presidential plane for eight years.	4	18	6	14	43915
8	Now I have my my shoes or boots to subirme a plane!	Now I have take off my shoes or boots to get on a plane!	2	23	2	23	46021
9	(Laughter) (Applause)	(Laughter) (Applause)	5	0	0	0	1716
10	I will tell you a quick story to illustrate what has been for me.	I will tell you a quick story to illustrate what has been like for me.	5	5	1	5	14248

Figure 4.3: TransCenter document editing report showing a subset of available information: MT output, post-edited output, user rating, number of distinct key presses, number of distinct mouse clicks, number of atomic edit operations (insertions and deletions), and number of milliseconds.



Time	Sentence 1 Edits
Initial	The latest issue of Musharraf?
1345076800710	The latest issue of Musharraf?
1345076800861	The la of Musharraf?
1345076801036	The las of Musharraf?
1345076801212	The last of Musharraf?
1345076801341	The last of Musharraf?
1345076801508	The last a of Musharraf?
1345076801644	The last ac of Musharraf?
1345076801852	The last act of Musharraf?
1345076810117	The last act of Musharraf?
1345076811637	The of Musharraf?
1345076811796	The ofMusharraf?
1345076811973	The oMusharraf?
1345076814613	The Musharraf?
1345076815893	Musharraf'?
1345076816100	Musharraf's?
1345076816269	Musharraf's ?
1345076816535	Musharraf's last act?
Final	Musharraf's last act?

Figure 4.4: TransCenter sentence editing report showing each atomic operation (insertion or deletion) with a millisecond time stamp.

- The number of atomic edit operations (insertions or deletions) used in editing each sentence

The millisecond-timestamped keystroke logs can also be used to compute several *pause measures* that can be used as approximations of the cognitive load placed on post-editors. Recent work in the translation studies community has indicated an association between pause patterns in post-editing activity and post-editors' cognitive effort levels. Empirically validated measures include average pause ratio (APR) and pause to word ratio (PWR) (Lacruz et al., 2012; Lacruz and Shreve, 2014a). TransCenter automatically includes these measures in every report:

$$\text{APR} = \frac{\text{average time per pause}}{\text{average time per reference word}} \qquad \text{PWR} = \frac{\text{\# of pauses}}{\text{\# of reference words}} \tag{4.1}$$

Whereas these measures were previously calculated with manual pause annotations, our software automatically detects and groups pauses using a minimum pause threshold of 2 seconds. All measures listed significantly reduce the amount of effort required for human experts to analyze the amount of work (both mechanical and cognitive) required of translators in a given task.

	Sim PE BLEU	HTER	Rating
Baseline	34.50	19.26	4.19
Adaptive	**34.95**	**17.01**	**4.31**

Table 4.1: Aggregate simulated post-editing BLEU scores, HTER scores, and average translator self-ratings (5 point scale) of post-editing effort for translations of TED talks from Spanish into English.

4.3 Live Post-Editing Experiments

Connecting TransCenter to our MT systems forms a complete post-editing pipeline that enables us to run live evaluations to measure the effect of our online model adaptation techniques on human productivity. These experiments are conducted in collaboration with Kent State University's Institute for Applied Linguistics[5], an academic institution for training professional translators. We begin by establishing an experimental setup wherein observing a pool of human translators can determine which of two MT systems (labeled "A" and "B") is better for post-editing and the degree of difference between them. First, an even number of evaluation documents is selected. Next, translators are assigned to one of two groups, *odd* or *even* based on their (sequentially assigned) user ID. Each user is then asked to translate each document by post-editing the outputs of a MT system. For odd numbered documents, odd numbered translators use MT system A while even numbered users use system B. For even documents, odd translators use system B while even translators use system A. In this way, each document is translated by multiple users with each MT system and the data is balanced for easy normalization. TransCenter does not display any information about which system is being used to translators, or even that there are multiple systems in use. Each translator simply post-edits as normal. This setup scales to any number of users or documents, with larger numbers of each yielding more reliable results.

In the first round of live experiments, we compare the static baseline system described in §1.4 to the adaptive system described in §3.3.1 that updates both translation grammar and weights after each sentence.[6] For our evaluation documents, we draw four excerpts from TED talks that have been translated from Spanish to English, totalling 100 sentences. Our translators are five graduate students from the applied linguistics program training to be Spanish–English translators. We apply the experimental setup described above, having each student use TransCenter to post-edit MT outputs for each document and logging all user interaction. We begin our evaluation by matching previously reported results with simulated post-editing; using pre-existing reference translations for the TED talk excerpts, we run our simulated post-editing pipeline and report BLEU scores. Shown in Table 4.1, with this small amount of data, the improvement from the adaptive system is less than half of a point. However, we see a different case when we evaluate the actual human data with the well established HTER and our user ratings. Here we see a significant improvement in HTER and a slight user preference. This provides evidence that (1) simulated post-editing gains are a good indicator that there will be actual human effort savings, and (2) simulated post-editing is a more difficult scenario, meaning that small gains in simulated scenarios can translate to significant gains in actual post-editing.

4.3.1 Sentence Level Analysis

While our main results are reported over a large enough set of users and documents to account for natural variations in translator style and document difficulty, we also examine the effect of live model adaptation at the sentence level. Using a representative document, we consider the output generated by the MT system

[5] http://appling.kent.edu/

[6] This was the best performing system at the time of these experiments. The extended post-editing feature set is a later development.

Baseline	and sobrepasado sitios dificultosos , gone a more hostile sites
Adaptive (+)	and exceeded difficult sites , gone a more hostile sites
Reference	and climbed over difficult places , went to more hostile places ,
Baseline	... we need inspirarlos , because we need to guide
Adaptive (+)	... we need inspirarlos , because we need to guide us
Reference	We need to inspire them , because they need to lead us
Baseline	to go outside and continue this important
Adaptive (+)	to go outside and continue this so important
Reference	to go out and continue this very important thing

Table 4.2: Translations from baseline and adaptive systems that show the greatest difference when compared against pre-generated references. Adaptation is carried out on live post-editing data.

with and without adaptation (baseline versus adaptive). Following our approach in §3.4.2, we isolate and examine the sentences with the largest automatic metric score differences between the two systems. As the post-edited translations differ from system to system (as they were created by two different users), we use pre-generated reference translations to measure metric difference. However, the incremental data added to the adaptive system is still provided by the live user.

This analysis shows a similar trend to that observed in simulated post-editing experiments: many sentences are significantly improved while some sentences are slightly degraded. However, a manual inspection shows that while the improvements are significant, the degradations are actually significantly less harmful than those seen in the simulated post-editing experiments. Table 4.2 shows the translations with the largest differences, all of which are positive. Of the very few sentences that show degradation, the changes are limited to language variation that metrics fail to recognize as correct translation. Examples include the use of “not talking” instead of “not to talk” and “we have survived” instead of “we survived”. While these examples are taken from a single document, they provide and additional point of evidence in line with the aggregate results: while simulated post-editing is a useful tool for building and evaluating adaptive MT systems, significantly larger gains are realized in live translation scenarios.

Chapter 5

Automatic Metrics of Post-Editing Effort: Optimization and Evaluation

Traditionally, machine translation is treated as a final product that humans will use to read content in their native languages and other language technologies such as information retrieval systems will use directly as input. Approaches to both human and automatic evaluation focus on improving the adequacy of MT system output for these purposes. In contrast, post-editing uses MT as an intermediate step to reduce the amount of work required by human translators. Whereas translation models that incorporate post-editing feedback target this task in terms of model estimation, automatic metrics that accurately evaluate the amount of work required to edit translation hypotheses target post-editing in terms of parameter optimization. Pairing online models with automatic post-editing metrics enables end-to-end translation systems specifically targeting human translation.

In this chapter, we begin by discussing the differences between traditional adequacy-driven MT tasks and post-editing, highlighting the need for improved optimization and evaluation targets (§5.2). We then introduce our extended version of the Meteor metric, shown to correlate well with human post-editing assessments (§5.3). We use data collected with TransCenter to determine what types of edit measures are the most reliable tuning targets for Meteor, leading to the idea of task-specific metrics (§5.4). The chapter concludes with a second round of post-editing experiments that show further improvement when using a version of Meteor based on editing effort as the optimization target (§5.5).

5.1 Related Work

5.1.1 Evaluation

As discussed in §2.3.3, metrics that automatically assign quality scores to translation hypotheses are vital in both parameter optimization and system evaluation. While standard metrics are engineered to correlate with human judgments of translation adequacy, recent work has aimed to predict the amount of post-editing required for MT output, both with and without pre-generated reference translations. Most work measures editing effort with human-targeted translation edit rate (§2.4.3) (Snover et al., 2006). The largest scale evaluation of reference-based evaluation metrics' ability to predict HTER is the 2010 ACL Joint Workshop on Statistical Machine Translation and MetricsMATR (Callison-Burch et al., 2010), in which several metrics are shown to outperform standard BLEU in predicting HTER at the sentence level. The metric with the highest correlation is TER-plus (Snover et al., 2009), an extension of TER that uses word stemmers, synonym dictionaries, and probabilistic paraphrase tables to add various types of weighted substitution operations to edit distance calculation. The Stanford probabilistic edit distance evaluation metric[1] (Wang and Manning, 2012), another weighted edit distance metric using flexible linguistic features including synonymy and paraphrasing, also performs well above the BLEU baseline.

While reference-based evaluation metrics are ideal for system optimization, *quality estimation* metrics that only consider a source sentence and translation hypothesis are useful for deciding if a given translation from a MT system is useful for post-editing. Specia and Farzindar (2010) predict sentence-level HTER using support vector machines with a variety of features including source and target length, N-gram language model scores, number of translations per word in probabilistic dictionaries, and mismatches in punctuation. The authors report good correlation with HTER for translations between English, French, and Spanish. In later work, Specia (2010) combines quality estimation features with several well-known reference-based metrics to train a classifier that predicts sentence-level HTER with improved accuracy. The 2012 NAACL Workshop on Statistical Machine Translation (Callison-Burch et al., 2012) features a quality estimation task where participants predict the amount of post-editing required for sentence-level translations of English into Spanish as assessed by professional translators. No reference translations are provided. Well-performing entries employ a variety of features including measuring translation similarity to outputs of other MT systems (Soricut et al., 2012) and measuring similarity between constituency and dependency parses of source and target sentences (Hardmeier et al., 2012).

As post-editing effort is widely measured by either HTER or professional translators' assessments of editing difficulty, recent work has examined how reliable these measures are for training and evaluating natural language processing systems. Specia and Gimenez (2010) and Koponen (2012) observe cases where human assessments differ from mechanically calculated HTER scores. In cases where sentences are very long or where significant reordering is required, translators tend to classify a sentence as much more difficult to post-edit than its HTER score would indicate. Koponen hypothesizes that these cases require increased cognitive effort that is not adequately captured by simple edit distance. Specia (2011) compare the suitability of post-editing time, distance (HTER), and score (human assessment) for training predictive models. Effort and time are both shown to outperform edit distance, although differences in data sets between languages make generalization difficult.

5.1.2 Optimization

As work on automatic metrics has largely focused on evaluation and quality estimation, the task of using new metrics as objective functions for optimization algorithms has been less explored. Initial work by Cer et al. (2010) shows that optimizing a MT system to a given evaluation metric generally increases translation score on that metric, often at the expense of scores on other metrics. The work shows that there is no compelling reason to move away from standard BLEU at the time. Liu et al. (2011) report improvement when applying the TESLA (translation evaluation of sentences with linear-programming-based analysis) metric to a MERT task. Based on N-gram matches like BLEU, the TESLA metric also allows flexible matching with synonyms and part-of-speech tags and weights N-gram matches based on match type and presence of content words. In a human evaluation, annotators prefer translations from a TESLA-tuned system over a BLEU-tuned system despite the fact that these translations generally receive lower BLEU scores. The 2011 EMNLP Workshop on Statistical Machine Translation (Callison-Burch et al., 2011) features a system optimization task where participants use a provided MERT implementation and development set to tune an existing translation system to various automatic metrics. By human evaluation, no metric outperforms standard BLEU, though several perform comparably.

