Real Time Adaptive Machine Translation for Post-Editing with cdec and TransCenter

Michael Denkowski Alon Lavie Isabel Lacruz* Chris Dyer
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA
*Institute for Applied Linguistics, Kent State University, Kent, OH 44242 USA
{mdenkows,alavie,cdyer}@cs.cmu.edu ilacruz@kent.edu

Abstract

Using machine translation output as a starting point for human translation has recently gained traction in the translation community. This paper describes cdec Realtime, a framework for building adaptive MT systems that learn from post-editor feedback, and TransCenter, a web-based translation interface that connects users to Realtime systems and logs post-editing activity. This combination allows the straightforward deployment of MT systems specifically for post-editing and analysis of human translator productivity when working with these systems. All tools, as well as actual post-editing data collected as part of a validation experiment, are freely available under an open source license.

1 Introduction

This paper describes the end-to-end machine translation post-editing setup provided by cdec Realtime and TransCenter. As the quality of MT systems continues to improve, the idea of using automatic translation as a primary technology in assisting human translators has become increasingly attractive. Recent work has explored the possibilities of integrating MT into human translation workflows by providing MT-generated translations as a starting point for translators to correct, as opposed to translating source sentences from scratch. The motivation for this process is to dramatically reduce human translation effort while improving translator productivity and consistency. This computer-aided approach is directly applicable to the wealth of scenarios that still require precise human-quality translation that MT is currently unable to deliver, including an ever-increasing number of government, commercial, and community-driven projects.

The software described in the following sections enables users to translate documents with the assistance of an adaptive MT system using a web-based interface. The system learns from user feedback, improving translation quality as users work. All user interaction is logged, allowing post-editing sessions to be replayed and analyzed. All software is freely available under an open source license, allowing anyone to easily build, deploy, and evaluate MT systems specifically for post-editing. We first describe the underlying adaptive MT paradigm (§2) and the Realtime implementation (§3). We then describe TransCenter (§4) and the results of an end-to-end post-editing experiment with human translators (§5). All data collected as part of this validation experiment is also publicly available.

2 Adaptive Machine Translation

Traditional machine translation systems operate in batch mode: statistical translation models are estimated from large volumes of sentence-parallel bilingual text and then used to translate new text. Incorporating new data requires a full system rebuild, an expensive operation taking up to days of time. As such, MT systems in production scenarios typically remain static for large periods of time (months or even indefinitely). Recently, an *adaptive* MT paradigm has been introduced specifically for post-editing (Denkowski et al., 2014). Three major MT system components are extended to support online updates, allowing human post-editor feedback to be immediately incorporated:

- An online translation model is updated to include new translations extracted from post-editing data.
- A dynamic language model is updated to include post-edited target language text.
- An online update is made to the system's feature weights after each sentence is post-edited.

These extensions allow the MT system to generate improved translations that require significantly less effort to correct for later sentences in the document. This paradigm is now implemented in the freely available `cdec` (Dyer et al., 2010) machine translation toolkit as Realtime, part of the `pycdec` (Chahuneau et al., 2012) Python API.

Standard MT systems use aggregate statistics from all training text to learn a single large translation grammar (in the case of `cdec`'s hierarchical phrase-based model (Chiang, 2007), a synchronous context-free grammar) consisting of rules annotated with feature scores. As an alternative, the bitext can be *indexed* using a suffix array (Lopez, 2008), a data structure allowing fast source-side lookups. When a new sentence is to be translated, training sentences that share spans of text with the input sentence are *sampled* from the suffix array. Statistics from the sample are used to learn a small, sentence-specific grammar on-the-fly. The adaptive paradigm extends this approach to support online updates by also indexing the new bilingual sentences generated as a post-editor works. When a new sentence is translated, matching sentences are sampled from the post-editing data as well as the suffix array. All feature scores that can be computed on a suffix array sample can be identically computed on the combined sample, allowing uniform handling of all data. An additional "post-edit support" feature is included that indicates whether a grammar rule was extracted from the post-editing data. This allows an optimizer to learn to prefer translations that originate from human feedback. This adaptation approach also serves as a platform for exploring expanded post-editing-aware feature sets; any feature that can be computed from standard text can be added to the model and will automatically include post-editing data. Implementationally, feature scoring is broken out into a single Python source file containing a single function for each feature score. New feature functions can be added easily.

