Improved Domain Adaptation for Statistical Machine Translation

Wei Wang and **Klaus Macherey** and **Wolfgang Macherey** and **Franz Och** and **Peng Xu**
Google Inc. 1600 Amphitheatre Pkwy.
Mountain View, CA 94043, USA
{wangwe,kmach,wmach,och,xp}@google.com

Abstract

We present a simple and effective infrastructure for domain adaptation for statistical machine translation (MT). To build MT systems for different domains, it trains, tunes and deploys a single translation system that is capable of producing adapted domain translations and preserving the original generic accuracy at the same time. The approach unifies automatic domain detection and domain model parameterization into one system. Experiment results on 20 language pairs demonstrate its viability.

1 Introduction

Research in domain adaption for machine translation (MT) has been mostly focusing on one domain. Various methods have been proposed to make a system work best on a resource-scarce domain when most of the training data is from another open, resource-rich domain, e.g., (Foster et al., 2010; Foster and Kuhn, 2007; Koehn and Schroeder, 2007). Decent improvements have been made on domain translation accuracy, but often, accuracy improvements for one domain are obtained at the expense of accuracy losses in another (e.g., the background) domain.

With these methods, it remains unaddressed how they can be generalized to work equally well with more than one domains at the same time. One could trivially build one system/model per domain, but that does not scale and will require manual domain detection if the incoming texts belong to heterogeneous domains. So far, there has been little work on better infrastructure for building and deploying large-scale multi-domain MT systems.

In this paper, we present a simple and effective domain adaptation infrastructure that makes a single MT system that has a single translation model capable of providing adapted, close-to-upper-bound domain translation accuracy and preserves the generic translation accuracy at the same time. This is fulfilled by introducing domain awareness into the traditional single-domain decoding and tuning components with the help of an automatic domain detector, to generalize an MT system to handle different domains. We study this approach with two domains (generic and patent), carry out large-scale experiments for 20 language pairs, demonstrating the viability of our approach.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 presents our domain adaptation approach. Section 4 explains our method to classify an input sentence into its domain. Sections 5 and 6 talk about genre-aware decoding and tuning. We present experiment results in Section 7 and conclude in Section 8.

2 Related Work

The work of Xu et al. (2007) and Banerjee et al. (2010) are perhaps the most relevant to our work. Xu et al. (2007) adapt a shared, generic translation model for better web or broadcast conversation translations and use a source-document classifier to classify an input document into a domain. This work makes the translation model shared across different domains, but domain-specific training data is not used and thus its impact is not studied. Neither did

they report the generic MT accuracy.

Banerjee et al. (2010) use source-sentence classification to combine two separate domain models, each trained from small amounts of domain-specific data obtained from a single corporate website. The work does not study nor report the impact of generic data on domain translation accuracy.

Both works realize the automatic domain detection with an experimental setup where the classifier works as a "switch" between two independent MT decoding runs, lacking a deeper integration, also making it difficult to introduce genre awareness into other components (e.g., tuning) other than decoding.

Current research along this line does not seem to provide enough evidence that the automatic domain classification idea is practical yet for building large-scale MT systems that are optimized for different domains. Neither has it proposed a practical, unified infrastructure. In particular, this idea remains unexamined under the following conditions:

- A wide range of classification error rates.
- Unbalanced classification error rates across different domains.
- Vast amounts of generic data and varying amounts of domain data.
- Different languages.

3 Our Approach

Our approach intends to generalize the standard single-domain MT infrastructure to more than one domains, by making the necessary infrastructure components aware of domains (or genres). The approach is simple but turns out to be effective.

We use a single translation-model parameterization to serve different domains, rather than introducing multiple models, each for one domain. In the model, there are no domain-dependent phrase features at all and thus any phrase and its features can be used in the decoding of sentences of any domain. But during phrase extraction, we do record all the domains where a phrase comes from. We train the translation model on the merged generic and domain-specific bilingual data. Therefore, our translation model has no major difference from the standard, widely-used single-domain translation model, except for the phrase provenance information we keep for each phrase. Using a single model makes the system scale more easily to many domains: We just need to maintain one model and any future modeling improvement can be immediately available to different domains.

