The Regression Model of Machine Translation

by

Mehmet Ergun Biçici

Dissertation submitted to the
Department of Computer Engineering
for fulfillment of the requirements for the degree of
Doctor of Philosophy
at
KOÇ UNIVERSITY

August, 2011

Thesis Supervisor: Deniz Yuret

August, 16, 2011

Koç University
Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of the dissertation
for the degree of Doctor of Philosophy by

Mehmet Ergun Biçici

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Date: August, 16, 2011

The Regression Model of Machine Translation

by

Mehmet Ergun Biçici

Dissertation submitted to the Department of Computer Engineering
for fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Machine translation is the task of automatically finding the translation of a source sentence in the target language. Statistical machine translation (SMT) use parallel corpora or bilingual paired corpora that are known to be translations of each other to find a likely translation for a given source sentence based on the observed translations. The task of machine translation can be seen as an instance of estimating the functions that map strings to strings.

Regression based machine translation (RegMT) approach provides a learning framework for machine translation, separating learning models for training, training instance selection, feature representation, and decoding. We use the transductive learning framework for making the RegMT approach computationally more scalable and consider the model building step independently for each test sentence. We develop training instance selection algorithms that not only make RegMT computationally more scalable but also improve the performance of standard SMT systems. We develop better training instance selection techniques than previous work from given parallel training sentences for achieving more accurate RegMT models using less training instances.

We introduce L_1 regularized regression as a better model than L_2 regularized regression for statistical machine translation. Our results demonstrate that sparse regression models are better than L_2 regularized regression for statistical machine translation in predicting target features, estimating word alignments, creating phrase tables, and generating translation outputs. We develop good evaluation techniques for measuring the performance of the RegMT model and the quality of the translations. We use F_1 measure, which performs good when evaluating

translations into English according to human judgments. F_1 allows us to evaluate the performance of the RegMT models using the target feature prediction vectors or the coefficients matrices learned or a given SMT model using its phrase table without performing the decoding step, which can be computationally expensive.

Decoding is dependent on the representation of the training set and the features used. We use graph decoding on the prediction vectors represented in n-gram or word sequence counts space found in the training set. We also decode using Moses (Koehn et al., 2007) after transforming the learned weight matrix representing the mappings between the source and target features to a phrase table that can be used by Moses during decoding. We demonstrate that sparse L_1 regularized regression performs better than L_2 regularized regression in the German-English translation task and in the Spanish-English translation task when using small sized training sets. Graph based decoding can provide an alternative to phrase-based decoding in translation domains having small sized vocabulary.

Thesis Advisor: Deniz Yuret
Title: Assistant Professor
Date: August, 16, 2011

Otomatik Çeviride Regresyon Modeli

Mehmet Ergun Biçici

Bilgisayar Mühendisligi Bölümü'nde doktora derecesinin
gereklerinin tamamlanması için hazırlanmış
doktora tezi

Özet

Otomatik çeviri, bir dilde yazılmış bir cümlenin başka bir dildeki çevirisini otomatik olarak bulma işidir. İstatistiksel otomatik çeviri (SMT) birbirlerinin çevirisi olduğu bilinen paralel dökümanlar veya çiftdilde dökümanlar kullanarak verilen bir cümleye en uygun gelecek çeviriyi önceden görülmüş çevirileri kullanarak hesaplamaya çalışır. Otomatik çeviri işi kelime dizimlerinden kelime dizimlerine eşleştirebilen fonksiyonların hesaplanması örneği olarak görülebilir.

Regresyon tabanlı otomatik çeviri (RegMT) yaklaşımı otomatik çeviriye öğrenme modellerini, öğrenme örnekleri seçimini, özellik gösterimini, ve çeviriyi yaratmayı ayıran bir ögrenme platformu sağlar. Transdüktif öğrenme platformu RegMT yaklaşımını sayısal olarak daha hesaplanabilir yapar ve her test cümlesi için bağımsız olarak model kurar. Geliştirdiğimiz öğrenme örnekleri seçim algoritmaları RegMT yaklaşımını sayısal olarak daha hesaplanabilir yapmanın yanında standart SMT sistemlerinin performansını arttırır. Paralel öğrenme cümlelerinden önceki işlerden daha iyi cümle seçme metodları geliştirerek daha doğru RegMT modellerini daha az öğrenme cümlesi kullanarak elde edebiliyoruz.

Otomatik çeviri için L_1 düzenli regresyon tekniğini L_2 düzenli regresyon tekniğinden daha iyi bir model olarak sunuyoruz. Elde ettiğimiz sonuçlar seyrek regresyon modellerinin L_2 düzenli regresyon modelinden hedef özellikleri tahmin ederken, kelime eşleşmelerini bulurken, kelime dizimi tabloları oluştururken, ve çeviri yaratırken daha iyi olduğunu göstermektedir. RegMT modelinin performansını ve çevirilerin kalitesini ölçmek için iyi ölçüm teknikleri gelistirdik. İngilizceye çevirileri ölçerken insanlar tarafından performansı iyi bulunan F_1 ölçüsünü kullanıyoruz. F_1 bizim RegMT modellerinin performansını hedef özellik tahmin vektörlerini veya

öğrenilen katsayı matrislerini veya verilen bir SMT modelini kendi kelime dizimi tablolarını kullanarak, hesaplaması pahalı olabilen çeviri adımını uygulamadan ölçmemize olanak sağlar.