[1]This metric is described as "Stanford" in the official results of WMT/MetricsMATR 2010. The first published description by the authors is in 2012 under the name "SPEDE".

	Rank	HTER	Translation
Reference	–	–	Only the crème de la crème of the many applicants will fly to the USA.
System 1	$1st$	0.40	Only the crème de la crème from many candidates, it's going to go to the US.
Post-edit 1	–	–	Only the crème de la crème from many candidates will fly to the US.
System 2	$2nd$	0.20	Only crème de la crème of many customers will travel to the US.
Post-edit 2	–	–	Only the crème de la crème of many applicants will fly to the US.

	BLEU	HTER	Translation
Reference	–	–	The problem is that life of the lines is two to four years.
System 1	0.49	0.29	The problem is that life is two lines, up to four years.
Post-edit 1	–	–	The problem is that life of the lines is two to four years.
System 2	0.34	0.14	The problem is that the durability of lines is two or four years.
Post-edit 2	–	–	The problem is that the life of lines is two to four years.

Table 5.1: Cases where lower-ranked (top) or lower BLEU-scoring (bottom) MT outputs require less work to post-edit. Lower HTER indicates fewer edit operations required.

5.2 Motivation: Examination of MT Evaluation for Post-Editing

Large machine translation evaluation campaigns such as the ACL Workshops on Statistical Machine Translation (Callison-Burch et al., 2011) and NIST Open Machine Translation Evaluations (Przybocki, 2009) focus on improving translation adequacy, the perceived quality of fully automatic translations compared to reference translations. As such, current techniques for MT system building, optimization, and evaluation are largely geared toward improving performance on this task. Originally introduced by the Linguistics Data Consortium, adequacy ratings elicit straightforward quality judgments of machine translation output according to numeric scales (LDC, 2005). Recent WMT evaluations (Callison-Burch et al., 2007; Callison-Burch et al., 2011) use *ranking*-based evaluation to abstract away from concepts such as adequacy and grammaticality as well as difficult-to-decide numeric ratings. Human judges are simply asked to rank several MT outputs for the same sentence from best to worst according to a reference translation. It is left up to judges to determine the relative severity of different types of translation errors when comparing translations. Whereas MT systems targeting adequacy should maximize the semantic similarity of automatic translations with reference translations, systems targeting post-editing utility should minimize the effort required by human translators to correct automatic translations. This is most often measured by cased human-targeted translation edit rate (HTER) (Snover et al., 2006) that depends on alignments from the TER metric.

5.2.1 Translation Evaluation Examples

The adequacy and post-editing tasks bear some similarities, as automatic translations that have high similarity to reference translations often require minimal post-editing. However, when MT outputs contain errors, the most adequate translations are often not the easiest to post-edit. Table 5.1 shows two examples from the difficult Czech-to-English translation track of the 2011 EMNLP Workshop on Statistical Machine Translation (Callison-Burch et al., 2011) with minimally post-edited translations and HTER scores. In the first case, the translation deemed more adequate by expert judges actually requires more effort to post-edit. In the second example, sentence 2 is penalized by BLEU for using a different word order from the reference even though it is both more adequate and less work to correct. These examples illustrate types of errors that have a large impact on sentence meaning but require relatively little work to correct, as well as accumulated

r	HTER	Effort
BLEU	-0.30	0.31
TER	0.29	-0.26
HTER	–	-0.60

ρ	HTER	Effort
BLEU	-0.30	0.27
TER	0.29	-0.27
HTER	–	-0.64

Table 5.2: Correlation of metric scores with HTER and effort assessments

minor errors that do not impact meaning, but are cumbersome to correct.

5.2.2 Challenges of Predicting Post-Editing Effort

To empirically evaluate the effectiveness of human and automatic assessments of post-editing effort, we conduct an analysis of annotated post-editing data released as part of a quality estimation task for the NAACL 2012 Workshop on Statistical Machine Translation (Callison-Burch et al., 2012). The starting point of this data is a Spanish–English bilingual corpus of 1832 news sentences. Using the English side as the source, this data is translated into Spanish by a statistical MT system. The target side of the corpus is held out as an independent human reference translation. Each Spanish MT output is rated by 3 bilingual human translators according to its suitability for post-editing. These translators see the source and MT, but not the reference translation. Ratings use the following scale:

1. Incomprehensible: cannot be edited, needs to be translated from scratch
2. 50-70%: requires a significant editing effort
3. 25-50%: various errors and mistranslations need to be corrected
4. 10-25%: generally clear and intelligible
5. Perfectly intelligible: requires little to no editing

The three ratings for each translation are averaged into a single expert assessment score. Finally, the MT outputs are post-edited by another set of human translators to be fluent, meaning-equivalent versions of the source sentences. These translators see only the source and MT, not the reference translations or post-editing assessments. The post-edited translations are used to compute HTER (§2.4.3), the edit distance between MT and post-edited translations (Snover et al., 2006).

Metric Experiments: We use the data described above to explore (1) how well standard automatic MT evaluation metrics predict post-editing effort and (2) how consistent human assessments of post-editing are (Denkowski and Lavie, 2012a). In our first experiment, we replicate the simulated post-editing scoring task that corresponds to the method we use to optimize and automatically evaluate our adaptive MT systems. Given only MT outputs and independent references (not post-edited references), an automatic metric must predict how much post-editing is required to correct the translations. The results give us insight into the reliability of the metrics we are using for this task. We score MT outputs with the BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) metrics (§2.3.3) and report the sentence-level correlation of these scores with HTER and editing effort assessments. To evaluate the consistency of different measures of editing effort, we also report correlation between HTER and effort assessments. For consistency with later work, we report both Pearson's r and Spearman's ρ, discussed in detail in §5.3. Shown in Table 5.2, both BLEU and TER have very low correlation with measures of editing effort. Additionally, there is significant disagreement between HTER and effort assessments. These results can be attributed to two highly related factors.

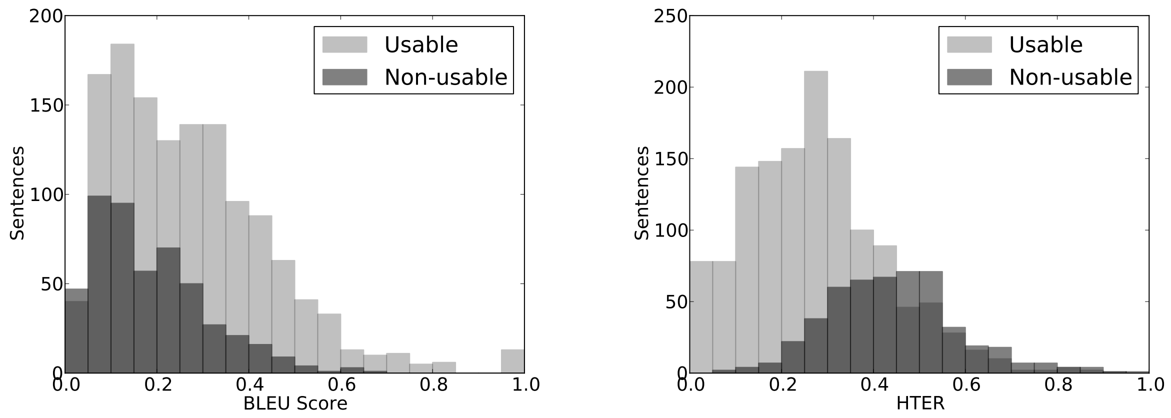


Figure 5.1: Distributions of BLEU scores and HTER for usable and non-usable translations in WMT post-editing data

First, existing metrics developed for adequacy-driven MT tasks appear to be suboptimal for predicting post-editing, failing to capture all information required to make reliable distinctions between translations. While our adaptive MT systems show significant improvement over baselines, the relatively poor predictive power of BLEU indicates that we could see further gains from developing metrics better suited for post-editing. Second, even human translators seem to have difficulty predicting the amount of post-editing that will be required to correct MT output. More reliable "gold standard" measures of editing effort can give tunable evaluation metrics a better prediction target.

In a second experiment, we evaluate BLEU and HTER in a distilled scoring task. We simplify the 5-point human effort predictions into two groups: usable (3-5) and non-usable (1-2), corresponding to whether or not the human translator expects to salvage at least 50% of the MT output. By examining the distributions of sentence-level BLEU and HTER scores for each group, we can see if each metric can make distinctions about post-editing effort at the most basic level. Shown in Figure 5.1, the BLEU score distributions overlap completely and translations are clustered in the same region. No quality threshold (visualized as a vertical line on the graph) can reliably separate usable from non-usable translations. The HTER scores show that an expert is largely able to detect easily correctable translations, judging nearly all translations with HTER under 0.2 to be usable. Above 0.2, translations requiring comparable numbers of edits are judged to be both usable and non-usable. These results illustrate that not only are standard metrics such as BLEU poor at discriminating between easy and difficult to post-edit translations, but measures of post-editing themselves do not always agree. As HTER and human pre-assessments are both removed from the actual editing process, the resulting scores can fail to capture vital information. The results of these two experiments directly motivate the development of better evaluation metrics and gold standard measures of editing effort described in the next sections.

5.3 The Meteor Metric for MT Evaluation and Optimization

Originally developed to more accurately model human acceptability of MT output, the Meteor metric (Banerjee and Lavie, 2005; Lavie and Denkowski, 2009) has consistently shown good correlation with human judgments of adequacy (Lavie and Agarwal, 2007) and preference (Agarwal and Lavie, 2008; Denkowski and Lavie, 2010b; Denkowski and Lavie, 2011). This section describes our extended version of Meteor, casting its features as predictors of post-editing effort. We report results for successfully adapting

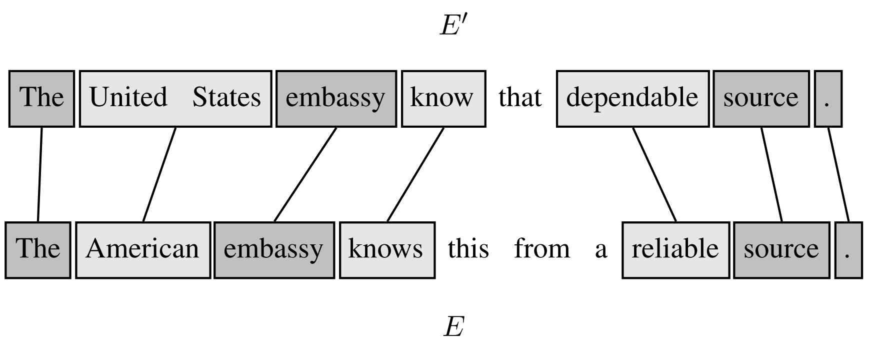


Figure 5.2: Example Meteor alignment between two English sentences. Dark gray shading indicates exact matches while light gray indicates approximate matches (from left to right: paraphrase, stem, synonym).

Meteor to predict HTER scores while at the same time revealing limitations of HTER as a measure of editing effort. We also discuss initial system tuning experiments that show a stability advantage over BLEU.

5.3.1 The Meteor Metric

Meteor is an automatic evaluation metric that scores MT hypotheses by aligning them to reference translations. Alignments are based on several types of flexible matches that go beyond surface forms to identify word and phrase correspondences that would be clear to humans but are missed by standard metrics. Based on an alignment, Meteor determines what information is present in both sentences, what information is present in one sentence but not the other, and to what extent text is reordered between the two sentences. Alignment statistics are combined in a weighted scoring function that can be tuned to maximize correlation with human judgments of translation quality.