The adaptive paradigm uses two language models. A standard (static) n-gram language model estimated on large monolingual text allows the system to prefer translations more similar to human-generated text in the target language. A (dynamic) Bayesian n-gram language model (Teh, 2006) can be updated with observations of the post-edited output in a straightforward way. This smaller model exactly covers the training bitext and all post-editing data, letting the system upweight translations with newly learned vocabulary and phrasing absent in the large monolingual text. Finally, the margin-infused relaxed algorithm (MIRA) (Crammer et al., 2006; Eidelman, 2012) is used to make an online parameter update after each sentence is post-edited, minimizing model error. This allows the system to continuously rescale weights for translation and language model features that adapt over time.

Since 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, a simulated post-editing paradigm (Hardt and Elming, 2010) can be used, wherein pre-generated reference translations act as a stand-in for actual post-editing. This approximation is effective for tuning and internal evaluation when real post-editing data is unavailable. In 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 text. After each sentence is translated, the following 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 word-alignment models learned from the initial data and used to update the translation grammar. Third, the reference is added to the Bayesian language model. As sentences are translated, the models gain valuable context information, allowing them to adapt to the specific target document and translator. Context is reset at the start of each development or test corpus. Systems optimized with simulated post-editing can then be deployed to serve real human translators without further modification.

3 `cdec` Realtime

Now included as part of the free, open source `cdec` machine translation toolkit (Dyer et al., 2010), Realtime[1] provides an efficient implementation of the adaptive MT paradigm that can serve an arbitrary number of unique post-editors concurrently. A full Realtime tutorial, including step-by-step instructions for installing required software and building full adaptive systems, is avail-

[1] `https://github.com/redpony/cdec/tree/master/realtime`

```
import rt

# Start new Realtime translator using a Spanish--English
# system and automatic, language-independent text normalization
# (pre-tokenization and post-detokenization)
translator = rt.RealtimeTranslator('es-en.d', tmpdir='/tmp', cache_size=5,
    norm=True)

# Translate a sentence for user1
translation = translator.translate('Muchas gracias Chris.', ctx_name='user1')

# Learn from user1's post-edited translation
translator.learn('Muchas gracias Chris.', 'Thank you so much, Chris.',
    ctx_name='user1')

# Save, free, and reload state for user1
translator.save_state(file_or_stringio='user1.state', ctx_name='user1')
translator.drop_ctx(ctx_name='user1')
translator.load_state(file_or_stringio='user1.state', ctx_name='user1')
```

Figure 1: Sample code using the Realtime Python API to translate and learn from post-editing.

able online.[2] Building an adaptive system begins with the usual MT pipeline steps: word alignment, bitext indexing (for suffix array grammar extraction), and standard n-gram language model estimation. Additionally, the `cpyp`[3] package, also freely available, is used to estimate a Bayesian n-gram language model on the target side of the bitext. The `cdec` grammar extractor and dynamic language model implementations both include support for efficient inclusion of incremental data, allowing optimization with simulated post-editing to be parallelized. The resulting system, optimized for post-editing, is then ready for deployment with Realtime.

At runtime, a Realtime system operates as follows. A single instance of the indexed bitext is loaded into memory for grammar extraction. Single instances of the directional word alignment models are loaded into memory for force-aligning post-edited data. When a new user requests a translation, a new *context* is started. The following are loaded into memory: a table of all post-edited data from the user, a user-specific dynamic language model, and a user-specific decoder (in this case an instance of MIRA that has a user-specific decoder and set of weights). Each user also requires an instance of the large static language model, though all users effectively share a single instance through the memory mapped implementation of KenLM (Heafield, 2011). When a new sentence is to be translated, the grammar extractor samples from the shared background data plus the user-specific post-editing data to generate a sentence-specific grammar incorporating data from all prior sentences translated by the same user. The sentence is then decoded using the user and time-specific grammar, current weights, and current dynamic language model. When a post-edited sentence is available as feedback, the following happen in order: (1) the source-reference pair is used to update feature weights with MIRA, (2) the source-reference pair is force-aligned and added to the indexed post-editing data, and (3) the dynamic language model is updated with the reference. User state (current weights and indexed post-edited data for grammars and the language model) can be saved and loaded, allowing models to be loaded and freed from memory as translators start and stop their work. Figure 1 shows a minimal example of the above using the Realtime package. While this paper describes integration with TransCenter, a tool primarily targeting data collection and analysis, the Realtime Python API allows straightforward integration with other computer-assisted translation tools such as full-featured translation workbench environments.