Çeviri, ögrenme kümesinin gösterimine ve kullanılan özelliklere bağlıdır. Biz ögrenme kümesinin bulundugu n-gram veya kelime dizisi sayıları uzayında gosterilen tahmin vektörleri üzerinde grafik çevirisi kullanıyoruz. Ayrica Moses (Koehn et al., 2007)'ı kullanarak çeviri sırasında kaynak ve hedef özellikler arasındaki eşleştirmeleri gösteren öğrenilmiş ağırlık matrisini Moses tarafından kullanılabilecek kelime dizimi tablosuna dönüştürdükten sonra çeviri yapıyoruz. Seyrek L_1 düzenli regresyonun L_2 düzenli regresyondan Almanca-İngilizce çeviri işinde ve küçük öğrenme kümeleri kullanırken İspanyolca-İngilizce çeviri işinde daha iyi olduğunu gösteriyoruz. Grafik tabanlı çeviri kelime dizimi tabanlı çeviriye az kelime hazineli çeviri işlerinde alternatif olabilir.

Tez Danışmanı: Deniz Yuret
Ünvan: Yardımcı Doçent
Tarih: August, 16, 2011

Acknowledgments

I would like to thank my family for their support during all the hardship of my PhD. They showed patience, love, and economic support during my PhD.

I would like to thank my advisor, Deniz Yuret, for helpful discussions and for guidance and support during the term of this thesis work. I am fortunate to share common interests with him. I am also thankful to Kemal Oflazer for giving valuable feedback and advice about my thesis and to Attila Gürsoy for being attentive and an excellent person to work with.

I am grateful to Koç University for generously offering full scholarship during my PhD. This thesis was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) through a short term research project and through a PhD scholarship award during 2006-2010.

During my graduate student work at Koç University, I served as a teaching assistant for many courses and professors. I would like to thank Deniz Yuret, Attila Gürsoy, Serdar Taşıran, Roy Küçükateş, and İlker Oyman for accepting me as their teaching assistant for their course work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine translation is the task of automatically finding the translation of a source sentence in the target language. Statistical machine translation (SMT) use parallel corpora or bilingual paired corpora that are known to be translations of each other to find a likely translation for a given sentence based on the observed translations. The task of machine translation can be seen as an instance of estimating the functions that map source strings to target strings. The machine translation problem requires special attention for constraining and regularizing the learning models and guiding the outputs of the translation for a smooth and natural performance.

This work focuses on the investigation of the regression-based machine translation model (RegMT) as an alternative to the current phrase-based statistical machine translation systems. RegMT approach provides a learning framework for machine translation, separating learning models for training, training instance selection, feature representation, and decoding.

Learning: We use L_2 regularized regression and sparse regression techniques including L_1 regularized regression to predict the target features for given input source features. We also investigate other learning models and compare with the models in the regression framework. We develop good evaluation techniques that allow us to measure the performance of the learned models or the predicted feature vectors before

actually performing the decoding, which can be computationally expensive.

Instance Selection: Regression is a computationally demanding learning model. We use the transductive learning framework for making the RegMT approach computationally more scalable and consider the model building step independently for each test sentence. For achieving more accurate RegMT models using less training instances, we develop better training instance selection techniques than previous work from given parallel training sentences. Machine translation can be considered as a data intensive learning problem. If you have the right translations in your parallel training sentences, then the translation task can be easier.

Decoding: Decoding is dependent on the representation of the training set and the features used. The features representing text are generated by string feature mappers, known as string kernels, which can incorporate various features defined on string sequences. We use graph decoding on the prediction vectors represented in n-gram or word sequence counts space found in the training set. We also decode using Moses (Koehn et al., 2007) after transforming the learned weight matrix representing the mappings between source and target features to a phrase table that can be used by Moses during decoding.

1.1 RegMT Publications

This work builds on several publications, which are referenced in the chapters that follow. We list RegMT system publications below:

- *L_1 Regularization for Learning Word Alignments in Sparse Feature Matrices* (Biçici and Yuret, 2010a): We learn word alignments using L_1 regularized regression and interpret the weight matrix as the phrase table and show the effectiveness of using sparse regression models for word and phrase alignment.

- *L_1 Regularized Regression for Reranking and System Combination in Machine Translation* (Biçici and Yuret, 2010b): We show that regression mapping is effective in reranking translation outputs and in selecting the best system combinations with encouraging results on different language pairs.

- *Adaptive Model Weighting and Transductive Regression for Predicting Best System Combinations* (Biçici and Kozat, 2010): We analyze adaptive model weighting techniques for reranking using candidate translation scores obtained by L_1 regularized transductive regression for translation outputs generated by different translation models. Without any pre-knowledge of the performance of the translation models, we succeed in achieving the performance of the best model in all translation experiments and surpass their performance in most of the language pairs we considered in an online machine translation scenario.

- *Instance Selection for Machine Translation using Feature Decay Algorithms* (Biçici and Yuret, 2011a): We present an empirical study of instance selection techniques for machine translation. Feature decay algorithms increase the diversity of the training set by devaluing features that are already included and achieve performances that exceed the baseline using smaller training data than the baseline system.

- *RegMT System for Machine Translation, System Combination, and Evaluation* (Biçici and Yuret, 2011b): We show that L_1 regularized regression performs better than L_2 regularized regression and other learning models we compared when measuring the quality of the prediction vectors and the word alignment performance. We present encouraging results when translating from German to English and Spanish to English.

1.2 Thesis Outline

Regression based machine translation model (RegMT) is a learning framework for machine translation that we develop, aiming the separation of training instance selection, feature representation, the learning algorithm, and decoding. RegMT uses regression to estimate the mappings between source and target features. We give the outline of the sections of this thesis below:

- **Chapter 2: Statistical Machine Translation Problem.**

  Presents and reviews the statistical machine translation problem, model training, decoding, evaluation, and phrase-based statistical machine translation work flow.

- **Chapter 3: Regression Based Machine Translation.**

  We model machine translation as an instance of estimating the functions that map source features to target features and use regression to learn the mappings. We give an overview of the RegMT work flow.

  Appendix A gives an overview of the linear regression model and the least squares estimation.

- **Chapter 4: Sparse Regression for Statistical Machine Translation.**

  We define and use sparse regression techniques including L_1 regularized regression model for learning and we demonstrate their usefulness in an example word alignment task.