The domain awareness is introduced in decoding and tuning. In decoding, a genre classifier is used to generalize the conventional single-domain decoding to distinguish genres: The genre classifier classifies an input sentence f to a domain $d(f)$, using knowledge (e.g., the phrase provenance information we keep for each phrase) solely from the translation model. Contrast to the single-domain decoding that finds the best translation $\hat{e}$ for f with formula

$$\hat{e} = \arg\max_{e} \left(\sum_{i=1}^{I} \lambda_i \cdot h_i\left(f, e\right) \right) \qquad (1)$$

where h_i's are features and λ_i's are feature weights, in our generalized decoding, once the decoder retrieves from the translation model a phrase that matches f, we re-label its features, say h_i, into $h_i^{d(f)}$ with the classified domain label $d(f)$. Re-labeling makes feature $h_i^{d(f)}$ a different feature from h_i^c used by a different domain c, even though both re-labeled features are originally the same feature in the translation model. A re-labeled feature now can have its own (domain-specific) weight. In another word, the runtime feature re-labeling makes a feature that is shared, domain-independent in the translation model become decoupled, domain-dependent at decoding runtime and consequently (and importantly) in the n-best lists for tuning. The generalized decoding finds the best translation with the following formula:

$$\hat{e} = \arg\max_{e} \left(\sum_{i=1}^{I} \lambda_i^{d(f)} \cdot h_i^{d(f)}\left(f, e\right) \right) \qquad (2)$$

Generalizing decoding for genre awareness in turn makes tuning genre-aware. Our tuning development set consists of sentences from D domains. And we do not need to know the domain for each sentence beforehand. The genre-aware decoding on the entire set automatically classifies it into D partitions. The runtime feature re-labeling makes each partition, S_d, $d = 0, \ldots, D-1$, have its own set of features that are decoupled from other domains.

Therefore the system has D sets of features in total. We then can use Minimum Error Rate Training (MERT) (Och, 2003) out of the box to learn the weights for all these features in a single MERT run which maximizes the overall BLEU of the entire genre-mixed development set:

$$\max_{\{\lambda_i^d, d=1...D, i=1...I_d\}} \text{BLEU}(\cup_{d=0}^{D-1} S_d) \qquad (3)$$

In this formula, there is no explicit treatment on genre, but there is a subtlety: Features are decoupled in n-best lists across domains and this gives MERT the freedom to adjust weights for one domain without much constraints from any other domain. In other words, MERT is made implicitly genre-aware.

Actually, we can further tailor the MERT tuning objectives to be explicitly aware of genre. We explain alternative genre-aware tuning objectives that the classical MERT can be altered to adopt in a later section and compare their effects experimentally.

Daume III (2007) does domain adaptation by augumenting features. Chiang et al. (2011) improve lexical smoothing also by augmenting features refined by genres. In our approach, the n-best lists for tuning can be viewed as containing augmented features as well. But we augment features in order to decouple them across domains, rather than providing refined domain/genre bias in the model.

The language model feature deserves further explanation. Even though we do not have domain specific features in the translation model, we have domain specific language models. In our approach, the generic language model is used by different domains, but a domain language model is turned on only if the input sentence is classified as the corresponding domain.

4 Genre Classifier

A genre classifier detects domains by classifying a source sentence into its genre (or domain), so that the MT system can be configured to use the proper domain feature weights and turn on the appropriate domain language model.

Text classification is a well-studied field, largely in the purview of Information Retrieval but with lots of crossovers to Natural Language Processing as well. The previous work that is directly related to us is (Xu et al., 2007; Banerjee et al., 2010). Their work uses either the source language model approach or the information retrieval approach to implement the genre classifier. The former requires an additional source language model and the latter needs source lexical statistics. Both approaches incur additional RAM and efficiency costs.

Our classifier is implemented by "re-using" the phrase table. The features in the classifier are designed to indicate the domain provenance of the source sentence by considering the relative portion of domain phrases with respect to all the phrases retrieved for the source sentence. We use the Perceptron algorithm to implement the classifier.

4.1 Features

Generally, if most source sentence words originate from a domain, it is likely that the sentence belongs to that domain. Or, in the set of phrases we retrieve for the source sentence, if a certain number of phrases are from the domain training data, the source sentence may come from that domain. The classifier features are defined in these two perspectives:

Domain word coverage The ratio between the number of words that are covered by any domain phrase and the source sentence length. The more words that are covered by phrases from a domain, the more likely the sentence belongs to that domain.