4 TransCenter: Web-Based Translation Research Suite

The TransCenter software (Denkowski and Lavie, 2012) dramatically lowers barriers in post-editing data collection and increases the accuracy and descriptiveness of the collected data. TransCenter

[2] `http://www.cs.cmu.edu/~mdenkows/cdec-realtime.html`

[3] `https://github.com/redpony/cpyp`

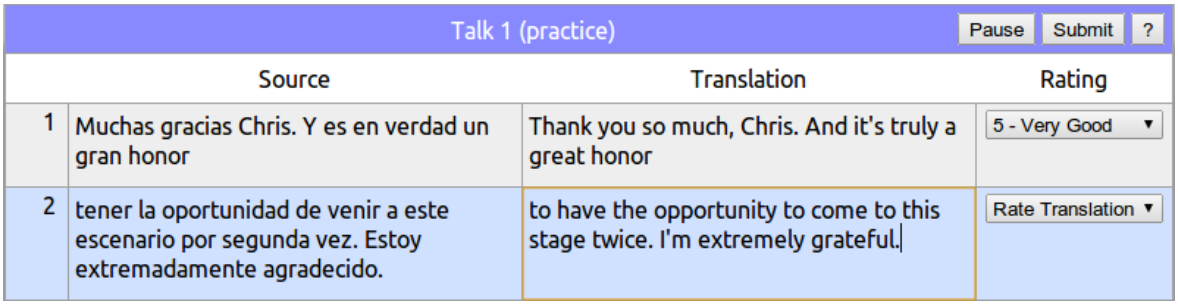

Figure 2: Example of editing and rating machine translations with the TransCenter web interface.

ID	MT	Post-Edited	Rating	Keypress	Mouseclick	Edits	Time
4	because car designers tend to be a little low on the totem,	because car designers tend to be a little low on the totem pole,	4	5	1	5	16677
5	we do not table with only one bottle inside, books	we don't make coffee table books with a single lamp inside,	1	107	1	107	44017
6	and it is thought both cars and product	and cars are thought of so much as a product	2	77	1	77	28907

Figure 3: Example TransCenter summary report for a single user on a document.

provides a web-based translation editing interface that remotely monitors and records user activity. The "live" version[4] now uses `cdec` Realtime to provide on-demand MT that automatically learns from post-editor feedback. Translators use a web browser to access a familiar two-column editing environment (shown in Figure 2) from any computer with an Internet connection. The left column displays the source sentences, while the right column, initially empty, is incrementally populated with translations from the Realtime system 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. During editing, all user actions (key presses and mouse clicks) are logged so that the full editing process can be replayed and analyzed. After editing, the final translation is reported to the Realtime system for learning and the next translation is generated. The user is additionally asked to rate the amount of work required to post-edit each sentence immediately after completing it, yielding maximally accurate feedback. The rating scale ranges from 5 (no post-editing required) to 1 (requires total re-translation). TransCenter also records the number of seconds each sentence is focused, allowing for exact timing measurements. A pause button is available if the translator needs to take breaks. TransCenter can generate reports of translator effort as measured by (1) keystroke, (2) exact timing, and (3) actual translator post-assessment. Final translations are also available for calculating edit distance. Millisecond-level timing of all user actions further facilitates time sequence analysis of user actions and pauses. Figure 3 shows an example summary report generated by TransCenter showing a user's activity on each sentence in a document. This information is also output in a simple comma-separated value format for maximum interoperability with other standards-compliant tools.

TransCenter automatically handles resource management with Realtime. When a TransCenter server is started, it loads a Realtime system with zero contexts into memory. As users log in to work on documents, new contexts are created to deliver on-demand translations. As users finish working or take extended breaks, contexts automatically time out and resources are freed. Translator and document-specific state is automatically saved when contexts time out and reloaded when translators resume work with built-in safeguards against missing or duplicating any post-editing data due to timeouts or Internet connectivity issues. This allows any number of translators to work on translation tasks at their convenience.