Meteor Alignment: Given a translation hypothesis E' and reference translation E, Meteor creates an alignment as follows. First, the search space of possible alignments is constructed by identifying all possible matches between the two sentences according to the following matchers:

- Exact: Match words if their surface forms are identical.
- Stem: Stem words using a language-appropriate Snowball stemmer (Porter, 2001) and match if the stems are identical.
- Synonym: Match words if they share membership in any synonym set according to the WordNet (Miller and Fellbaum, 2007) database.
- Paraphrase: Match phrases if they are listed as paraphrases in the Meteor paraphrase tables. Paraphrase tables are constructed from the bilingual text available as part of the 2010 ACL Workshop on Statistical Translation and MetricsMATR (Callison-Burch et al., 2010) using the statistical phrase table "pivot" approach (Bannard and Callison-Burch, 2005) with additional pruning to improve precision (Denkowski and Lavie, 2011).

All matches are generalized to phrase matches in the form $\langle E'^{i+n}_{i}, E^{j+m}_{j} \rangle$ where i and j are start indices in the hypothesis and reference and n and m are match lengths. Matches are said to *cover* one or more words in each sentence. Exact, stem, and synonym matches always cover one word in each sentence while paraphrase matches can cover any number of words in either sentence. Once matches are identified, the final alignment is resolved as the largest subset of all matches meeting the following criteria in order of importance:

1. Require each word in each sentence to be covered by zero or one matches.
2. Maximize the number of covered words across both sentences.
3. Minimize the number of chunks ($\texttt{Ch}$), where a *chunk* is defined as a series of matches that is contiguous and identically ordered in both sentences.
4. Minimize the sum of absolute distances between match start positions i and j over all matches. (Break ties by preferring to align words and phrases that occur at similar positions in both sentences.)

An example Meteor alignment is shown in Figure 5.2. While the Meteor aligner is most often used as part of scoring translation quality, it can also be used in other tasks that require rich monolingual phrase alignments. Notably, Meteor is used to create alignments between different systems' translation hypotheses of each source sentence as part of the system combination approach described by Heafield and Lavie (2011).

Meteor Scoring: Given an alignment between hypothesis E' and reference E, the Meteor metric score is calculated as follows. First calculate initial statistics:

- $\langle \mathcal{C}_{\texttt{f}}(E'), \mathcal{C}_{\texttt{f}}(E)\rangle$: function words in E' and E. Count any word that appears in the Meteor function word lists estimated from large monolingual data (Denkowski and Lavie, 2011).
- $\langle \mathcal{C}_{\texttt{c}}(E'), \mathcal{C}_{\texttt{c}}(E)\rangle$: content words in E' and E. Count any word that does not appear in the function word lists.
- $\langle h_i(\mathcal{C}_{\texttt{c}}(E')), h_i(C_{\texttt{f}}(E')), h_i(\mathcal{C}_{\texttt{c}}(E)), h_i(\mathcal{C}_{\texttt{f}}(E))\rangle$: the number of content and function words in E' and E covered by each type of match h_i. (For example, counts of content and function words covered by exact matches in the hypothesis and reference.)
- $\texttt{Ch}$: the minimum number of *chunks* (series of matches that are contiguous and identically ordered in both sentences) that the alignment can be divided into.

Calculate weighted precision and recall using match type weights $w_i \in W$ and content-vs-function word weight (δ):

$$\mathcal{P} = \frac{\sum_i (w_i \times (\delta \times h_i(\mathcal{C}_{\texttt{c}}(E')) + (1-\delta) \times h_i(\mathcal{C}_{\texttt{f}}(E'))))}{\delta \times \mathcal{C}_{\texttt{c}}(E') + (1-\delta) \times \mathcal{C}_{\texttt{f}}(E')} \tag{5.1}$$

$$\mathcal{R} = \frac{\sum_i (w_i \times (\delta \times h_i(\mathcal{C}_{\texttt{c}}(E)) + (1-\delta) \times h_i(\mathcal{C}_{\texttt{f}}(E))))}{\delta \times \mathcal{C}_{\texttt{c}}(E) + (1-\delta) \times \mathcal{C}_{\texttt{f}}(E)} \tag{5.2}$$

The harmonic mean of $\mathcal{P}$ and $\mathcal{R}$ parameterized by α (van Rijsbergen, 1979) is then calculated:

$$\mathcal{F}_\alpha = \frac{\mathcal{P} \times \mathcal{R}}{\alpha \times \mathcal{P} + (1-\alpha) \times \mathcal{R}} \tag{5.3}$$

A fragmentation score is calculated using the total number of matched words $\texttt{M}$ (average over hypothesis and reference) and number of chunks ($\texttt{Ch}$):

$$\texttt{M} = \frac{\sum_i (h_i(\mathcal{C}_{\texttt{c}}(E')) + h_i(\mathcal{C}_{\texttt{f}}(E')) + h_i(\mathcal{C}_{\texttt{c}}(E))) + h_i(\mathcal{C}_{\texttt{f}}(E))}{2} \qquad \texttt{Frag} = \frac{\texttt{Ch}}{\texttt{M}} \tag{5.4}$$

The final Meteor score is calculated with fragmentation parameters β and γ:

$$\texttt{Meteor}(E', E) = \left(1 - \gamma \times \texttt{Frag}^{\beta}\right) \times \mathcal{F}_\alpha \tag{5.5}$$

Each of the Meteor scoring statistics can be interpreted as a key predictor of post-editing effort. Precision ($\mathcal{P}$) is an inverse measure of the amount of content in the hypothesis that must be *deleted* to match the reference. Recall ($\mathcal{R}$) inversely measures the amount of content that must be *inserted*. Fragmentation ($\texttt{Ch}$) is a measure of how much *reordering* is required to match the reference. Compared to edit distance-based metrics, Meteor makes a greater distinction between word *choice* and word *order*.

The following metric parameters can be tuned to maximize agreement between Meteor scores and human assessments of translation quality:

- $W = \langle w_i, ..., w_n \rangle$: an individual weight for each type of match, allowing distinctions such as penalizing a stem match, which likely requires editing for grammaticality, more harshly than a synonym match, which may not require editing. There are currently 4 weights: exact, stem, synonym, and paraphrase. The weight for exact matches is fixed at 1.
- α: the balance between precision and recall, allowing greater penalties for either insertion or deletion requirements.
- β, γ: the weight and severity of fragmentation, allowing fine-tuning the cost for reordering text.
- δ: the relative contribution of content versus function words, allowing greater penalties for important words that tend to be more difficult to translate.

Parameter Optimization: Tuning a version of Meteor to approximate a given evaluation task requires a set of n MT outputs with reference translations plus a set of human-annotated scores $Y = \langle y_1, ..., y_n \rangle$ (quality scale assessments, HTER scores, or other numerical scores). Meteor scores the MT outputs, producing metric scores $X = \langle x_1, ..., x_n \rangle$. Ideally, X should be strongly correlated with Y, meaning that a high metric score should correspond to a high human score and vice versa. During tuning, sufficient statistics are calculated for each MT output, allowing it to be rapidly re-scored with various parameter settings. We then conduct an exhaustive parametric sweep over feasible parameter values to maximize correlation between X and Y over all MT outputs. This guarantees globally optimal metric parameters for the data set. Our work uses two different measures of correlation that offer different advantages depending on the task: Pearson's r and Spearman's ρ.

The Pearson product-moment correlation coefficient r measures the *linear* correlation between two variables on a scale from 1 to -1 (Pearson, 1895). The extremes are total correlation (positive or negative) while 0 is no correlation. Given X and Y, this correlation coefficient is calculated:

$$r(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}} \tag{5.6}$$

where $\bar{X}$ and $\bar{Y}$ are means. Pearson's r is useful in cases where relationships between scores should be linear. For example, in MT evaluation, it is desirable for one additional metric point (0.01) to indicate a consistent amount of improvement whether the increase is from 0.09 to 0.10 or from 0.49 to 0.50. By tuning a metric using r, we force the scoring function to be as linear as possible.

Spearman's rank correlation coefficient ρ assesses the extent to which two variables can be described using a monotonic function (Spearman, 1904). To compute ρ, we first convert X and Y into rank lists X' and Y' by replacing the values in each list with ascending integers reflecting their index if the list was to be sorted. Ties are handled by assigning the same averaged rank to each tied item. For example, if $X = \langle 10, 5, 2, 5 \rangle$, $X' = \langle 4, 2.5, 1, 2.5 \rangle$. We then compute ρ as the Pearson's r between the rank lists:

$$\rho(X,Y) = r(X',Y') \tag{5.7}$$

Metric	Tuning Data	P2 r	P3 r
BLEU	–	-0.545	-0.489
TER	–	0.592	0.515
Meteor	P2	-0.642	-0.594
	P3	-0.625	-0.612

Table 5.3: Correlation of metric scores with HTER on GALE data

Task	α	β	γ	δ	w_{exact}	w_{stem}	w_{syn}	w_{par}
WMT Ranking	0.85	0.20	0.60	0.75	1.00	0.60	0.80	0.60
GALE HTER	0.40	1.50	0.35	0.55	1.00	0.20	0.60	0.80

Table 5.4: Comparison of Meteor parameters for ranking (relative adequacy) and HTER (post-editing effort) tasks

Removing the linearity constraint generally allows metrics to reach much higher correlation values while sacrificing some interpretability of absolute scores. This is useful in the case of system optimization where the goal is select a parameter set that yields the best possible translations. Here, accurately selecting the translations that would be highest ranked by humans is often more vital than being able to interpret the final metric scores on the development corpus.

5.3.2 Evaluation Experiments

Meteor has been successfully tuned to replicate several types of human quality judgments. The most widely used "ranking" version of Meteor (Denkowski and Lavie, 2011) is shown to reliably assign higher scores to the types of translations preferred by human annotators in WMT evaluations in a variety of languages (Callison-Burch et al., 2012). A version tuned to numerical adequacy scale scores (Denkowski and Lavie, 2010b) shows good linear correlation with this type of human judgment for English (Callison-Burch et al., 2010)[2]. Current work focuses on using Meteor to predict human post-editing effort.

The first round of experiments that investigate Meteor's ability to predict post-editing effort use HTER scores (§2.4.3) from the DARPA Global Autonomous Language Exploitation (GALE) project (Snover et al., 2006; Olive et al., 2011). We use two sets of HTER scores calculated from post-editing translations into English in two phases of the project: P2 and P3. For each data set, we tune a version of Meteor to maximize Pearson's r. We evaluate Meteor's ability to *fit* the data by measuring correlation on the same data set and ability to *generalize* to other HTER data by measuring correlation on the alternate data set. We evaluate Meteor against the baseline metrics BLEU and TER (§2.3.3). The correlation results of these experiments are shown in Table 5.3 while the optimal metric parameters for Meteor are shown in Table 5.4. While Meteor outperforms all baseline metrics, the parameter set reveals some shortcomings in the formulation of HTER. As HTER makes no distinction between content and function words, δ is near 0.5. As identical base words with different inflections are treated as non-matches by TER, the weight for stem matches is near 0. Whereas these parameters allow greater distinctions to be made for adequacy and ranking data, they make only minimal contribution for HTER data. These results further highlight the need for more accurate editing measures that can be used to train metrics that better predict post-editing effort.