Domain word coverage by phrase length The ratio between the number of words that are covered by a max-length-l domain phrase and the source sentence length. The insight of having these finer-grained features in addition to the above coarse version is that a word being covered by a longer domain phrase could be a stronger indication of domain provenance than being covered by a shorter one.

Average phrase length The average max-domain-phrase-length (per word). It is computed by summing up the length of the longest-covering domain phrase for each word and then divide it by the sentence length. If a sentence tends to be covered by longer domain phrases, rather than short (and frequent) ones, the sentence is more likely to belong to that domain.

Domain phrase ratio The ratio between domain phrases to the total number of phrases retrieved for

the source sentence. The bigger the ratio, the more likely the sentence is from that domain.

Domain phrase ratio by phrase length The ratio between the domain phrases of length l to the total number of phrases of length l retrieved for the source sentence. The insight is that domain phrase ratio at a longer phrase length could be more discriminative than at a shorter length. Making the domain phrase ratio features finer-grained at phrase length enables the training to treat them differently.

Except for the average phrase length feature, we convert the above ratios to the logarithm space before using a learning algorithm to learn their weights. In experiments, we observed classification accuracy improvement with this conversion. When there are multiple domains, the above features are computed for each respective domain, and then a multi-class classifier is used to make the final decision.

In principle, we could also use language model based features that capture n-gram coverage. But in practice, we use only phrase-based features as described above. This is an efficiency optimization. We do not want to query the domain language models for generic sentences, but we need to query the phrase table regardless. In terms of accuracy, as we will show later, we can achieve a decent classification accuracy by using only phrase-based features.

4.2 Classification algorithm

The classifier we use is an averaged perceptron, which has a weight w_i for each feature f_i, $i = 1 \ldots I$, and uses a linear combination of the features. If we have only two domains, the classification decision y is made by:

$$y(f_{i=1\ldots I}) = \begin{cases} \text{domain } A & \text{if } \sum_{i=1}^{I} f_i * w_i < T \\ \text{domain } B & \text{otherwise} \end{cases}$$

T is a threshold whose value we can adjust to achieve a desired tradeoff between the classification accuracies of the two domains. Raising the accuracy for a domain can decrease that of another. When high classification accuracy is not possible for both domains, finding a proper tradeoff could be important for meeting user's MT accuracy requirements – if we desire a lossless generic BLEU, we need to change T to raise the classification accuracy on generic texts, usually by sacrificing some domain classification accuracy.

Multi-class classification can be realized by using multiple binary classifiers, each classifying between domains c and d. The class that has the majority votes wins.

5 Genre-Aware Decoding

The genre classifier is integrated into the decoder to detect the domain of an input sentence. Then the decoder chooses the proper decoding configuration to decode the sentence. A decoding configuration includes the feature weights for the detected domain and the additional domain specific language model feature. For a domain c decoding, the decoder turns on the domain c language model and turns off that of domain d. The generic language mode is used by all domains.

The genre-aware decoder records the genre information in the feature names when it emits n-best lists. As a result, once the decoding on the entire development set is finished, the set is partitioned into different portions, each corresponding to a domain and each having its own feature set. Features are not shared across domains in n-best lists.

6 Genre-Aware Tuning Objectives

The conventional single-domain tuning induces the optimal feature weights $\{\lambda_i, i = 1 \ldots I\}$ that maximizes the BLEU of the entire development set S:

$$\max_{\{\lambda_i, i=1\ldots I\}} \text{BLEU}(S) \tag{4}$$

In our genre-aware tuning, the tuning development set consists of sentences from D domains. A genre-aware decoding classifies S into D partitions, each partition S_d having its own set of features $\{\lambda_i^d, i = 1 \ldots I_d\}$. So the system has D sets of features in total. Then MERT induces the weights for all these features by maximizing an objective. There are more than one ways to reflect genre in the tuning objectives.