- **Chapter 5: Instance Selection for Machine Translation using Feature Decay Algorithms.**

  We use transductive regression techniques to learn mappings between source and target features of given parallel training sentences and use these mappings to generate machine translation outputs.

We present an empirical study of instance selection techniques for machine translation. Selection of training instances relevant to the test set improves the final translation quality as in transductive learning and decreases human effort by identifying the most informative sentences for translation as in active learning. Feature decay algorithms increase the diversity of the training set by devaluing features that are already included. We evaluate the best instance selection methods trained with a Moses baseline SMT system using the whole 1.6 million sentence English-German section of the Europarl corpus. We demonstrate that feature decay algorithms perform better than previous work both by selecting more relevant instances and by obtaining a higher BLEU performance. We show that selecting the best 3000 training sentences for a specific test sentence is sufficient to obtain a score within 1 BLEU of the baseline, using 5% of the training data is sufficient to exceed the baseline, and a $\sim$2 BLEU improvement over the baseline score is possible by optimally selected subset of the training data. In out-of-domain translation, we are able to reduce the training set size to about 7% and achieve similar performance as the baseline.

- **Chapter 6: L_1 Regularized Regression for Reranking and System Combination in Machine Translation.**

  We use L_1 regularized transductive regression to learn mappings between source and target features of the training sets derived for each test sentence and use these mappings to rerank translation outputs. The results show the effectiveness of using L_1 regularization versus L_2 regularization used in ridge regression. We show that regression mapping is effective in reranking translation outputs and in selecting the best system combinations and we obtain statistically significant improvements over the baseline system results with the RegMT system on multiple language pairs.

- **Chapter 7: Adaptive Model Weighting and Transductive Regression for Reranking Machine Translation Outputs.**

We analyze adaptive model weighting techniques for reranking using candidate translation scores obtained by L_1 regularized transductive regression for translation outputs generated by different translation models. Competitive statistical machine translation is an online learning technique for sequential translation tasks where we try to select the best among competing statistical machine translators. The competitive predictor assigns a probability per model weighted by the sequential performance. We define additive, multiplicative, and loss-based weight updates with exponential loss functions for competitive statistical machine translation. Without any pre-knowledge of the performance of the translation models, we succeed in achieving the performance of the best model in all translation experiments and surpass their performance in most of the language pairs we considered.

- **Chapter 8: Prediction, Evaluation, and Decoding with RegMT.**

  We show that L_1 regularized regression performs better than L_2 regularized regression in the regression measurements as measured by the prediction performance and obtain comparable performance in the translation experiments. We present encouraging results when translating from German to English and Spanish to English using graph decoding. Especially, when the vocabulary size is low, we observe that graph decoding can achieve the performance of Moses or perform better. We also present translation results when the phrase table of a phrase-based decoder is replaced with the mappings we find with the regression model.

- **Chapter 9: Conclusion.**

  We present an overview of the RegMT model and list our research contributions and findings. We also discuss some future directions.

- **Appendix A: Linear Regression Model and Least Squares**

Estimation.

Appendix A gives an overview of the linear regression model and the least squares estimation. We discuss regularized least square, its dual representation, and stochastic least squares estimation.

- **Appendix B: Statistical Significance Testing of Results.**

  We give the details of statistical significance testing in general and as it is used for SMT in Appendix B.

1.3 Research Contributions

This thesis investigates techniques for making the RegMT model more practical by:

- using transductive learning when building the RegMT learning model,
- developing better training data selection techniques that improves the relevance of the selected instances and reduces the training set size,
- building better regression models that fits the sparse nature of the translation problem,
- creating translation performance evaluation metrics that fit our learning approach better,
- evaluating the performance at various stages of the learning process and performing comparisons with a phrase-based decoder,
- investigating decoding alternatives including graph decoding using the target feature prediction vectors obtained with the regression model.

We make the following main research contributions:

- **RegMT is useful for reranking:** We show that regression mapping score can be used to improve over a baseline SMT system by reranking the N-best lists generated by it.

- **RegMT is useful in online SMT:** We develop adaptive learning models for online SMT problems and achieve the performance of the best model in the system combination challenge and we are able to surpass its performance as well as RegMT's performance with some of the weight update models we considered.
- **Better Training Instance Selection Techniques:** We develop instance selection algorithms that not only make RegMT computationally more scalable but also improve the performance of SMT systems.
- **Sparse RegMT:** We use sparse regression models for statistical machine translation and achieve better target feature predictions and word alignment performance than other learning models. We use F_1 measure, which performs good when evaluating translations into English according to an evaluation by human judgments. F_1 measure allows us to evaluate the performance of the RegMT models using the target feature prediction vectors or the coefficients matrices learned or a given SMT model using its phrase table without performing the decoding step, which can be computationally expensive.
- **Decoding with RegMT:** We show that sparse regression models are better than other learning models we compared for statistical machine translation in predicting target features, estimating word alignments, creating phrase tables, and generating translation outputs. Our graph based decoding experiments demonstrate that sparse L_1 regularized regression performs better than L_2 regularized regression in the German-English translation task as well as in the Spanish-English translation task, which has small sized training set and low vocabulary size. Graph based decoding can provide an alternative to phrase-based decoding in translation domains having small sized vocabulary.

Chapter 2

The Statistical Machine Translation Problem

This section presents the statistical machine translation problem and discusses phrase-based statistical machine translation (SMT).

In machine translation, we are interested in automatically finding a target sentence containing the same information as the source sentence. Given a source sentence, $\mathbf{x}$, from source language $\mathcal{L}_X$, find a target sentence, $\mathbf{y}$, in the target language $\mathcal{L}_Y$, conveying *approximately* the same information:

$$\texttt{info}_{\mathcal{L}_X}(\mathbf{x}) \approxeq \texttt{info}_{\mathcal{L}_Y}(\mathbf{y}).$$

We define $\texttt{info}_L(.)$ as a function returning the information content of its argument sentence in language L.