5 Experiments

In a preliminary experiment to evaluate the impact of adaptive MT in real-world post-editing scenar-

[4] `https://github.com/mjdenkowski/transcenter-live`

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

Table 1: Aggregate HTER scores and average translator self-ratings (5 point scale) of post-editing effort for translations of TED talks from Spanish into English.

ios, we compare a static Spanish–English MT system to a comparable adaptive system on a blind out-of-domain test. Competitive with the current state-of-the-art, both systems are trained on the 2012 NAACL WMT (Callison-Burch et al., 2012) constrained resources (2 million bilingual sentences) using the cdec toolkit (Dyer et al., 2010). Blind post-editing evaluation sets are drawn from the Web Inventory of Transcribed and Translated Talks (WIT[3]) corpus (Cettolo et al., 2012) that makes transcriptions of TED talks[5] available in several languages, including English and Spanish. We select 4 excerpts from Spanish talk transcripts (totaling 100 sentences) to be translated into English. Five students training to be professional translators post-edit machine translations of these excerpts using TransCenter. Translations are provided by either the static or fully adaptive system. Tasks are divided such that each user translates 2 excerpts with the static system and 2 with the adaptive system and each excerpt is post-edited either 2 or 3 times with each system. Users do not know which system is providing the translations.

Using the data collected by TransCenter, we evaluate post-editing effort with the established human-targeted translation edit rate (HTER) metric (Snover et al., 2006). HTER computes an edit distance score between initial MT outputs and the "targeted" references created by human post-editing, with lower scores being better. Results for the two systems are aggregated over all users and documents. Shown in Table 1, introducing an adaptive MT system results in a significant reduction in editing effort. We additionally average the user post-ratings for each translation by system to evaluate user perception of the adaptive system compared to the static baseline. Also shown in Table 1, we see a slight preference for the adaptive system. This data, as well as precise keystroke, mouse click, and timing information is made freely available for further analysis.[6] TransCenter records all data necessary for more sophisticated editing time analysis (Koehn, 2012) as well as analysis of translator behavior, including pauses (used as an indicator of cognitive effort) (Lacruz et al., 2012).

6 Related Work

There has been a recent push for new computer-aided translation (CAT) tools that leverage adaptive machine translation. The CASMACAT[7] project (Alabau et al., 2013) focuses on building state-of-the-art tools for computer-aided translation. This includes translation predictions backed by machine translation systems that incrementally update model parameters as users edit translations (Martínez-Gómez et al., 2012; López-Salcedo et al., 2012). The MateCat[8] project (Cattelan, 2013) specifically aims to integrate machine translation (including online model adaptation and translation quality estimation) into a web-based CAT tool. Bertoldi et al. (2013) show improvements in translator productivity when using the MateCat tool with an adaptive MT system that uses cache-based translation and language models.

7 Conclusion

This paper describes the free, open source MT post-editing setup provided by `cdec` Realtime and TransCenter. All software and the data collected for a preliminary post-editing experiment are all freely available online. A live demonstration of adaptive MT post-editing powered by Realtime and TransCenter is scheduled for the 2014 EACL Workshop on Humans and Computer-assisted Translation (HaCaT 2014).

Acknowledgements

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 XSEDE program under grant TG-CCR110017.

[5] `http://www.ted.com/talks`

[6] `www.cs.cmu.edu/~mdenkows/transcenter-round1.tar.gz`

[7] `http://casmacat.eu/`

[8] `http://www.matecat.com/`

References

Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González-Rubio, Philipp Koehn, Luis A. Leiva, Bartolomé Mesa-Lao, Daniel Ortiz-Martínez, Hervé Saint-Amand, Germán Sanchis-Trilles, and Chara Tsoukala. 2013. Casmacat: An open source workbench for advanced computer aided translation. In *The Prague Bulletin of Mathematical Linguistics*, pages 101–112.

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.

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.

Alessandro Cattelan. 2013. Second version of MateCat tool. Deliverable 4.2.

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*.

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

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

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

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

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. 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*.

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.

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.

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*.

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, United Kingdom, July.

Philipp Koehn. 2012. Computer-aided translation. Machine Translation Marathon.

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).

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

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.

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.

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.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of ACL*.