As we explained in Section 3, we can use the optimization objective that maximizes BLEU at the entire genre-mixed development corpus level. We call it *max joint BLEU* as it learns all the feature weights

jointly:

$$\max_{\{\lambda_i^d, d=1...D, i=1...I_d\}} \text{BLEU}(S = \cup_{d=1}^{D} S_d) \quad (5)$$

Similar to the above single-domain optimization problem, this objective does not explicitly take genre into account. However, the weights of different domains, $\{\lambda_i^d, d = 1 \ldots D, i = 1 \ldots I_d\}$, are decoupled (by the runtime feature name re-labeling (Section 3)) in the n-best lists, rather than being shared, thus the weight optimization for one domain can concentrate on the domain itself, without being constrained too much by any other domain. Since BLEU is not decomposable at the sentence level, this objective generally can not guarantee maximized BLEUs on respective domain partitions, but rather an optimal overall BLEU (Chiang et al., 2008).

Another optimization objective is the *max BLEU sum* that maximizes the sum of BLEUs of the individual genre partitions:

$$\max_{\{\lambda_i^d, d=1...D, i=1...I_d\}} \left(\sum_{d=1}^{D} \text{BLEU}(S_d) \right) \quad (6)$$

When none of the features is shared in the n-best lists across different domains (which is our case), this objective is equivalent to the summation of the individually maximized BLEUs:

$$\sum_{d=1}^{D} \left(\max_{\{\lambda_i^d, i=1...I_d\}} \text{BLEU}(S_d) \right) \quad (7)$$

There could be other genre-aware tuning objectives that mix (or "nest") the above two.[1] But in our paper, we are mainly interested in the experimental comparison between the max joint BLEU and the max BLEU sum objectives, due to concerns about the potential length-penalty effects on genre-aware tuning/decoding (motivated by Table 3 of (Chiang et al., 2008)).

Due to classification errors, the classified development set or the classified test set of a domain can be different from the true set. This affects the translation quality in two ways. In tuning, the feature weights of domain d are actually tuned on a polluted domain d development set by false positives. A high false positive rate will make the feature weights of domain d deviate from the optimal feature weights, resulting in less accurate translations. In decoding, the false positives that belong to other domains are decoded using the domain d weights, leading to less accurate translations as well. We study the impact of classification error rates on the domain translation accuracy in Section 7.

[1] For example, for some domains, we can separately optimize their weights, respectively; for some other domains, we can jointly optimize the BLEU of their merged development set. These two types of optimization can then be combined into one single objective.

7 Experiments

7.1 Setup

Our experiments are carried out for 20 language pairs, in both directions, between English and 10 European languages: Italian, Spanish, French, Portuguese, German, Swedish, Finnish, Turkish, Danish, and Dutch. We have two domains: generic and patent. The generic parallel data size is around 250 million words for each language pair. The patent parallel data is from the European Patent Office (`www.epo.org`). The parallel data we have for each language pair ranges from 0.8 million words to 10 million words.

The MERT tuning set constitutes a generic development subset (3400 sentences) and a patent domain development set (2000 sentences). For testing, each generic test set contains 5000 sentences and each patent domain test set contains 2000 sentences. Sentence overlapping between training, development and test sets are removed from the training data.

For each language pair, we train a 4-gram generic target LM. We also train a 4-gram patent LM from the target side of the patent parallel data. LM data overlap with the development set and the test set is removed.

We use a phrase-based system (Koehn et al.,), which has a source tree pre-ordering module (Xu et al., 2009). MERT optimizes BLEU on lattices (Macherey et al., 2008).

7.2 Genre classification accuracy

We use the MT development sets as the training data of the genre classifier and the MT test sets as the

Classifier	Generic (%)		Patent (%)	
Features	dev	test	dev	test
word coverage (5)	97.2	97.0	53.1	52.9
phrase ratio (5)	96.8	97.1	43.3	40.3
avg. phrase length (1)	97.3	97.0	24.2	23.7
all (11)	97.0	97.2	87.6	85.5

all features, per language pair:

language pair	generic (%)		patent (%)	
English/Danish	97.1	96.1	87.8	85.7
English/Dutch	97.8	97.9	91.1	92.6
English/Finnish	96.3	95.8	85.1	82.4
English/French	99.8	99.6	98.2	96.1
English/German	99.5	99.2	95.6	96.5
English/Italian	97.5	96.9	91.3	89.2
English/Portuguese	95.5	95.5	85.4	86.5
English/Spanish	96.8	96.6	90.4	90.6
English/Swedish	96.2	98.9	73.4	67.2
English/Turkish	96.3	96.6	85.3	80.1
Danish/English	97.3	96.0	86.1	82.3
Dutch/English	97.3	97.4	88.2	88.3
Finnish/English	92.8	93.8	77.4	76.7
French/English	99.7	99.6	97.8	94.6
German/English	99.3	99.0	96.1	95.9
Italian/English	96.3	96.0	88.1	83.8
Portuguese/English	96.0	96.9	86.3	86.2
Spanish/English	97.3	96.7	88.4	86.6
Swedish/English	95.3	97.7	78.0	70.1
Turkish/English	95.3	97.0	83.1	78.3