Phrase-based statistical machine translation approaches the problem as follows. Given a source sentence $\mathbf{x}_1^J = x_1, \ldots, x_j, \ldots, x_J$ containing the words x_j with J words, which is to be translated into a target sentence $\mathbf{y}_1^I = y_1, \ldots, y_i, \ldots, y_I$ containing the words y_i with I words, the *maximum aposteriori* (MAP) translation attempts to find the most likely translation for a given source sentence:

$$\hat{\mathbf{y}}_1^I = \arg\max_{\mathbf{y}_1^I} \; Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J) \tag{2.1}$$

where $\mathbf{x}_1^J$ is the source sentence to be translated and $Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J)$ gives the posterior probability that $\mathbf{y}_1^I$ is the translation of $\mathbf{x}_1^J$. Equation 2.1

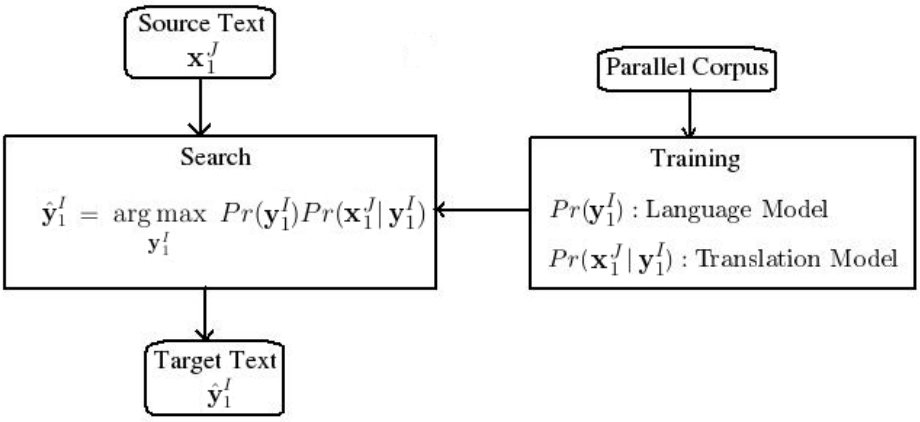

Figure 2.1: The noisy channel model of SMT.

can be rewritten by using the Bayes' rule:

$$\begin{aligned} \hat{\mathbf{y}}_1^I &= \arg\max_{\mathbf{y}_1^I} Pr(\mathbf{y}_1^I)Pr(\mathbf{x}_1^J \mid \mathbf{y}_1^I)/Pr(\mathbf{x}_1^J), \\ &\approx \arg\max_{\mathbf{y}_1^I} \underbrace{p(\mathbf{y}_1^I)}_{\text{increased}} \underbrace{p(\mathbf{x}_1^J \mid \mathbf{y}_1^I)}_{\text{modularity}}, \end{aligned} \tag{2.2}$$

where $Pr(\mathbf{x}_1^J)$ in the denominator is eliminated since it acts as a constant and $p(.)$ is a model-based probability distribution approximating the general probability distribution $Pr(.)$. The Bayesian literature refers to $Pr(\mathbf{x}_1^J)$ as the *evidence*, $Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J)$ as the *posterior*, $Pr(\mathbf{x}_1^J \mid \mathbf{y}_1^I)$ as the *likelihood*, and $Pr(\mathbf{y}_1^I)$ as the *prior*. Equation 2.2 offers an alternative model which also makes use of the probability of $\mathbf{y}_1^I$ in the estimation. This yields a modular approach in which two different knowledge sources, namely the language model and the translation model, that are trained independently are used instead of modeling only $Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J)$.

Equation 2.2 is sometimes referred to as the noisy channel model (Knight, 1999b), since $\mathbf{y}_1^I$ is hypothetically corrupted by some "noise" and turned into $\mathbf{x}_1^J$. Figure 2.1 presents the noisy channel model of SMT. $Pr(\mathbf{y}_1^I) \approx p(\mathbf{y}_1^I)$ corresponds to the target language model score evaluated on the target sentence $\mathbf{y}_1^I$, which is higher for frequently observed sentences in

the target language and it is usually estimated using the target side of the parallel training sentences or a large external corpus in the target language. $Pr(\mathbf{x}_1^J|\mathbf{y}_1^I) \approx p(\mathbf{x}_1^J|\mathbf{y}_1^I)$ corresponds to the translation model score evaluated on both the source ($\mathbf{x}_1^J$) and the target sentence ($\mathbf{y}_1^I$), which assigns higher probability to sentences that are accepted to be the translations of each other and it is estimated using the bilingual corpora.

SMT problem can be considered as an instance of structured learning, which learn models in structured space or in the space of joint correlations and constraints (Taskar, 2004). Structured learning models are commonly applied to sequence labeling problems where the output can be a set of values like characters in words in the optical character recognition problem (Nguyen and Guo, 2007) or words in a sentence as in the machine translation problem.

2.1 Parallel Corpus and Parallel Sentences for Training

SMT systems harvest parallel corpora using statistical techniques. Parallel corpora resources include books that are translated to different languages, parliament discussions, web sites, etc., documents that are translated in more than one language. $\mathcal{L}_X$ and $\mathcal{L}_Y$ represents the source and the target languages respectively. A *parallel corpus* is a paired bilingual corpus, $(\mathcal{C}_X, \mathcal{C}_Y)$, where $\mathcal{C}_X$ denotes the source language corpus and $\mathcal{C}_Y$ denotes the target language corpus such that $\mathcal{C}_Y$ is the translation of $\mathcal{C}_X$. Since the translation could have been done out of order or lossy, the task of *sentence alignment* is to find a mapping function, $\mathbf{m} : \mathcal{C}_X \rightarrow \mathcal{C}_Y$, such that a set of sentences $S_Y \subseteq \mathcal{C}_Y$ where $S_Y = \mathbf{m}(S_X)$ is the translation of a set of sentences $S_X \subseteq \mathcal{C}_X$. Then, under the mapping $\mathbf{m}$, we can use S_Y whenever we use S_X. The pair (S_X, S_Y) forms a *parallel sentence*, which may contain more than two sentences and can be used for training SMT models.