[2]http://www.itl.nist.gov/iad/mig/tests/metricsmatr/2010/results/correlation_English_Adequacy7Average_segment.html

p	HTER	Rating	Keystroke	Time	APR	PWR
HTER	–	-0.84	0.91	0.61	-0.55	0.64
Rating	-0.84	–	-0.82	-0.56	0.46	-0.53
Keystroke	0.91	-0.82	–	0.70	-0.56	0.66
Time	0.61	-0.56	0.70	–	-0.53	0.69
APR	-0.55	0.46	-0.56	-0.53	–	-0.65
PWR	0.64	-0.53	0.66	0.69	-0.65	–

Table 5.5: Spearman's correlation between several automatic measures of editing effort computed by TransCenter

Task	α	β	γ	δ	w_{exact}	w_{stem}	w_{syn}	w_{par}
HTER	0.90	0.10	0.55	0.60	1.00	0.00	0.00	0.80
Keystroke	0.65	0.10	0.55	0.65	1.00	0.00	0.00	0.80
Rating	0.45	0.00	1.00	1.00	1.00	0.20	0.00	0.80

Table 5.6: Comparison of Meteor parameters for different measures of editing effort

5.4 Improved Editing Measures for Improved Metrics

The post-editing data collected by our TransCenter software (§4.2) allows us to explore a range of possible alternatives to HTER for measuring human editing effort. We augment the data collected in our adaptive MT validation experiments (§4.3) with a second similar round of post-editing using a new set of translators from Kent State University's applied linguistics program.[3] Combined, this data consists of 1000 post-edited sentences of TED talks translated from Spanish into English (Denkowski et al., 2014b). Included effort measures are traditional HTER, translator usability ratings, keystroke counts, editing times, and two pause measures: APR and PWR. To examine the relationship between these various measures, we compute correlation between all measures. As we would like to use these measures to tune automatic metrics that can be used for system optimization, we use Spearman's ρ. Shown in Table 5.5, all measures tend to correlate with each other to some degree. Notably, HTER and keystroke have a correlation of 0.91, indicating that for this data, HTER is a very close approximation of actual editing effort. Further, a correlation of -0.84 between HTER and user rating indicates that translators are able to reliably assess the amount of editing they have just completed. As opposed to previous work that asks a translator how much post-editing is *expected* for a given MT output, TransCenter asks the translator how much post-editing *was* required *immediately* after the MT output is edited. Finally, keystroke count stands out as particularly indicative of overall effort; it is highly correlated with both HTER and user rating and has the highest correlation with editing time of any measure.

In a second experiment, we tune a version of Meteor to maximize correlation (Spearman's ρ) with each of these measures. Examining the optimal parameters provides insight into what types of translations require more or less effort to edit according to each measure. Shown in Table 5.6, we focus on the three most promising measures: HTER, Keystroke, and rating. One striking result is the focus on content words (δ) in rating parameters. This indicates that translators do not consider smoothing out grammatical errors to be nearly as significant as correcting mistranslations of important concepts. Also of note is that rating

[3]Due to technical difficulties encountered in the second round of post-editing (data loss due to Internet connectivity issues between users and our server), certain sentences had to be discarded for certain users. This resulted in a sufficient number of incomplete documents to prevent proper normalization and reporting HTER and rating results. However, a large number of individual data points remain and can be pooled with previously collected data for tasks that treat sentences as independent.

		BLEU	Meteor	Length
News	News-BLEU	**32.4**	34.2	**100.1**
	News-Meteor	30.9	**34.4**	105.6
TED	News-BLEU	31.6	34.0	92.2
	News-Meteor	31.7	34.2	96.3
	TED-BLEU	31.8	34.0	94.7
	TED-Meteor	**32.3**	**34.3**	**97.1**

Table 5.7: Metric scores for simulated post-editing experiments with task-specific metrics. Labels indicate tuning set and metric. Scores are averages over 3 optimizer runs.

	HTER	Rating	SPE BLEU
TED-BLEU	20.1	4.16	**27.3**
TED-Meteor	**18.9**	**4.24**	26.6

Table 5.8: Results for live post-editing experiments with task-specific metrics

parameters favor precision (α), which is actually contrary to annotators' natural preference for recall observed in evaluations such as WMT (Callison-Burch et al., 2012). Finally, the HTER parameters are far more extreme than those learned from GALE data: recall is preferred almost exclusively, the fragmentation penalty is harsher (γ), and stem and synonym matches are disregarded. The aggregated results of our metric experiments point to two central observations. First, parameters are specific to the *data set* as well as the type of edit measure. Second, within a task, several different measure types correlate highly with one another. Together, these observations point to a revised role for automatic metrics in adaptive machine translation. Rather than developing a single "post-editing" version of a metric, we can use the post-editing data that naturally arises from using adaptive systems to tune *task-specific* metrics specifically for use with these systems. This paradigm is described in the next section.

5.5 Post-Editing Experiments with Task-Specific Metrics

We incorporate task-specific metrics into our adaptive MT systems as follows. First, we build and deploy an adaptive MT system as described Chapter 3. This requires no post-editing data, using simulated post-editing and the BLEU metric during optimization and internal evaluation. Once this system is put into production, serving translations to actual human post-editors as described in Chapter 4, post-edited data is naturally created. Once a sufficient amount of data is collected, it can be used to tune a custom version of Meteor that is specific to the MT system and the domain of the data being post-edited. This system is then re-tuned using this version of Meteor and re-deployed. As the system continues to translate and adapt, the task-specific version of Meteor is used as the optimization target; BLEU is entirely removed from the system.

To evaluate this approach, we conduct both simulated and live post-editing experiments. We begin by selecting our task-specific metric: the version of Meteor tuned on keystroke data collected in our previous rounds of Spanish–English TED talk post-editing. This metric is then used to re-tune the fully adaptive system described in §3.3.1 (used in previous live post-editing experiments) and retained as the optimization target during decoding. The resulting Meteor-tuned system is evaluated against the BLEU-tuned baseline. To prevent the Meteor-tuned system from having an advantage simply from having access to more data, two variants of each system are evaluated: one tuned on standard WMT news data, and one tuned on the TED

BLEU	RW : I would restart –
Meteor (+)	RW : I would like to restart –
Reference	I 'd like to start
BLEU	and decided to make water
Meteor (+)	and I decided to make water
Reference	and I decided to make water .
BLEU	because it is funny – you know , this is a strange .
Meteor (+)	because it is funny – you know , is a strange thought .
Reference	because it 's a funny ... you know , it 's a strange thought ,

Table 5.9: Translations from BLEU and Meteor tuned systems that show the greatest difference when compared against pre-generated references. Adaptation is carried out on live post-editing data for both systems.

talk data that post-edited talk excerpts are drawn from.[4]

For the simulated evaluation, we compare Meteor and BLEU-tuned adaptive MT systems on both news and TED talk data. The "News" systems are optimized on the standard WMT11 news set and the "TED" systems are optimized on the first set of TED talks, from which all live post-editing data is drawn (TED1). Systems are evaluated on the standard WMT12 news test set and the entirely unseen TED2 set. Shown in Table 5.7, the Meteor-tuned system trades off BLEU for Meteor on news data while outperforming the BLEU-tuned system across both metrics on TED talks. Tuning to in-domain data is helpful in all cases. Notably, the Meteor-tuned system more closely fits the target length ratio, the same characteristic that causes a drop in BLEU score from domain mismatch in the news test. This is an example of a task-specific metric learning a relatively simple translation characteristic than can have a significant impact on quality. This provides evidence that custom-tuned automatic metrics can leverage a small amount of post-editing data to further bridge the gap in domain between a MT system's training data and the content it must translate in production.

In the live evaluation, we work with another set of 5 translators from Kent State University. We select excerpts from 10 talks in the second TED data set (TED2) totaling 200 sentences and conduct another round of post-editing with TransCenter using the same procedure as before (§4.3). We compare the two system variants tuned on TED talk data: "TED-BLEU" and "TED-Meteor". Following previous experiments, we also report the simulated post-editing BLEU score over just these excerpts. Shown in Table 5.8, the simulated BLEU score actually indicates a drop in performance when Meteor is used for optimization. However, data collected from TransCenter shows that Meteor-driven translations require less mechanical effort to correct and are preferred by post-editors. In addition to empirically demonstrating the benefit of using task-specific metrics in adaptive MT systems, these results provide a clear example of BLEU's inability to predict post-editing effort in live translation scenarios. In contrast, our metrics that focus on the post-editing task are able to successfully guide the MT system toward easier to edit translations.

We finally conduct the sentence level analysis described in §3.4.2, using a representative document to examine sentences that are translated substantially differently by the systems we are comparing. In this case, both systems are adaptive, one tuned to BLEU and the other Meteor. Since final translations are again produced by two different users, we score MT outputs against independent reference translations. We see a similar trend as in the previous round of experiments comparing static to adaptive systems: many sentences are significantly improved while a few are slightly degraded. Shown in Table 5.9, the sentences with the

[4]For TED talk tuning data, we use independently generated reference translations so that we can tune on the full development set rather than just the sentences that have post-edits. In a production scenario where no independent references exist, additional post-editing data can be collected to form a sufficiently large development set.

largest metric score differences are all improved. Degraded sentences are largely due to language variation or unseen phrases that neither system translates adequately. These results provide further evidence that tuning systems to a task-specific version of Meteor significantly reduces the amount of work required of human post-editors, even over other adaptive MT systems.

Chapter 6

Adaptive MT in Low-Resource Scenarios

Building effective machine translation systems for low resource languages is a significant and persistent challenge for governments and international organizations. As modern statistical MT systems depend on the availability of data, techniques that work well for language pairs with millions of bilingual sentences frequently break down when applied to language paris without this wealth of data. With minimal training data, the output of MT systems is often too poor to serve as a starting point for post-editing; translators spend far more effort attempting to decipher mostly broken MT than they would translating a sentence from scratch. We apply our fully adaptive MT paradigm to data sets from two low-resource language pairs to demonstrate the effectiveness of our techniques when training data is scarce.

6.1 Data

We focus on two low resource languages: Dari and Pashto. Dari is a native language of Afghanistan spoken by approximately 9.6 million people worldwide (Lewis et al., 2015). Pashto is a language used in both Afghanistan and Pakistan, spoken by 40-60 million people worldwide (Lewis et al., 2015). Despite the large number of speakers, these languages have relatively little bilingual text with English, making the development of statistical MT systems difficult. For each language, we have two sets of bilingual text: one collected by the United States Army Research Laboratory[1] (ARL), and one from the United States Government Catalog of Language Resources (GCLR) (Klavans, 2012). These data sets contain various texts including military manuals, field medicine guides, and other civilian training materials. As this data was aggregated from many sources with various levels of metadata annotation, genre and document boundaries are largely unknown. The only separation is between ARL and GCLR data for each language. The total number of bilingual sentences and monolingual words is shown in Table 6.1. This includes the same monolingual selection from the English Gigaword described in §1.4.

6.1.1 Simulated Document Sampling

To build and evaluate MT systems, we must divide the data for each language into training, dev, and test sets. This is frequently done by reserving the last several thousand sentences of a bitext for dev and test sets, a technique that works well enough when the bitext consists largely of the same genre, such as parliamentary proceedings or news articles. However, given that our data sets contain a variety of genres, simply reserving a single section would lead to a mismatch between training and evaluation data where the representation of different domains would be skewed from one set to another. The typical solution to this problem is uniform *sampling* over the bitext. For instance, if 1000 sentence dev and test sets were to be sampled from

[1] `http://www.arl.army.mil/www/default.cfm`

	Training Data	
	Bilingual (sents)	Monolingual (words)
Dari–English	221,967	50,709,888
Pashto–English	241,976	51,665,091

Table 6.1: Data set sizes for Dari and Pashto

	Dari–English				
	Dev	ARL1	ARL2	GCLR1	GCLR2
Baseline	*26.1*	33.2	32.1	22.3	20.6
Adaptive	*26.8*	**33.4**	**32.6**	**23.1**	**21.7**

	Pashto–English				
	Dev	ARL1	ARL2	GCLR1	GCLR2
Baseline	*22.2*	20.0	19.7	19.8	19.3
Adaptive	*22.9*	**20.6**	**20.7**	**21.4**	**20.5**

Table 6.2: BLEU scores for baseline and adaptive systems. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All adaptive systems show statistically significant improvement over baselines ($p < 0.05$ in approximate randomization).

a 100,000 line bitext, the bitext would be divided into groups of 100 lines. The first 98 lines would go into training, the next line to dev, and the next line to test. This would be repeated throughout the data to produce a training set of 98,000 sentences and dev and test sets of 1000 sentences each. While this works well for traditional MT scenarios, both human translators and our adaptive MT systems rely heavily on local document context. Techniques that allow humans and our systems to zero in on a target text such as a TED talk break down when sentences are drawn seemingly at random from a variety of texts. To address this, we simulate sampling whole documents rather than individual lines. The process is similar to the one described above, but occurs over *groups* of lines.