Table 1: Genre classification precisions measured on a generic test set and a patent test set. In the first table, each accuracy is an average over 20 language pairs; and numbers in round brackets are the number of features used.

classifier test sets. The gold-standard label for each source sentence is automatically known based on its file origin. Since the MT dev and test sets lack short sentences, for each language pair, we collect the 100 most frequent words from its generic development set as additional generic training data for the genre classifier of that language pair. This encourages the classifier to learn to classify short sentences to the generic domain.

Even though the classifier takes a monolingual (source) sentence as input, it needs to know the language pairs of the MT system as well, because the classifier features are computed using the translation phrases in the phrase table. We train the averaged perceptron for 200 iterations.

Table 1 shows the dev and test precisions of classifying between the two domains. In our case, we are targeting a lossless generic translation accuracy, so we adjust the classification threshold (on our dev set) to obtain a high generic classification precision. On an average over the 20 language pairs, the genre classifier classifies generic sentences at a 97.2% (test) precision and patent sentences at a 85.5% (test) precision. For individual language pairs, the patent classification precision is as low as 67.2% (English/Swedish test), or as high as 96.5% (English/German test). The dev and test precisions seem to be very consistent.

Table 1 also shows the performance of different types of classifier features. Just using one type does not suffice and the combination of all brings us the best patent classification accuracy.

7.3 BLEU

We carry out a series of experiments to examine how patent resources like parallel data, dev set and language model are helpful and to examine how effective our presented domain adaptation approach is.

The experiments are described in Table 2, where the difference of experiments lies in the parallel training data, dev data, language model used and if an experiment uses the genre classifier. To examine how the patent domain bilingual data is helpful, we build systems from the following data and tune them only on the generic development sets. Here T stands for training data.

- T1: generic bilingual data.
- T2: generic + patent training data.
- T3: just patent training data.

To examine the effect of different choices of dev sets on translation accuracy, we tune systems on (where D stands for dev set):

- D4: T2 + combined dev set
- D5: T2 + just patent dev set

To examine the effect of using domain language models, we run (where L stands for language model):

- L6: D4 + patent LM
- L7: D5 + patent LM

Experiment	Train Data	Dev Data	LM	Classifier		Generic BLEU		Patent BLEU	
				tuning	testing	dev	test	dev	test
T1	G	G	G	none	none	30.41	30.14	32.66	33.21
T2	G+P	G	G	none	none	30.43	30.17	35.27	35.56
T3	P	G	G	none	none	13.25	13.04	33.76	33.88
D4	G+P	G+P	G	none	none	29.63	29.58	37.40	37.18
D5	G+P	P	G	none	none	28.52	28.71	37.79	37.19
L6	G+P	G+P	G+P	none	none	28.79	28.75	39.40	38.95
L7	G+P	P	G+P	none	none	21.63	21.71	41.20	40.17
C8.1	G+P	G+P	G+P	yes	yes	30.17	29.98	40.77	39.82
Oracle	G+P	G+P	G+P	oracle	oracle	30.42	30.16	41.20	40.17
tune-Oracle	G+P	G+P	G+P	oracle	yes	30.25	30.00	40.72	39.78

Table 2: Generic BLEUs and patent domain BLEUs. G=generic, P=patent, yes=a real genre classifier, oracle=a perfect/cheating genre classifier. Significant tests were performed between C8.1 (our approach) and T1 (baseline) for the dev and test sets of all 20 language pairs, respectively: At $p < 0.005$, none of the system significantly differs in generic BLEU and all patent-BLEU improvements are significant. In column one, T/D/L/C indicates what is being compared. I.e., T means training data, D means dev data, L means language model and C means classifier.