Figure 2.2 depicts a given parallel text and a given sentence alignment found in it. The mappings need not necessarily be 1-to-1, mono-

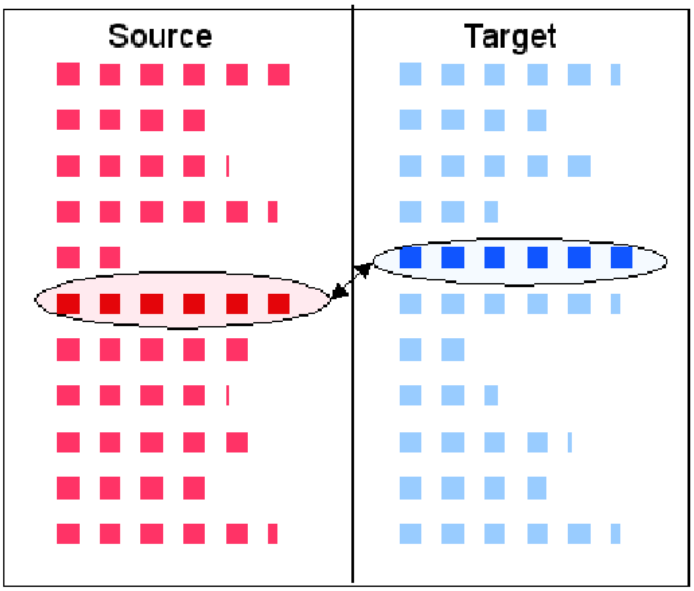

Figure 2.2: Sentence alignment instance in some parallel text.

tonic, or continuous. Sentence alignment is an important preprocessing step that affects the quality of parallel text. Our previous efforts in sentence alignment led to the development and investigation of context-based sentence alignment techniques (Biçici, 2007, 2008) and learning techniques for sentence alignment. From now on we assume that for a given parallel corpus the alignment of sentences is known or given. We refer to the parallel sentences obtained from a given parallel corpus and their representation as a line by line paired parallel sentences as *parallel training sentences*.

2.2 Training

Training is typically based on the *maximum likelihood estimation* (MLE) of the parameters ($p(\mathcal{D}|\Theta)$ for training data $\mathcal{D}$ and parameter set Θ). The language model, $Pr(\mathbf{y}_1^I) \approx p_\gamma(\mathbf{y}_1^I) = p(\mathbf{y}_1^I|\gamma)$, depends on parameters γ and the translation model, $Pr(\mathbf{x}_1^J \mid \mathbf{y}_1^I) \approx p_\theta(\mathbf{x}_1^J \mid \mathbf{y}_1^I) = p(\mathbf{x}_1^J \mid \mathbf{y}_1^I, \theta)$, depends on parameters θ. Given parallel training sentences containing m sentences, $\mathcal{T} = (\mathbf{X}_1^m, \mathbf{Y}_1^m) = (\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$,

training proceeds as follows (Och and Ney, 2002):

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} \prod_{i=1}^{m} p_\theta(\mathbf{x}_i \mid \mathbf{y}_i), \\ \hat{\gamma} &= \arg\max_{\gamma} \prod_{i=1}^{m} p_\gamma(\mathbf{y}_i), \end{aligned} \tag{2.3}$$

The arg max rule for determining the translation for a source sentence $\mathbf{x}_1^J$ becomes:

$$\hat{\mathbf{y}}_1^I = \arg\max_{\mathbf{y}_1^I} \; p_{\hat{\gamma}}(\mathbf{y}_1^I) p_{\hat{\theta}}(\mathbf{x}_1^J \mid \mathbf{y}_1^I), \tag{2.4}$$

MLE model served as the general form of the initial SMT models (Brown et al., 1993). Och and Ney (2002) argue that it is not straightforward to add additional dependencies into Equation 2.4 and a different combination of the models may produce better results.

Maximum entropy training retains a set of M feature functions $h_k(\mathbf{x}_1^J, \mathbf{y}_1^I)$, $k = 1, \ldots, M$, where for each feature function, there exists a feature function weight or model scaling factor λ_k. The direct translation probability is given by:

$$\begin{aligned} Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J) &\approx p_{\boldsymbol{\lambda}_1^M}(\mathbf{y}_1^I \mid \mathbf{x}_1^J) \\ p_{\boldsymbol{\lambda}_1^M}(\mathbf{y}_1^I \mid \mathbf{x}_1^J) &= \frac{\exp\left[\sum_{k=1}^{M} \lambda_k h_k(\mathbf{x}_1^J, \mathbf{y}_1^I))\right]}{\sum_{\mathbf{y'}_1^J} \exp\left[\sum_{k=1}^{M} \lambda_k h_k(\mathbf{x}_1^J, \mathbf{y'}_1^I)\right]} \end{aligned} \tag{2.5}$$

Equation 2.5 corresponds to a *softmax* operation and it can be used to combine multiple models and can also be referred to as the *multiple logistic* model (Ripley, 1996). Estimating the maximum $p_{\boldsymbol{\lambda}_1^M}(\mathbf{y}_1^I \mid \mathbf{x}_1^J)$ is known as the *logistic discrimination* (Ripley, 1996). Since the results of softmax are in the range $[0, 1]$ and taking its derivative is easy, it is used in many applications. Figure 2.3 present SMT model with feature functions trained with the maximum entropy model. The new decision