Simulated document sampling is conducted as follows. First, a document size N is determined (20 in our work). Next, the bitext is divided into simulated documents of size N (For example, a 100,000 line bitext would be divided into 5000 documents). Uniform sampling over *documents* is conducted to produce the desired dev and test set size. Following the above example, for every set of 100 documents, the first 98 would go into training, the next one to dev, and the next one to test. This would result in a training set of 4900 documents (98,000 sentences), and dev and test sets of 50 documents each (1000 sentences). Any remaining data at the end of the bitext that does not fit into a document rotation is added to the training set. This process produces significantly more coherent training and evaluation data provided that the bitext originated as a series of documents; while simulated documents are not guaranteed to map to actual documents, they can dramatically improve the amount of local coherence in dev and test sets over uniform sampling.

We conduct document sampling for each language to produce two ARL test sets of 2000 sentences each and two GCLR test sets of 2000 sentences each. We also reserve a dev set that consists of 1000 ARL sentences and 1000 GCLR sentences (2000 total). All other data is pooled together for system training.

Baseline	narcotics for Helmand police officials said .
Adaptive (+)	Police set fire to Helmand drugs .
Reference	Police set fire to Helmand drugs
Baseline	Director of Public Affairs : resolve road problems and transportation for thousands
Adaptive (+)	Public works chief : solving road , transport problems for thousands of people
Reference	Public works chief : solving road , transport problems for 1,000s
Baseline	Laghman officials to travel to the appropriate .
Adaptive (+)	Laghman officials travel to learn proper contract .
Reference	Laghman officials travel to learn proper contract procedures
Baseline	everything good planning .
Adaptive (+)	Everything in the well - organized .
Reference	Everything is well organized
Baseline	The national consultative peace decision
Adaptive (+)	The National Consultative Peace Jirga and the decisions
Reference	- Resolution , National Consultative Peace Jirga , Kabul
Baseline	I if wd ?
Adaptive (+)	Where did you wd ?
Reference	Where did you put it ?

Table 6.3: Examples of Dari (top) and Pashto (bottom) translations where adaptive systems yield significant improvement over baselines by learning domain-specific vocabulary and phrasing.

6.2 Experiments

We use the best performing version of our adaptive MT framework to build systems for both language pairs. This version, which uses the Moses toolkit (Koehn et al., 2007) and is described in §3.4, updates both the translation model and feature weights after each sentence is translated and uses an extended post-editing feature set. We use simulated post-editing to optimize and evaluate these systems, comparing them against static baselines. Shown in Table 6.2, the adaptive system outperforms the static system for every data set in each language pair. Gains range from 0.2 to 1.6 BLEU with Pashto results generally being stronger and GCLR data sets being better suited for adaptation. These results indicate that our techniques for rapidly adapting to post-editing data generalize to low-resource languages. While low resource MT still poses a significant challenge, we show strong evidence that using our adaptive systems would reduce the burden placed on human translators working in low resource domains.

To examine the differences between translations from static and adaptive systems, we again score the outputs for each evaluation set at the sentence level using the BLEU and Meteor metrics (Papineni et al., 2002; Denkowski and Lavie, 2011). While many sentences across both language pairs remain the same or nearly the same, three classes of differences stand out. First, many sentences change from incomprehensible to perfect. This is mainly due to the repetition of phrases throughout the evaluation documents and not to specific properties of low resource language pairs. The second class consists of sentences that are slightly degraded. Like with other language scenarios, this is largely a reduction and redistribution of noise in the translation model and many sentences that metrics consider degraded are simply equivalent translations that are worded differently. The final, most interesting class consists of sentences that improve dramatically but are still imperfect. Illustrated by the examples in Table 6.3, the original baseline systems have simply not seen the correct translation senses for many input words. This is a frequent problem in low resource MT as systems must learn from a very limited number of examples. Compared to high resource MT systems that struggle with the ambiguity problem of selecting the correct translation out of many possible candidates, low resource systems struggle with the sparsity problem of only having seen words or phrases translated in a limited number of contexts. When words appear in new contexts, systems make translation errors such as those in the example sentences. Our adaptive MT systems successfully learn new translation senses from incremental data, producing much larger pieces of correct translation that would significantly reduce the number of edits required. This further demonstrates that our systems would be helpful to human translators working in similar language scenarios.

Chapter 7

Conclusions and Future Work

7.1 Summary of Contributions

The following sections summarize the main points of each chapter, highlighting research contributions and key results. Full descriptions of each contribution as well as detailed analyses of results can be found in the respective chapter for each section. Background material can be found in Chapter 2.

7.1.1 Online Learning for Machine Translation

We cast machine translation for post-editing as an *online* learning task that proceeds as a series of trials. Each trial consists of three stages: (1) the model makes a prediction, (2) the model receives the "true" answer, and (3) the model updates its parameters. In post-editing workflows, these stages map to the system's producing an initial translation, a human post-editor's editing the MT output to produce a fully correct translation, and the system's using the new source-target sentence pair to update its models. We introduce three extensions to traditional translation systems that allow models to incorporate new data from post-editors in real time. These extensions include a translation grammar extraction algorithm that immediately incorporates new training instances, the practice of running an online optimizer during decoding, and an extended feature set that allows the translation model to leverage multiple sources of data. Combining these individual components results in a highly adaptive MT system that immediately learns from human feedback and can avoid making the same mistakes repeatedly.

Online Translation Grammar Adaptation: We extend the on-demand translation model described in §2.2.6 (Lopez, 2008a; Lopez, 2008b) to accept additional training data in the form of post-edited sentences (Denkowski et al., 2014a). When a translator edits a MT hypothesis, the sentence pair resulting from the input sentence and post-edited translation is word-aligned and aligned phrase pairs are then stored in a lookup table. When an on-demand grammar is extracted, phrase translations are sampled from both the background and post-editing data, resulting in both the addition of new translation rules and the refinement of feature scores for existing rules. An additional "post-edit support" indicator feature marks rules that are consistent with post-editor feedback. As the underlying translation model is hierarchical, it can also learn new non-local reordering patterns from post-editing data.

Online Parameter Optimization: In place of traditional corpus-level batch optimization (Och, 2003), we use the margin-infused relaxed algorithm described in §2.3.2 (Crammer et al., 2006a; Chiang et al., 2008; Eidelman, 2012). This online learning algorithm makes an adjustment to the weight vector after each sentence is translated. Optimization is still carried out at the corpus level, as MIRA makes a fixed number of passes over the development corpus to iteratively refine and initial weight vector. To better fit the online

	Spanish–English				English–Spanish			
	WMT11	WMT12	TED1	TED2	*WMT11*	WMT12	TED1	TED2
Baseline	*29.3*	31.6	34.0	30.2	*30.5*	30.9	27.0	26.5
Adaptive	*30.9*	**33.0**	36.1	**32.4**	*31.6*	**31.7**	**29.8**	**28.9**
	Arabic–English				English–Arabic			
	MT08	MT09	TED1	TED2	*MT08*	MT09	TED1	TED2
Baseline	*22.2*	26.0	11.2	11.5	*19.1*	23.7	7.8	8.7
Adaptive	*23.1*	**27.8**	**15.1**	**16.0**	*20.1*	**24.8**	**9.5**	**10.7**

Table 7.1: BLEU scores for baseline and adaptive systems. Reported scores are averages over three optimizer runs. Italics indicate scores on development (tuning) sets while bold numbers indicate highest scores on held-out test sets. All adaptive systems show statistically significant improvement over respective baselines ($p < 0.05$ in approximate randomization).

learning paradigm, we continue running the MIRA optimizer during decoding as new reference translations are made available through post-editing. This allows our systems to employ true sentence-level optimization in production.

Extended Post-Editing Feature Set: To overcome the limitations of relying on a single set of features to represent both background and post-editing data, we introduce an extended feature set that presents the decoder with more fine grained information about the likelihood of translation rules according to each data source. Three versions of each feature are computed: one using the union (aggregation) of background and post-editing data, one using only the background data, and one using only the post-editing data. Each feature is visible to the decoder and has an independent feature weight, effectively tripling the size of the phrase feature set and allowing the translation system to weigh the contribution of background versus post-editing data on a per-feature basis.

Evaluating Adaptive MT Systems: We evaluate our MT system extensions in all simulated post-editing scenarios outlined in §1.4, translating a mixture of language directions and domains that cover a broad range of difficulty levels. The greatest gains are realized by our fully adaptive system (updating both translation grammars and feature weights) with our extended feature set. Table 7.1 compares this adaptive system to a static baseline, showing significant improvement in all cases, especially for out-of-domain sets.

7.1.2 Live Post-Editing Evaluation: Software and Experiments

The most important measure of efficacy for our systems is the ultimate impact on human productivity in live translation scenarios. Measuring human productivity requires conducting live post-editing experiments, which in turn require an interface between translators and our MT systems. We have developed a lightweight web-based translation editing environment called *TransCenter* in which users translate documents by post-editing MT output served by our adaptive systems (Denkowski and Lavie, 2012b; Denkowski et al., 2014b). As translators work, the adaptive systems learn from their post-edits in real time and TransCenter records all user activity. This forms an end-to-end translation and post-editing pipeline that is used to evaluate our adaptive MT systems.

TransCenter: Post-Editing User Interface: Our software uses a simple two column interface that translators are familiar with. The left column displays the source sentences while the right column is incrementally populated with translations from one of our MT systems as the user works. For each sentence, the translator

	HTER	Rating
Baseline	19.26	4.19
Adaptive	**17.01**	**4.31**

Table 7.2: Aggregate simulated post-editing BLEU scores, HTER scores, and average translator self-ratings (5 point scale) of post-editing effort for translations of TED talks from Spanish into English.

is asked to edit the MT output to be grammatically correct and convey the same information as the source sentence. After editing, the final translation is archived and (if the system is adaptive) fed back to the MT system for learning. The next sentence is then machine translated and post-edited. The user is additionally asked to rate the amount of work required to post-edit each sentence immediately after completing it, using a scale that ranges from 5 (no post-editing required) to 1 (requires total re-translation).

Data Collection: In addition to gathering final edited translations and user ratings, TransCenter records all user interaction at a level of detail sufficient to replay the entire post-editing session. This includes number of keystrokes, number of milliseconds each sentence is focused, and a millisecond-timestamped record of each individual keystroke. Our software uses this information to generate several reports measuring of human effort.

Live Post-Editing Experiments: Connecting TransCenter to our MT systems forms a complete post-editing pipeline that enables us to run live evaluations to measure the effect of our online model adaptation techniques on human productivity. These experiments are conducted in collaboration with Kent State University's Institute for Applied Linguistics[1], an academic institution for training professional translators. We establish an experimental setup wherein observing a pool of human translators can determine which of two MT systems is better for post-editing. In the first round of live experiments, we compare the static baseline system described in §1.4 to an adaptive system that updates both translation grammar and weights after each sentence. Shown in Table 7.2, we see a significant improvement in HTER and a slight user preference. This provides evidence that gains in simulated scenarios can translate to gains in actual post-editing.

7.1.3 Automatic Metrics of Post-Editing Effort: Optimization and Evaluation

As a complement to translation models that incorporate post-editing feedback, we develop automatic metrics that more accurately evaluate the amount of work required to edit translation hypotheses. Pairing online models with automatic post-editing metrics enables end-to-end translation systems that target human translation in both the model estimation and optimization phrases.