Please note that, in L6 and L7, the patent language model is used by both the generic decoding and the patent decoding, just as the generic language model. L6 and L7 (as well as T1-D5) do not use the genre classifier but just treat the dev and test as belonging to the same genre. The purpose of T1-L7 is to show the effects of different patent resources (of 20 language pairs) without resorting to genre classification. In-domain parallel data and in-domain language model have been previously shown helpful for domain adaptation, for example, by Koehn and Schroeder (2007).

To verify that our presented approach really produces optimized translation accuracy for different domains, we run experiments C8.1 (where C stands for classifier) that uses the genre classifier in both tuning and decoding, and that uses the max BLEU sum tuning objective in Eq. (7):

- C8.1: L6 + classifier + max BLEU sum

In C8.1, the patent language model is used only when the genre classifier classifies an input sentence as patent, but the generic language model is used by both domains.

We also run an oracle experiment (Oracle in Table 2) to establish the upper bounds of both domains for C8.1. This system uses a perfect genre classifier that has a 100% classification precision. It is re-tuned, and the perfect classifier is used in both tuning and testing.[2] We'll explain another oracle experiment, tune-Oracle, in a latter section.

Experiment	Generic TER		Patent TER	
	dev	test	dev	test
T1	49.05	48.75	50.71	49.35
C8.1	48.93	48.81	44.65	45.16
Oracle	49.05	48.75	44.35	44.84

Table 3: Generic TERs and patent domain TERs for T1 (baseline), C8.1 (our approach) and Oracle.

The BLEU results are shown in Table 2. Each number is an average over 20 language pairs. We also compute the TER scores (in Table 3) for T1 (baseline), C8.1 (our approach) and the Oracle, respectively. The TER scores confirm the BLEU gains achieved by our approach.

T1 vs. T2 shows that simply merging the patent bilingual data into the vast amounts of generic data improves the patent test BLEU by 2.35 points without any negative effect on generic BLEU. T2 vs. T3 shows that the use of generic data for training leads to accuracy improvement for patent translation and also preserves the generic translation accuracy.

D4 vs. T2 shows that adding domain sentences in the dev set effectively improves patent test BLEU by another 1.62 points, but drops the generic test BLEU by 0.6 points at the same time. D5 vs. D4 shows that

[2] In principle, the Oracle generic BLEUs are expected to be identical to the generic BLEUs of T2 and the Oracle patent BLEUs are expected to be identical to L7: A system with a 100%-precision classifier, tuned with objective in Eq. (7) behaves like two independent systems: T2 for generic and L7 for patent. We still run the actual Oracle experiment here for clarity purpose.

putting only domain sentences in the dev set hurts generic BLEU even more.

L7 vs. D5 (or L6 vs. D4) shows that a domain LM improves the patent test BLEU by 3 points (or 1.77 points), but drastically lowers down the generic BLEU[3] when the tuning set has a significant portion of the domain-specific sentences and when we do not classify/distinguish genres.

Oracle establishes the generic and patent BLEU upper bounds for C8.1. These two upper bounds are not closely approached at once in any system built by experiments T1-L7. Our approach, however, is able to produce a system (C8.1) whose translation accuracies are very close to the respective BLEU upper bounds: We are 0.25 dev (or 0.18 test) BLEU points away from the generic upper bound, and 0.43 dev (or 0.35 test) BLEU points away from the patent domain upper bound.

Compared to the baseline T1, our approach builds a single system that improves patent translation by 6.6 test BLEU points (or 4.2 test TER points according to Table 3) with a slight BLEU drop in generic translation. We perform significance test between T1 and C8.1 for both dev and test of all 20 language pairs, respectively, using the paired bootstrap resampling (Koehn, 2004): at $p < 0.0005$, none of the generic-BLEU drop is significant, all patent-BLEU improvements are significant.

7.4 Classification error rate vs. BLEU loss

As explained in Section 6, genre classification errors incur BLEU loss compared to its oracle upper bound. We are therefore interested in investigating how BLEU loss varies with respect to the classification error. Moreover, knowing a quantitative correspondence between them would make it possible to predict if the accuracy of a genre classifier under development (e.g., for a new language pair) meets our requirement on BLEU without running the actual decodings.