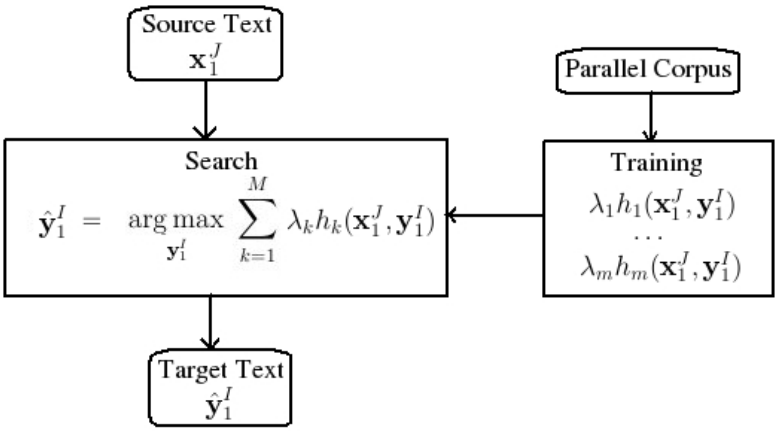

Figure 2.3: SMT with maximum entropy model of training.

rule becomes:

$$\begin{aligned} \hat{\mathbf{y}}_1^I &= \arg\max_{\mathbf{y}_1^I} \left\{ Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J) \right\}, \\ &= \arg\max_{\mathbf{y}_1^I} \left\{ \log Pr(\mathbf{y}_1^I \mid \mathbf{x}_1^J) \right\}, \\ &\approx \arg\max_{\mathbf{y}_1^I} \sum_{k=1}^{M} \lambda_k h_k(\mathbf{x}_1^J, \mathbf{y}_1^I) \end{aligned} \tag{2.6}$$

Note that the MLE formulation given in Equation 2.4 is a special case of the maximum entropy model (Equation 2.6) (i.e. when we use two feature functions $h_1(\mathbf{y}_1^I, \mathbf{x}_1^J) = \log p_{\hat{\gamma}}(\mathbf{y}_1^I)$ and $h_2(\mathbf{y}_1^I, \mathbf{x}_1^J) = \log p_{\hat{\theta}}(\mathbf{x}_1^J \mid \mathbf{y}_1^I)$ with $\lambda_1 = \lambda_2 = 1$). In fact, maximum likelihood and maximum entropy provide dual solutions (Berger et al., 1996).

The maximum class posterior probability criterion is used as the training criterion for the maximum entropy model (Och and Ney, 2002):

$$\hat{\boldsymbol{\lambda}}_1^M = \arg\max_{\boldsymbol{\lambda}_1^M} \left\{ \sum_{i=1}^{m} \log \; p_{\boldsymbol{\lambda}_1^M}(\mathbf{y}_i \mid \mathbf{x}_i) \right\} \tag{2.7}$$

Such optimization of the posterior probabilities directly in Bayes decision rule is referred as *discriminative training*.

There are two main machine learning models: generative ($p_\Theta(\mathbf{y}_1^I, \mathbf{x}_1^J)$) and discriminative (i.e. $\boldsymbol{\lambda}_1^M \mathbf{h}_1^M(\mathbf{y}_1^I, \mathbf{x}_1^J)$). Models that are based on the

distribution of inputs and outputs are referred as *generative models* since they allow the generation of synthetic datasets by sampling (Bishop, 2006). Generative models are computationally demanding since they involve finding the joint distribution. Discriminative models use discriminant functions to discriminate among labels. *Discriminative methods* model the posterior probabilities directly and lead to better predictive performance (Bishop, 2006). Generative models can handle missing values and sequences of varying length for hidden Markov models and discriminative models generally perform better than generative models on discriminative tasks (Bishop, 2006). These modeling techniques are parametric methods since the probability distributions that govern the data are based on a small number of parameters. The probabilistic conditional models ($p_\Theta(\mathbf{y}_1^I|\mathbf{x}_1^J)$) do not use $p_\Theta(\mathbf{y})$ and use conditional feature distributions to determine the label.

The trend in SMT training is towards purely discriminative approaches. Perceptron (Liang et al., 2006) or large margin training (Watanabe et al., 2007) techniques are used to handle the large number of features used in these models. Regression model attempts to find mappings between features of inputs and outputs and the learned coefficients matrix fits in the discriminative learning framework. The regression approach in kernel induced spaces that we discuss in chapter 3 provides a general discriminative approach where we can use kernels from generative models in a discriminative framework.

2.3 Decoding

Once we are finished with optimizing the parameters, what is left is the following decision rule:

$$\begin{aligned} \hat{\mathbf{y}}_1^I &= \arg\max_{\mathbf{y}_1^I} \sum_{k=1}^{M} \hat{\lambda}_k h_k(\mathbf{x}_1^J, \mathbf{y}_1^I) \\ &= \arg\max_{\mathbf{y}_1^I} \hat{\boldsymbol{\lambda}}^T \mathbf{h}(\mathbf{x}_1^J, \mathbf{y}_1^I) \\ &\approx \arg\max_{\mathbf{y}_1^I \in \texttt{GEN}(\mathbf{x}_1^J)} \langle \hat{\boldsymbol{\lambda}}, \mathbf{h}(\mathbf{x}_1^J, \mathbf{y}_1^I) \rangle \end{aligned} \tag{2.8}$$

where $\texttt{GEN}(.)$ in the last equation produces a finite enumeration of all valid target sentences for a given source sentence and we assume that $\texttt{GEN}(\mathbf{x}_1^J) \subseteq \mathcal{L}_Y$. When $\texttt{GEN}(\mathbf{x}_1^J) = \mathcal{L}_Y$, last equation in Equation 2.8 becomes an equality.