The Meteor Metric: Meteor is an automatic evaluation metric that scores MT hypotheses by aligning them to reference translations (Denkowski and Lavie, 2011). Alignments are based on several types of flexible matches that go beyond surface forms to identify word and phrase correspondences that would be clear to humans but are missed by standard metrics. These include word stem, synonym, and paraphrase matches as well as a distinction between matched content and function words. Once matches are identified, the final alignment is resolved as the largest subset of non-overlapping matches across both sentences. Weighted precision and recall values are combined with a fragmentation penalty (measuring translation gaps and re-ordering) in a parameterized function to produce the final metric score. Each of the Meteor scoring statistics can be interpreted as a key predictor of post-editing effort and the scoring parameters can be tuned to maximize agreement between Meteor scores and human assessments of translation quality. Optimization is

[1] `http://appling.kent.edu/`

	HTER	Rating
BLEU-tuned	20.1	4.16
Meteor-tuned	**18.9**	**4.24**

Table 7.3: Results for live post-editing experiments with task-specific metrics

conducted as an exhaustive parametric sweep over feasible parameter values to maximize correlation with human judgments.

Improved Editing Measures for Improved Metrics: We use the post-editing data collected by TransCenter to explore a range of possible measures of human editing effort. Notably, HTER and keystroke have extremely high correlation, indicating that HTER is a reliable approximation of actual editing effort. Further, a high correlation between HTER and user rating indicates that translators are able to reliably assess the amount of editing they have just completed. Finally, keystroke count stands out as particularly indicative of overall effort, being highly correlated with both HTER and user rating.

In a second experiment, we tune a version of Meteor to maximize correlation with each of these measures and examine the optimal parameters. One striking result is the focus on content words in rating parameters, indicating that translators do not consider smoothing out grammatical errors to be nearly as significant as correcting mistranslations of important concepts. The rating parameters also favor precision, which is contrary to annotators' natural preference for recall (Callison-Burch et al., 2012). Finally, the HTER parameters are far more extreme than those learned from other post-editing data (Denkowski and Lavie, 2011). These results demonstrate that parameters are specific to the *data set* as well as the type of edit measure, pointing to a revised role for automatic metrics in adaptive machine translation. Rather than developing a single "post-editing" version of a metric, we can use the post-editing data that naturally arises from using adaptive systems to tune *task-specific* metrics specifically for use with these systems.

Post-Editing Experiments with Task-Specific Metrics: We incorporate task-specific metrics into our adaptive MT systems as follows. First, we build and deploy an adaptive MT system using simulated post-editing and the BLEU metric. Once this system is put into production, we collect a sufficient amount of data to tune a custom version of Meteor (specific to the system and translation domain). The MT system is re-tuned using the targeted version of Meteor and re-deployed. As the system continues to translate and adapt, the task-specific version of Meteor is used as the optimization target instead of BLEU.

We evaluate this approach with another set of post-editing experiments. We select the version of Meteor tuned on keystroke data collected in our previous rounds of Spanish–English TED talk experiments. This metric is used to re-tune the fully adaptive system and act as the optimization target during decoding. The resulting Meteor-tuned system is evaluated against the BLEU-tuned adaptive system in another set of live experiments. Shown in Table 7.3, Meteor-driven translations require less mechanical effort to correct and are preferred by post-editors.

7.2 Future Research Directions

In previous chapters, we have described several extensions to traditional machine translation systems that facilitate the incorporation of new data that is incrementally available, such as that generated from post-editing. The end-to-end translation post-editing workflow we have introduced aims to be as general as possible, serving as a platform on which to build future generations of systems for this task. In the following sections, we present what we believe to be the most promising next steps for each of the areas we have discussed. We conclude by discussing the role adaptive MT and other CAT technology could play in large scale translation projects in the next several years.

7.2.1 Adaptive Machine Translation

Our adaptive MT systems begin with the simplest possible extensions to current phrase-based models: adding new data to the bitext from which translation grammars are sampled and updating feature weights after each sentence is edited. The additions of a post-edit support feature and later an extended post-editing feature set yield significant improvement in both simulated and live evaluations. However, these systems still employ relatively small feature sets and standard optimizers. Recent work in the MT and machine learning communities has led to advances in both of these areas, though they have not yet been applied to post-editing. Given the general, modular nature of our adaptive systems, incorporating both expanded feature sets and new optimizers would be relatively straightforward.

Sparse Features: In contrast to *dense* features that score each translation rule, *sparse* features only score rules when applicable. Sparse feature sets are typically orders of magnitude larger than dense feature sets as they are far more specialized. We propose adding two common classes of sparse features to our systems: phrase and word translation features. Each phrase translation feature is specific to a rule in the translation grammar; when that rule is used by the decoder, the count for the sparse feature corresponding to that rule is incremented (Dyer et al., 2010). Word translation features work similarly at the lexical level. Within translation rules, word alignments are used to count the number of instances in which each source word translates to each target word (Chiang et al., 2009). Additional features use unaligned words to track insertions and deletions. Each phrase or word translation feature is independent, having its own weight that is optimized during system tuning.

Phrase and word translation features can be seen as phrase and lexical translation scores that are learned from the development corpus rather than the large bilingual training corpus. Rather than presenting scores learned from aggregate counts from all available data, they present raw counts, relying on feature weights to form the scoring function. In practice, using both dense and sparse features allows the optimizer to re-tune the scores for various phrase or lexical translations for the target domain. In the case of adaptive MT, a sparse feature set allows much more rapid adaptation to human feedback; individual word or phrase translations can be learned or unlearned based on the feedback of a single sentence. However, sparse features also introduce a significant risk of overfitting as they are effectively estimated from a much smaller amount of data. Greater responsibility for effectively targeting the post-editing data without overfitting is now placed on the optimizer.

Optimizers: MT systems are typically optimized with batch learning algorithms such as minimum error rate training that only scale to a few dozen features (Och, 2003). Even MIRA, which can handle hundreds of thousands of features, struggles with *diverse* feature sets that contain a mix of dense and sparse features (Crammer et al., 2006a). Dense features are seen for every translation rule while sparse features occur rarely. If all features are updated at the same rate during optimization, the weights learned for sparse features will be less accurate, as they are based on far fewer observations. Duchi et al. (2011) present methods for handling this diversity by incorporating knowledge of the geometry of the data observed in previous iterations to identify and set learning rates for very predictive but rarely seen features. These techniques have had good results for sparse feature sets in standard MT setups (Green et al., 2013). We expect them to be particularly useful here, where incremental post-edited data is very sparse.

Standard MT learning algorithms also assume that training examples are independent and identically distributed. This assumption is generally not harmful when sentences in development corpora are drawn from a single larger data set. However, in post-editing, we have reason to believe that more recent sentences are more informative when making updates: within documents, topics tend to be discussed sequentially with repeated terminology and phrasing. Rather than favoring nearby sentences in optimization, traditional methods used for online learning in MT such as stochastic gradient descent tend to *discount* recent training examples compared to earlier instances (Bertsekas, 2003). Recently proposed algorithms in the machine learning community relax the independence assumptions placed on training data. Nesterov (2005) originally describes methods that use a weighted average of historical gradients that upweights the contribution of more recent gradients. Xiao (2010) further develops these techniques, taking regularization into account. However, these recency-favoring algorithms still lack the crucial per-feature learning rates discussed above. Zeiler (2012) proposes an algorithm termed *AdaDelta* that incorporates both recent instance upweighting and per-feature learning. While it has only been evaluated on traditional data sets, we expect it to be a good fit for learning from MT post-editing.

7.2.2 Post-Editing Interfaces

Our live post-editing experiments have focused on improving the quality of the machine translation served to human translators. Our TransCenter interface is intentionally simple so that collected data can be maximally accurate for analyzing the interaction between users and our systems. In practice, translators use rich translation editing suites such as those developed in the large CASMACAT and MateCat projects (Ortiz-Martínez et al., 2012; Federico, 2014). Again, given the modular nature of our systems, it would be relatively straightforward to plug our adaptive MT systems into any of the rich human translation interfaces under active development. Two promising translation interfaces are the streamlined interface developed by Green (2014) and the monolingual interface developed by Schwartz (2014).

Streamlined Post-Editing: This interface simplifies the translation environment into a single column that alternates between source text and target translations. A variety of tools are available directly from the keyboard, such as next phrase prediction and full post-editing (filling in the rest of the sentence). Human translators report that phrase prediction is particularly helpful, allowing them to create their own translations with the aid of MT rather than simply correcting the errors of an automatic system (Green, 2014). This requires the ability to rapidly re-translate the source sentence, keeping the already translated portion fixed (a process called prefix decoding). The underlying MT toolkits we use to implement our adaptive systems do not support prefix decoding, leaving one of two choices for integration with this interface. First, prefix decoding can be simulated using an output lattice from an MT system (Ortiz-Martínez et al., 2012). This approximation sacrifices accuracy but is substantially easier to implement as an initial feasibility test. The second longer term option consists of either implementing prefix decoding in our toolkit or online model adaptation in a toolkit that supports prefix decoding. We believe that either would provide an improvement over using a static MT system.

Monolingual Post-Editing: This interface presents monolingual editors (speaking only the target language) with source text, MT output, and word alignments between source and target words. Schwartz (2014) reports that monolingual domain experts can use this information to post-edit MT output with great success. As this interface operates on full sentences, our adaptive MT systems could supply the translations and learn from the post-edited references. We believe this combination could significantly empower monolingual experts to translate material into their languages.

7.2.3 Automatic Metrics

Our work on the Meteor metric has shown that adaptive MT systems can improve significantly when tuned to task-specific metrics. Once a necessary amount of post-edits are collected, a version of Meteor can be tuned to fit editing effort for a system, domain, and pool of translators. In our work, we measure mechanical effort in the form of keystrokes required to edit translations. There are two additional editing measures we believe warrant exploration: editing time and translator pauses.

Editing Time: This measure is often associated with "bottom line" cost of translation. Total translation time can be used to compute the cost of localizing a document for company that employs salaried translators or to compute the hourly income of a freelance translator paid on a per-word basis. However, translation time is an incredibly difficult measure to predict (fit with an automatic evaluation metric). Beyond more experienced translators' tendency to work faster, our work has found high variance in translation time even between individuals with similar levels of experience. We have also been unable to find reliable links in our data between translation time, amount of editing, and quality of final post-edited translation. The best way to collect and normalize timing data remains an open research question. One possible solution would be to move to a ranking approach, wherein MT outputs for each post-editor are ranked from slowest to fastest to edit (seconds per word) and divided into a number of bins (ranks). This would effectively normalize speed by post-editor.

Translator Pauses: Recent work in the translation studies community uses pauses in translator activity as a measure of cognitive effort (Lacruz et al., 2012; Lacruz and Shreve, 2014a). Mechanical measures such as keystroke and HTER do not take into account cognitive activity such as reading the source sentence, scanning the MT output, or taking long pauses to consider translation options. TransCenter is capable of computing two measures: average pause ratio (APR) and pause to word ratio (PWR). In a possible future direction, pause information could be combined with mechanical edit logs to form a new measure that takes into account both cognitive and mechanical effort in post-editing.

7.2.4 The Future of Adaptive MT and CAT Tools

In this work we have described an end-to-end human translation pipeline consisting of a fully adaptive MT system paired with a task-specific optimization function and a simple post-editing interface in the form of TransCenter. Following the most promising threads of possible future work would enable an improved MT system with sparse features and task-appropriate optimization. This system could serve translations to either of the advanced post-editing interfaces described above, combining the strengths of all work discussed into a state of the art, production ready computer-aided translation pipeline. As system building would only require standard MT training data and much of the runtime technology is already able to operate over an Internet connection, this pipeline could be deployed in virtually any translation scenario from large commercial enterprises to community volunteer projects. While results from current work indicate that this would lead to dramatic reductions in the amount of human labor required to produce publishable translations, two significant challenges must be addressed before the technology can see widespread adoption: integration and scalability.