In our approach, genre classification errors lead to potential BLEU loss in two ways. In decoding, wrongly classified sentences will be decoded using non-perfect decoding feature weights (and wrong domain language model); In tuning, wrongly classified sentences of domain A may "pollute" the dev set (thus the tuning quality) of domain B when these sentences are mis-classified to domain B. We use a real classifier in our approach rather than a perfect one in tuning for the purpose of simplicity, so that decoders in tuning and testing behave the same.

We first examine how the combined errors (of tuning and testing) affect BLEU. An averaged correspondence between BLEU loss and genre classification error rate can be directly computed from Table 1 and Table 2 and is shown in Table 4. BLEU loss here equals the difference between the C8.1 BLEUs and the Oracle BLEUs. In Table 4, each number is an average over 20 language pairs. The 3.0% dev (or 2.8% test) error rate in generic classification leads to 0.25 dev (or 0.18 test) generic BLEU loss, so 1% generic classification error incurs a loss of (less than) 0.1 generic BLEU points. In comparison, the 12.4% dev (or 14.5% test) error rate in patent classification leads to 0.43 dev (or 0.35 test) patent BLEU loss, so 1% patent classification error results in a loss of (less than) 0.04 patent BLEU points. In other words, the patent BLEU loss is less sensitive to genre classification errors than the generic BLEU loss. This unbalanced behavior proves the importance of using generic data for domain adaptation – The vast amounts of generic training data can ensure a decent fallback translation accuracy for wrongly classified patent texts, but not the other way round.

A per-language-pair correspondence between patent BLEU loss and patent classification error rate for each of the 20 language pairs is shown in Figure 1. The plot again shows that patent BLEU loss does not drop as quickly as the error rate increases. As an approximation, we compute a linear regression (the dotted line) from the observed loss in BLEU as a function of the classification error rate, getting

$$\text{BLEU loss} = -0.048 \times \text{error rate} + 0.351 \quad (8)$$

We did not draw the figure for the generic case because we control (on dev) the generic classification error rate to be mostly under 3% (Table 1), which we think is important for achieving domain adaptation success (e.g., closely approaching Oracle bounds of both domains).

[3]Recall that, in experiments L6 and L7, the patent LM is used for both generic and patent, which are treated as the same genre.

Domain	Classification Error Rate (%)		BLEU (%) Loss	
	dev	test	dev	test
generic	3.0	2.8	-0.25	-0.18
patent	12.4	14.5	-0.43	-0.35

Table 4: Genre classification error rates vs. (C8.1) BLEU loss on average (over 20 language pairs). BLEU loss = C8.1 BLEU-Oracle BLEU. Numbers are computed from Tables 1 and 2.

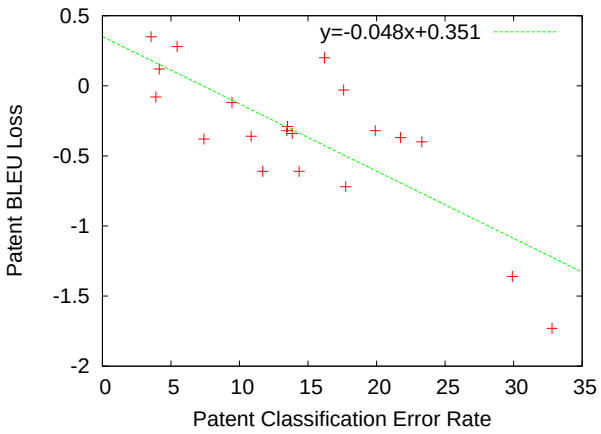

Figure 1: Patent classification error rates vs. patent BLEU loss (or distance to upper bound) for 20 language pairs, respectively. Patent BLEU loss = C8.1 BLEU - upper bound.

We then examine how using a real classifier in tuning (rather than a perfect/cheating one) affects the tuning quality, we run experiment tune-Oracle (results in Table 2) that uses a perfect classifier in tuning and a real classifier in testing. The comparison among C8.1, tune-Oracle and Oracle indicates that, empirically, using a real classifier in tuning (C8.1) yields as good tuning quality as using a perfect one (tune-Oracle).

7.5 Tuning objectives vs. BLEU

Due to concerns about the potential length-penalty effects on genre-aware tuning/decoding (motivated by Table 3 in (Chiang et al., 2008)), we perform experiment C8.2 and compare it with C8.1 to know the impact of different genre-aware tuning objectives (Section 6) on the generic BLEU and the patent BLEU. C8.1 uses the max BLEU sum objective in Eq. (7) and C8.2 uses the max joint BLEU objective in Eq. (5).