2.3.1 Computational Complexity of Statistical Machine Translation

Decoding can be thought of being composed of two problems: selection of appropriate words or phrases in the target language and ordering them properly. Both of these two problems cause SMT decoding to be NP-complete (Knight, 1999a). Knight performs a reduction from source word ordering problem to Hamilton circuit problem and reduces the problem of selecting concise set of target phrases to minimum set cover problem, both of which are NP-complete problems. Similarly, finding the most likely alignment of tokens between the source sentence $\mathbf{x}$ and the target sentence $\mathbf{y}$ using Viterbi alignment:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} p(\mathbf{x}, \mathbf{a}|\mathbf{y}) \tag{2.9}$$

as well as the expectation maximization (EM) parameter estimation of IBM models 3, 4, and 5 (Brown et al., 1993) and the conditional probability estimation of the following form:

$$p(\mathbf{x}|\mathbf{y}) = \sum_{\mathbf{a}} p(\mathbf{x}, \mathbf{a}|\mathbf{y}) \tag{2.10}$$

are shown to be $\#P$-complete problems and the exact decoding problem of the following form:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_{\mathbf{a}} p(\mathbf{x}, \mathbf{a}|\mathbf{y}) p(\mathbf{y}) \tag{2.11}$$

is shown to be $\#P$-hard problems (Udupa and Maji, 2006).

If we concentrate on Equation 2.8 again, we note that the problem in the search approach using enumeration over possible sentences is mainly due to the need to check the value of exponentially large number of sentences. Heuristic search techniques such as beam search, greedy search, and A^* search are generally employed to search for a translation.

2.3.2 Reranking

An N-best list stores the top N best scoring translations generated by an SMT system the top of which is returned as the translation. However, there can be discrepancies in the orderings achieved by human judgments of this list and the ordering present in the output N-best list. Multiple explanations can be given for this discrepancy:

1. Some complex features that can better evaluate the quality of a translation may be computationally complex to apply on the whole training set and instead a reranking or rescoring approach to the generated N-best list might be more appropriate.

2. Trained parameters might correspond to a local optimum and therefore may not model the performance well enough.

3. The training corpus used might belong to a domain that is different than the test set and therefore statistical models built and the parameters learned may not be good enough to be applicable on the test set. Assuming that training and test sets are sampled from the same distribution can be a good simplification in theory but in practice, this is rarely the case.

4. The learned models can fit the training data perfectly but perform poorly on the test set as there can be many functions that explain the training set. This is called *over-fitting* in machine learning and it is also related to the *bias-variance trade-off*, which models the expected loss of different models obtained from the same dataset by resampling via the difference between the average model and the true distribution (bias), the difference between the average model and individual models (variance), and the inherent noise in the dataset.

The opportunity loss that results from the existence of a better translation among the translations produced by an SMT system is called model error (Lopez, 2007). Some systems rerank or rescore the generated N-best list by using additional scoring functions that have the potential of ranking the best translation as the top of the list. The scores obtained from these new feature functions can be interpolated with the original translation scores of the SMT system.

2.4 Translation Performance Evaluation

We give the definition of two commonly used translation evaluation techniques: NIST (Doddington, 2002) and BLEU (Papineni et al., 2001). NIST and BLEU scores use n-gram co-occurrence counts derived from the whole set of sentences in the translated documents. BLEU score is defined for order n as follows:

$$\log \text{BLEU} = \min(1 - r/t, 0) + \sum_{i=1}^{n} \lambda_i \log p_i \tag{2.12}$$

The first term is the brevity penalty, which computes a penalty for the difference between the sum of reference sentence lengths (r) versus the sum of translation sentence lengths (t) in terms of the number of tokens. The positive weights λ_i are chosen as $1/i$ (Papineni et al.,

2001). p_i is the modified i-gram precision term defined as:

$$p_i = \frac{\sum_{p=1}^{i} \sum_{\text{pgram} \in \text{matched-pgrams}} \text{count(pgram)}}{\sum_{p=1}^{i} \sum_{\text{pgram} \in \text{translation-pgrams}} \text{count(pgram)}}, \tag{2.13}$$

where p-gram matches for $1 \leq p \leq i$ are computed for reference and translation sentences and these are divided by the overall p-gram occurrences in the translation sentences. As can be seen from the formulation, BLEU score is not symmetric and to prevent degenerate translations, it makes sense to divide by the overall counts in the translation sentences. Thus, BLEU may not be the best choice to evaluate the performance on the sentence level.

NIST score (Doddington, 2002) weights n-grams that occur less frequently more using the information coming from n-gram counts calculated as follows:

$$\text{info}(w_1, \ldots, w_n) = \log_2 \left(\frac{\text{count}(w_1, \ldots, w_{n-1})}{\text{count}(w_1, \ldots, w_n)} \right), \tag{2.14}$$

where w_i corresponds to a word in n-gram $w_1, \ldots, w_n$. The final NIST score becomes:

$$\text{NIST} = \sum_{p=1}^{n} \left\{ \frac{\sum_{\text{pgram} \in \text{matched-pgrams}} \text{info(pgram) count(pgram)}}{\sum_{\text{pgram} \in \text{translation-pgrams}} \text{count(pgram)}} \right\} \exp\left\{\beta \log^2 \min(r/t, 1)\right\} \tag{2.15}$$

where β is a constant chosen to make the brevity penalty 0.5 when $r/t = 2/3$.

2.5 Phrase-based Statistical Machine Translation Work Flow

This section gives an overview of the phrase-based statistical machine translation work flow. The initial step is sentence alignment, which maps the sentences of a parallel corpus to parallel units where each line is a translation pair, one on the source side and one on the target side. We obtain parallel training sentences from a parallel corpus after sentence alignment. Then, a word alignment step allows the generation of a phrase table to be used during translation. Some parallel training

sentences are set aside as development sentences for tuning. Model tuning and translation generation or decoding steps follow. Parallel training sentences as well as monolingual corpora are used to develop language models to be used during decoding.