Integration with Production Workflows: Our work to date has focused on the core translation technology in post-editing workflows. As such, each of our experiments has employed a pool of translators to edit plain text translations of pre-selected documents. In production scenarios, a variety of *formatted* content must be translated: businesses maintain web pages and author content in Microsoft Word while volunteer projects such as Wikipedia annotate text with various markup languages. All of these formats fall into the class of *tagged* text, plain text annotated with various markup tags that contain information about how that text should be formatted. As each input sentence may contain a mix of content text and markup tags, a CAT pipeline must be able to carry out the following process described by Hudík and Ruopp (2011): (1) identify and remove markup tags from the input sentence, (2) translate the now plain text input sentence, and (3) use information from the translation derivation to reinsert markup tags in the MT output. While this functionality is not directly implemented in our adaptive systems, the underlying Moses toolkit includes all the necessary components to carry out this process.

The more significant implementation challenge consists of integrating our systems with *project management* software commonly used in the translation industry. This software is used to set up, distribute, and track large translation tasks that involve sending large numbers of documents to several translators. In order for our systems to be linked with such software, the software must support real time sentence-level communication for online learning; documents cannot simply be distributed to translators and translations collected upon completion. This requires additional implementation work on the side of commercially developed project management software, which could present significant challenges.

Scaling to Teams of Translators: The second significant challenge is one of scalability. The underlying technology in the Moses toolkit is already highly scalable: multi-threaded decoding with memory mapped models scales to any number of available processors. Very large tasks can also be distributed across multiple machines using large scale parallelization. For adaptive MT systems, the challenge in scalability centers on dealing with many *translators* rather than many machines. In our work to date, each document is treated as independent. When a translator begins working, the MT system starts with an empty context (no incremental post-editing data). The model learns as the translator works, but is reset at the end of the document. When many translators are working together to translate many documents, such as for large commercial or community projects, treating each translator-document instance as independent would discard large amounts of highly relevant data. Ideally, an adaptive system would share relevant data between users and documents, learning from all available information sources.

This challenge could be met with a direct extension of our current work: further expanding our feature set to incorporate information from an arbitrary number of sources. Rather than keeping three versions of the translation model (background, incremental, and union), our systems could be easily extended to contain a version for each data source. For example, a system providing translations for a post-editor working on a news article could access feature scores computed on data from other news articles in the same domain, other news articles in different domains, other news articles translated by other post-editors, and so on. Each model could also distinguish between translated and post-edited data in the same way as our original background-incremental-union distinction. An appropriate optimizer could learn how much to trust each data source as the post-editor works. Several instances of this system could run at once, each serving translations and sharing new post-editing data as it is available. All of this functionality could be enabled with minimal additional implementation in the latest version of our adaptive MT software.

Appendices

Appendix A

Released Software and Data

The following software tools and data are released to the community under open source licenses. They can be freely used to replicate the results described in this work or as components in other machine translation and post-editing pipelines.

`cdec` Realtime

License:	Apache License
Tutorial:	`http://www.cs.cmu.edu/~mdenkows/cdec-realtime.html`
Source code:	`https://github.com/redpony/cdec/tree/master/realtime`

`cdec` Realtime is an implementation of our online translation model adaptation techniques using the `cdec` MT toolkit. It includes tools for building and deploying adaptive systems that learn from human feedback, using incremental post-editing data to update translation models and feature weights. These systems correspond to those described in §3.3.1 and used in our live translation experiments in Chapters 4 and 5.

Real Time Adaptive Machine Translation (RTA)

License:	*License pending*
Tutorial:	*Release pending*
Source code:	*Release pending*

RTA is an implementation of our best performing adaptive MT systems using components from both the Moses and `cdec` toolkits. It includes tools for building and deploying online systems with the extended post-editing feature set described in §3.4. A release of this software is pending.

TransCenter

License:	GNU Lesser General Public License (LGPL)
Tutorial:	Included in `README.md`
Source code:	`https://github.com/mjdenkowski/transcenter-live`

TransCenter is a web-based post-editing interface used to connect human translators to adaptive MT systems. User activity is logged at a level of detail sufficient to replay the entire post-editing session for deeper analysis. We use TransCenter to both evaluate our MT systems and collect post-editing data for metric development (Chapters 4 and 5).

Meteor

License:	GNU Lesser General Public License (LGPL)
Tutorial:	`www.cs.cmu.edu/~alavie/METEOR/`
Source code:	`https://github.com/mjdenkowski/meteor`

Meteor is a tunable, alignment-based automatic MT evaluation metric with high levels of correlation with human judgments of translation quality. Meteor can be used to evaluate MT output for a variety of default tasks or tuned to fit a specific set of quality assessments, such as mechanical effort to post-edit MT output. We use Meteor for our task-specific metric experiments in Chapter 5.

Kent State Post-Editing Data

License:	Public Domain
Download:	`http://www.cs.cmu.edu/~mdenkows/download/kent-data.tar.gz`

This archive contains all data collected from TransCenter in our live translation experiments (Chapters 4 and 5). It is distributed in the form of full TransCenter reports that contain source text, initial MT output, post-edited MT output, and a variety of measures of editing effort. Full edit traces are also included for further analysis.

References

[Agarwal and Lavie2008] Abhaya Agarwal and Alon Lavie. 2008. Meteor, M-BLEU and M-TER: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 115–118, Columbus, Ohio, June. Association for Computational Linguistics.

[Alabau et al.2012] Vicent Alabau, Luis A. Leiva, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2012. User evaluation of interactive machine translation systems. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 20–23.

[Alabau et al.2013] Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González, Luis Leiva, Bartolomé Mesa-lao, Daniel Ortiz, et al. 2013. Advanced computer aided translation with a web-based workbench. In *2nd Workshop on Post-Editing Technologies and Practice*. Citeseer.

[Aziz et al.2012] Wilker Aziz, Sheila Castilho Monteiro de Sousa, and Lucia Specia. 2012. PET: a tool for post-editing and assessing machine translation. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*.

[Banerjee and Lavie2005] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Bannard and Callison-Burch2005] Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Bertoldi et al.2013] Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2013. Cache-based online adaptation for machine translation enhanced computer assisted translation. In *Proceedings of the XIV Machine Translation Summit*, pages 35–42.

[Bertsekas2003] Dimitri P. Bertsekas. 2003. *Convex Analysis and Optimization*. Athena Scientific.

[Blain et al.2011] Frédéric Blain, Jean Senellart, Holger Schwenk, Mirko Plitt, and Johann Roturier. 2011. Qualitative analysis of post-editing for high quality machine translation. In *Proceedings of the twelfth Machine Translation Summit International Association for Machine Translation*.

[Bojar et al.2013] Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, August.

[Brown et al.1993] Peter Brown, Vincent Della Pietra, Stephen Della Pietra, and Robert Mercer. 1993. The mathematics of statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

[Brown1996] Ralf D. Brown. 1996. Example-based machine translation in the pangloss system. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 169–174.

[Brown1999] Ralf Brown. 1999. Adding linguistic knowledge to a lexical example-based translation system. In *Proceedings of the Eighth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-99)*, pages 22–32, August.

[Brown2004] Ralf D. Brown. 2004. A modified burrows-wheeler transform for highly-scalable example-based translation. In *Machine Translation: From Real Users to Research, Proceedings of the 6th Conference of the Association for Machine Translation*, pages 27–36, September/October.

[Callison-Burch et al.2005] Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 255–262, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Callison-Burch et al.2006] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

[Callison-Burch et al.2007] Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June. Association for Computational Linguistics.

[Callison-Burch et al.2010] Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics-MATR*, pages 17–53, Uppsala, Sweden, July. Association for Computational Linguistics. Revised August 2010.

[Callison-Burch et al.2011] Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Callison-Burch et al.2012] Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June. Association for Computational Linguistics.

[Carl et al.2011] Michael Carl, Barbara Dragsted, Jakob Elming, Daniel Hardt, and Arnt Lykke Jakobsen. 2011. The process of post-editing: A pilot study. *Copenhagen Studies in Language*, 41:131–142.

[Carl1999] Michael Carl. 1999. Inducing translation templates for example-based machine translation. In *Proceedings of the Seventh Machine Translation Summit (MT-Summit VII)*, pages 250–258.

[Casacuberta et al.2014] Francisco Casacuberta, Marcello Federico, and Philipp Koehn, editors. 2014. *AMTA 2014 Workshop on Interactive and Adaptive Machine Translation (IAMT 2014)*, Vancouver, Canada, October. Association for Machine Translation in the Americas (AMTA).

[Cattelan2014] Alessandro Cattelan. 2014. Third version of MateCat tool. Deliverable 4.3, October.

[Cer et al.2010] Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. 2010. The best lexical metric for phrase-based statistical mt system optimization. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 555–563, Los Angeles, California, June. Association for Computational Linguistics.

[Cettolo et al.2012] Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the Sixteenth Annual Conference of the European Association for Machine Translation*.

[Chahuneau et al.2012] Victor Chahuneau, Noah A. Smith, and Chris Dyer. 2012. pycdec: A python interface to cdec. *The Prague Bulletin of Mathematical Linguistics*, 98:51–61.

[Chen and Goodman1996] Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA, June. Association for Computational Linguistics.

[Cherry and Foster2012] Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, June.

[Chiang et al.2008] David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii, October. Association for Computational Linguistics.

[Chiang et al.2009] David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33.

[Cicekli and Güvenir2000] Ilyas Cicekli and H. Altay Güvenir. 2000. Learning translation templates from bilingual translation examples. *Applied Intelligence*, pages 57–76.

[Clark et al.2011a] Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011a. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Clark et al.2011b] Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011b. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June. Association for Computational Linguistics.

[Clark2015] Jonathan Clark. 2015. Locally non-linear learning via feature induction and structured regularization in statistical machine translation. In *Dissertation, Carnegie Mellon University*, April.

[Crammer and Singer2003] Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

[Crammer et al.2006a] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006a. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, pages 551–558, March.

[Crammer et al.2006b] Koby Crammer, Ofer Dekel, Shai Shalev-shwartz, and Yoram Singer. 2006b. Online passiveaggressive algorithms. *Journal of Machine Learning Research*, 7:551—-585.

[Daume III2007] Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

[Denkowski and Lavie2010a] Michael Denkowski and Alon Lavie. 2010a. Choosing the right evaluation for machine translation: an examination of annotator and automatic metric performance on human judgment tasks. In *Proceedings of the Ninth Biennial Conference of the Association for Machine Translation in the Americas*.

[Denkowski and Lavie2010b] Michael Denkowski and Alon Lavie. 2010b. METEOR-NEXT and the METEOR paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 339–342, Uppsala, Sweden, July. Association for Computational Linguistics.

[Denkowski and Lavie2011] Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Denkowski and Lavie2012a] Michael Denkowski and Alon Lavie. 2012a. Challenges in predicting machine translation utility for human post-editors. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

[Denkowski and Lavie2012b] Michael Denkowski and Alon Lavie. 2012b. TransCenter: Web-based translation research suite. In *AMTA 2012 Workshop on Post-Editing Technology and Practice Demo Session*.

[Denkowski et al.2014a] Michael Denkowski, Chris Dyer, and Alon Lavie. 2014a. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden, April. Association for Computational Linguistics.

[Denkowski et al.2014b] Michael Denkowski, Alon Lavie, Isabel Lacruz, and Chris Dyer. 2014b. Real time adaptive machine translation for post-editing with cdec and TransCenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 72–77, Gothenburg, Sweden, April. Association for Computational Linguistics.

[DePalma and Sargent2013] Donald A. DePalma and Benjamin B. Sargent. 2013. *Transformative Translation*. Common Sense Advisory, Inc., September.

[DePalma et al.2014] Donald A. DePalma, Vijayalaxmi Hegde, Hélène Pielmeier, and Robert G. Stewart. 2014. *The Language Services Market: 2014*. Common Sense Advisory, Inc., June.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. 12:2121–2159, July.

[Dyer et al.2010] Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July. Association for Computational Linguistics.