- C8.1: L6 + classifier + max BLEU sum

	Generic BLEU		Patent BLEU	
	dev	test	dev	test
C8.1	30.17	29.98	40.77	39.82
C8.2	29.93_{+0-5}	29.81_{+0-5}	40.72_{+0-1}	39.84_{+2-1}

Table 5: The impact of different genre-aware tuning objectives on generic BLEU and patent BLEU, respectively. Each BLEU is an average over BLEUs of 20 language pairs. BLEU_{+m-n}: m is the number of language pairs for which C8.2 is statistically significant better than C8.1; n is the number that is worse; The rest $20 - m - n$ language pairs are those in which C8.1 and C8.2 do not significantly differ.

- C8.2: L6 + classifier + max joint BLEU

C8.1 produces generic/patent feature weights by maximizing the (separate) BLEU of the (classified) generic/patent dev portion in a single MERT run. C8.2 jointly obtains feature weights for both domains by optimizing the BLEU of the entire development set. In either experiment, the weights that MERT produces have two subsets, each for a domain, while any T1-L7 experiment produces just one subset of features that are shared by both domains.

In Table 5, C8.1 vs. C8.2 shows that tuning to maximize the BLEU of each domain development set (via the max BLEU sum objective) has a similar average effect to maximizing the overall BLEU of the entire combined development set (via the max joint BLEU objective), with the former having a slightly better generic translation accuracy. We perform significance test using the paired bootstrap resampling: For 5 language pairs, on both dev and test, C8.1 performs better ($p < 0.005$) than C8.2 for generic translation; and for the rest 15 language pairs, there is no significant difference; For patent translation, C8.1 and C8.2 seem to differ only for one or two language pairs. This further confirms the overall similarity between the two tuning objectives and hints their slight difference. This similarity could be attributed to the fact that, in our approach, features are made decoupled across different domains in tuning, so that the C8.2 tuning is still aware of genre even if it is maximizing the BLEU of the entire development corpus.

8 Conclusions

Most work in domain adaptation for statistical machine translation are focused on only one domain.

In this paper, we introduce a domain adaptation infrastructure to makes a single MT system capable of providing adapted, close-to-upper-bound domain translation accuracy and preserves the generic translation accuracy at the same time. Our approach uses a single translation model and generalizes the traditional single-domain decoding and tuning to deal with different domains in a single system. We use a large number of experiments to demonstrate the viability of our approach.

We use this approach to adapt large-scale generic MT systems for 20 language pairs for patent translation. Our results show that we achieve improved patent translation accuracy that is 0.35 BLEU points away from its upper bound, by sacrificing only 0.18 BLEU points for generic translation.

We explore simple but effective ways of using domain resources, showing domain accuracy improvements made by the use of bilingual training data, domain development data and domain language models. We show the importance of using large amounts of generic training data, particularly in the case where the domain detection error rates for different domains are unbalanced. We also investigate the correlation between genre classification errors and BLEU loss, and the impacts of different genre-aware tuning objectives on BLEUs.

The topic on effectively building multi-domain MT systems have been remained understudied in previous work. We present an improved approach to this problem, study it in a variety of dimensions and show that it is practically working.

Acknowledgments

The authors would like to thank the European Patent Office for the parallel patent data, Daisy Stanton for her initial work on this topic, the reviewers for their helpful comments and suggestions and Jakob Uszkoreit for suggesting the domain phrase ratio feature.

References

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Kr. Naskar, Andy Way, and Josef van Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *Proceedings of AMTA 2010*.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP 2008*, pages 610–619.

David Chiang, Steve DeNeefe, and Michael Pust. 2011. Two easy improvements to lexical weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 455–460, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL 2007*, pages 256–263.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 128–135.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of EMNLP 2010*, pages 451–459.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of EMNLP 2008*, pages 725–734.

Franz Och. 2003. Minimum error rate training for machine translation. In *Proceedings of ACL 2003*.

Jia Xu, Yonggang Deng, Yuqing Gao, and Hermann Ney. 2007. Domain dependent statistical machine translation. In *Proceedings of the MT Summit XI*, pages 515–520.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of HLT-NAACL 2009*, pages 245–253.