[Dyer et al.2013] Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[Eidelman2012] Vladimir Eidelman. 2012. Optimization strategies for online large-margin learning in machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 480–489, Montréal, Canada, June. Association for Computational Linguistics.

[Federico2014] Marcello Federico. 2014. Machine translation enhanced computer assisted translation. Project Periodic Report, June.

[Galley et al.2004] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

[Gao and Vogel2008] Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57. Association for Computational Linguistics.

[Germann2014] Ulrich Germann. 2014. Dynamic phrase tables for machine translation in an interactive post-editing scenario. In *Proceedings of the AMTA 2014 Workshop on Interactive and Adaptive Machine Translation*, pages 20–31, October.

[Green et al.2013] Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 311–321, Sofia, Bulgaria, August. Association for Computational Linguistics.

[Green2014] Spence Green. 2014. Mixed-initiative natural language translation. In *Dissertation, Stanford University*.

[Guerberof2009] Ana Guerberof. 2009. Productivity and quality in mt post-editing. In *Proceedings of MT Summit XII - Workshop: Beyond Translation Memories: New Tools for Translators MT*.

[Habash et al.2009] Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

[Hardmeier et al.2012] Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Tree kernels for machine translation quality estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 109–113, Montréal, Canada, June. Association for Computational Linguistics.

[Hardt and Elming2010] Daniel Hardt and Jakob Elming. 2010. Incremental re-training for post-editing smt. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

[He et al.2010] Yifan He, Yanjun Ma, Johann Roturier, Andy Way, and Josef van Genabith. 2010. Improving the post-editing experience using translation recommendation: A user study. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

[Heafield and Lavie2011] Kenneth Heafield and Alon Lavie. 2011. CMU system combination in WMT 2011. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 145–151, Edinburgh, Scotland, United Kingdom, 7.

[Heafield et al.2013] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August.

[Heafield2011] Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July. Association for Computational Linguistics.

[Hudík and Ruopp2011] Tomáš Hudík and Achim Ruopp. 2011. The integration of moses into localization industry. In *15th Annual Conference of the EAMT*, pages 47–53.

[Klavans2012] Judith Klavans. 2012. Government catalog of language resources (gclr). In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

[Knight1999] Kevin Knight. 1999. A statistical MT tutorial workbook, August. Unpublished.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL/HLT 2003*.

[Koehn et al.2005] Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of the 2005 International Workshop on Spoken Language Translation*.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

[Koehn2012] Philipp Koehn. 2012. Computer-aided translation. Machine Translation Marathon.

[Koponen2012] Maarit Koponen. 2012. Comparing human perceptions of post-editing effort with post-editing operations. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 181–190, Montréal, Canada, June. Association for Computational Linguistics.

[Lacruz and Shreve2014a] Isabel Lacruz and Gregory M. Shreve. 2014a. Pauses and cognitive effort in post-editing. In Sharon O'Brien, Laura Winther Balling, Michael Carl, Michel Simard, and Lucia Specia, editors, *Post-editing of Machine Translation: Processes and Applications*.

[Lacruz and Shreve2014b] Isabel Lacruz and Gregory M. Shreve, 2014b. *Pauses and Cognitive Effort in Post-Editing*. Cambridge Scholars Publishing, march.

[Lacruz et al.2012] Isabel Lacruz, Gregory M. Shreve, and Erik Angelone. 2012. Average Pause Ratio as an Indicator of Cognitive Effort in Post-Editing: A Case Study. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 21–30, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[Lavie and Agarwal2007] Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June. Association for Computational Linguistics.

[Lavie and Denkowski2009] Alon Lavie and Michael Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23.

[Lavie et al.2008] Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 87–95, Columbus, Ohio, June. Association for Computational Linguistics.

[LDC2005] LDC. 2005. Linguistic Data Annotation Specification: Assessment of Fluency and Adequacy in Translations. Revision 1.5.

[Levenberg et al.2010] Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, California, June. Association for Computational Linguistics.

[Lewis et al.2015] M. Paul Lewis, Gary F. Simons, and Charles D. Fennig. 2015. Ethnologue: Languages of the world, eighteenth edition. Online version.

[Liu et al.2006] Yang (1) Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

[Liu et al.2009] Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566, Suntec, Singapore, August. Association for Computational Linguistics.

[Liu et al.2011] Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2011. Better evaluation metrics lead to better machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 375–384, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

[Lopez2008a] Adam Lopez. 2008a. Machine translation by pattern matching. In *Dissertation, University of Maryland*, March.

[Lopez2008b] Adam Lopez. 2008b. Tera-scale translation models via pattern matching. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 505–512, Manchester, UK, August. Coling 2008 Organizing Committee.

[López-Salcedo et al.2012] Francisco-Javier López-Salcedo, Germán Sanchis-Trilles, and Francisco Casacuberta. 2012. Online learning of log-linear weights in interactive machine translation. *Advances in Speech and Language Technologies for Iberian Languages*, pages 277–286.

[Maarit Koponen and Specia2012] Luciana Ramos Maarit Koponen, Wilker Aziz and Lucia Specia. 2012. Post-editing time as a measure of cognitive effort . In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 11–20, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[Macherey et al.2008] Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, Hawaii, October. Association for Computational Linguistics.

[Manber and Myers1993] Udi Manber and Gene Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing*, 22:935–948.

[Martínez-Gómez et al.2012] Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. 2012. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45:3193–3203.

[Miller and Fellbaum2007] George Miller and Christiane Fellbaum. 2007. WordNet. http://wordnet.princeton.edu/.

[Moran et al.2014] John Moran, Christian Saam, and Dave Lewis. 2014. Towards desktop-based cat tool instrumentation. In *Proceedings of the The Third Workshop on Post-editing Technology and Practice*, page 99, October.

[Nepveu et al.2004] Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George Foster. 2004. Adaptive language and translation models for interactive machine translation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 190–197, Barcelona, Spain, July. Association for Computational Linguistics.

[Nesterov2005] Yu Nesterov. 2005. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, May.

[NISHIDA et al.1988] Fujio NISHIDA, Shinobu TAKAMATSU, Tadaaki TANI, and Tsunehisa DOI. 1988. Feedback of correcting information in postediting to a machine translation system. In *Proceedings of the International Conference on Computational Linguistics*.

[Noreen1989] Eric W. Noreen. 1989. *Computer intensive methods for testing hypotheses*.

[O'Brien et al.2012] Sharon O'Brien, Michel Simard, and Lucia Specia, editors. 2012. *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[O'Brien et al.2014] Sharon O'Brien, Michel Simard, Lucia Specia, and Joss Moorkens, editors. 2014. *AMTA 2014 Workshop on Post-Editing Technology and Practice (WPTP 2014)*, Vancouver, Canada, October. Association for Machine Translation in the Americas (AMTA).

[Och and Ney2002] Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29.

[Och and Ney2004] Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. 30.

[Och et al.1999] Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

[Olive et al.2011] Joseph Olive, Caitlin Christianson, and John McCary, editors. 2011. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer.

[Ortiz-Martínez et al.2010] Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 546–554, Los Angeles, California, June. Association for Computational Linguistics.

[Ortiz-Martínez et al.2012] Daniel Ortiz-Martínez, Germán Sanchis-Trilles, Francisco Casacuberta, Vicent Alabau, Enrique Vidal, José-Miguel Benedí, Jesús González-Rubio, Alberto Sanchis, and Jorge González. 2012. The CASMACAT project: The next generation translator's workbench. In *Proceedings of the 7th Jornadas en Tecnología del Habla and the 3rd Iberian SLTech Workshop (IberSPEECH)*, pages 326–334.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

[Parker et al.2011] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition, June. Linguistic Data Consortium, LDC2011T07.

[Pearson1895] Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242.

[Penkale and Way2012] Sergio Penkale and Andy Way. 2012. SmartMATE: An Online End-To-End MT Post-Editing Framework. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 51–59, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[Porter2001] Martin Porter. 2001. Snowball: A language for stemming algorithms. http://snowball.tartarus.org/texts/.

[Poulis and Kolovratnik2012] Alexandros Poulis and David Kolovratnik. 2012. To post-edit or not to post-edit? Estimating the benefits of MT post-editing for a European organization. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 60–68, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[Przybocki2009] Mark Przybocki. 2009. Nist open machine translation 2009 evaluation. http://www.itl.nist.gov/iad/mig/tests/mt/2009/.

[Przybocki2012] Mark Przybocki. 2012. Nist open machine translation 2012 evaluation (openmt12). http://www.nist.gov/itl/iad/mig/openmt12.cfm.

[Saluja et al.2012] Avneesh Saluja, Ian Lane, and Ying Zhang. 2012. Machine translation with binary feedback: a large-margin approach. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

[Sanchis-Trilles2012] Germán Sanchis-Trilles. 2012. Building task-oriented machine translation systems. In *Ph.D. Thesis, Universitat Politcnica de Valncia*.

[Schwartz2014] Lane Schwartz. 2014. Monolingual post-editing by a domain expert is highly effective for translation triage. In *Proceedings of the The Third Workshop on Post-editing Technology and Practice*, October.

[Simard and Foster2013] Michel Simard and George Foster. 2013. PEPr: Post-edit propagation using phrase-based statistical machine translation. In *Proceedings of the XIV Machine Translation Summit*, pages 191–198,, September.

[Snover et al.2006] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the. Association for Machine Translation of the Americas*, pages 223–231.

[Snover et al.2009] Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268, Athens, Greece, March. Association for Computational Linguistics.

[Soricut et al.2012] Radu Soricut, Nguyen Bach, and Ziyuan Wang. 2012. The sdl language weaver systems in the wmt12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 145–151, Montréal, Canada, June. Association for Computational Linguistics.

[Spearman1904] Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

[Specia and Farzindar2010] Lucia Specia and Atefeh Farzindar. 2010. Estimating machine translation post-editing effort with HTER. In *Proceedings of the AMTA-2010 Workshop Bringing MT to the User: MT Research and the Translation Industry*, pages 33–41.

[Specia and Gimenez2010] Lucia Specia and Jesús Gimenez. 2010. Combining confidence estimation and reference-based metrics for segment level MT evaluation. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

[Specia2011] Lucia Specia. 2011. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th International Conference of the European Association for Machine Translation*.

[Tatsumi et al.2012] Midori Tatsumi, Takako Aikawa, Kentaro Yamamoto, and Hitoshi Isahara. 2012. How Good Is Crowd Post-Editing? Its Potential and Limitations. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 69–77, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).

[Tatsumi2010] Midori Tatsumi. 2010. Post-editing machine translated text in a commercial setting: Observation and statistical analysis. In *Ph.D. thesis, Dublin City University*, October.

[TED Conferences1984] LLC. TED Conferences. 1984. TED: Ideas Worth Spreading. www.ted.com.

[van Rijsbergen1979] C. J. van Rijsbergen, 1979. *Information Retrieval*, chapter 7. Butterworths, London, UK, 2nd edition.

[Veale and Way1997] Tony Veale and Andy Way. 1997. Gaijin: A bootstrapping, templatedriven approach to example-based mt. In *Proceedings of the NeMNLP '97, New Methods in Natural Language Processing*.

[Wang and Manning2012] Mengqiu Wang and Christopher Manning. 2012. Spede: Probabilistic edit distance metrics for mt evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 76–83, Montréal, Canada, June. Association for Computational Linguistics.

[Watanabe et al.2007] Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, June.

[Wikipedia2001] Wikipedia. 2001. Wikipedia: The Free Encyclopedia. http://www.wikipedia.org.

[Xiao2010] Lin Xiao. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *J. Mach. Learn. Res.*, 11:2543–2596, December.

[Yamada and Knight2001] Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, July. Association for Computational Linguistics.

[Zeiler2012] Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.

[Zhechev2012] Ventsislav Zhechev. 2012. Machine Translation Infrastructure and Post-editing Performance at Autodesk. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 87–96, San Diego, USA, October. Association for Machine Translation in the Americas (AMTA).