The work flow is given in Figure 2.4. Additional details on the mathematical models used can be found in (Knight, 1999b). Lopez (2007) gives a survey of statistical machine translation.

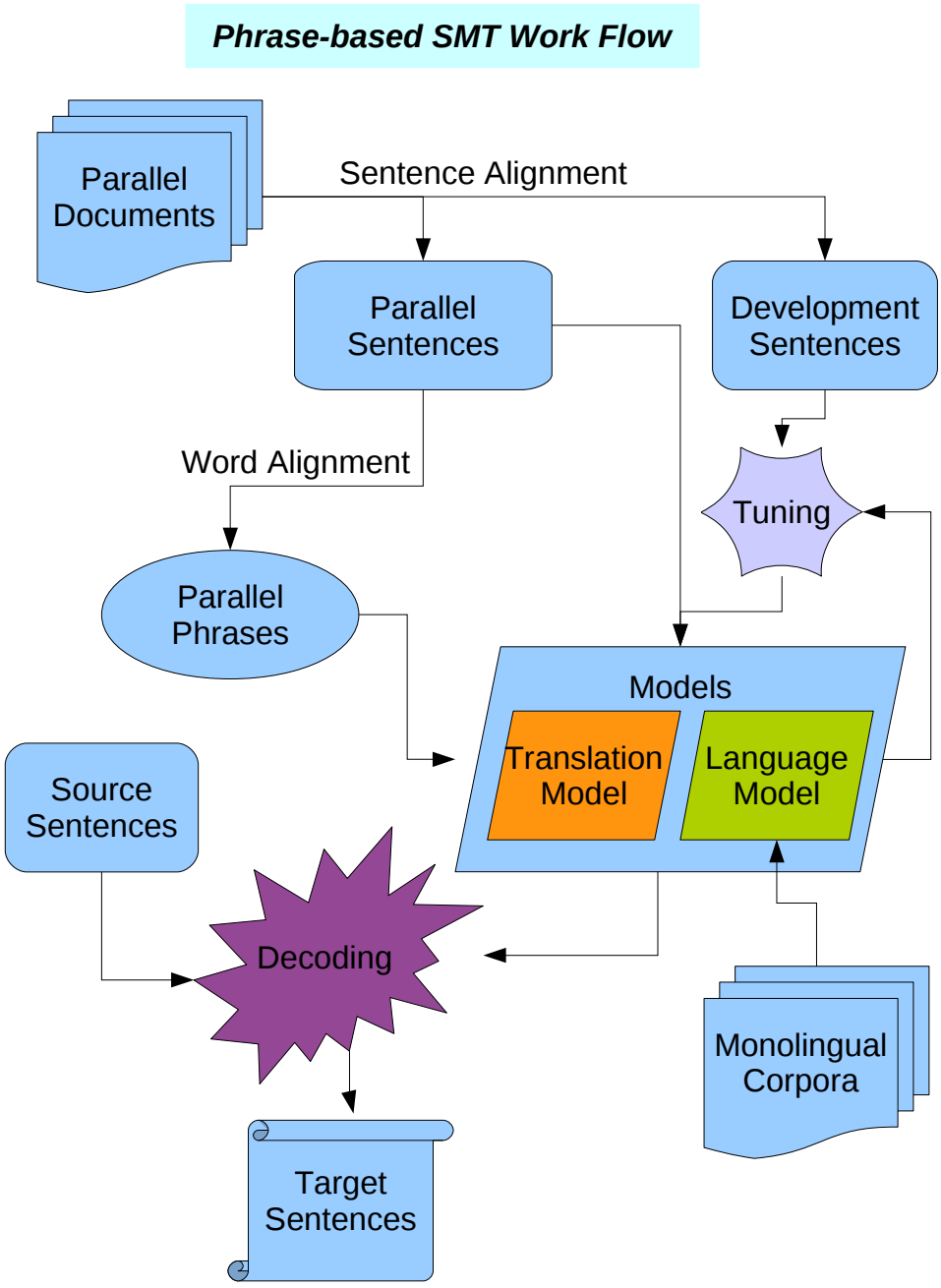

Figure 2.4: Phrase-based SMT work flow.

Chapter 3

Regression Based Machine Translation

Machine translation is the task of automatically finding the translation of a source sentence in the target language. The task of machine translation can be seen as an instance of estimating the functions that map strings to strings. Assuming that X and Y correspond to the sets of tokens that can be used in the source and target strings, a training data of m inputs, $\mathcal{T}$, which may be generated by an underlying generative model, can be represented as:

$$\mathcal{T} = (\mathbf{X}_1^m, \mathbf{Y}_1^m) = \Big\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\Big\} \subseteq X^* \times Y^*,$$

where each parallel training instance $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}$ corresponds to a pair of source and target language token sequences for $1 \leq i \leq m$.

Our goal is to find a mapping $f : X^* \rightarrow Y^*$ that can convert a given source sentence to a corresponding target sentence sharing the same meaning or containing the same information in the target language; hence, f is the many-to-many mapping that we are trying to find. Regression is generally used to estimate a mapping function between a real valued covariate, $\mathbf{x} \in X^*$, and a real valued response variable, $\mathbf{y} \in Y^*$ using labels in $\mathbb{R}^N$ for $N \geq 1$. We can define feature mappings as $\Phi_X : X^* \rightarrow F_X = \mathbb{R}^{N_X}$ and $\Phi_Y : Y^* \rightarrow F_Y = \mathbb{R}^{N_Y}$ that map each string sequence to a point in high dimensional real space where the dimensions $dim(F_X) = N_X$ and $dim(F_Y) = N_Y$. Note that N_X and

N_Y are determined based on the number of features generated via Φ_X and Φ_Y respectively on the training data. This scenario is depicted in Figure 3.1 (also presented in (Cortes et al., 2007)).

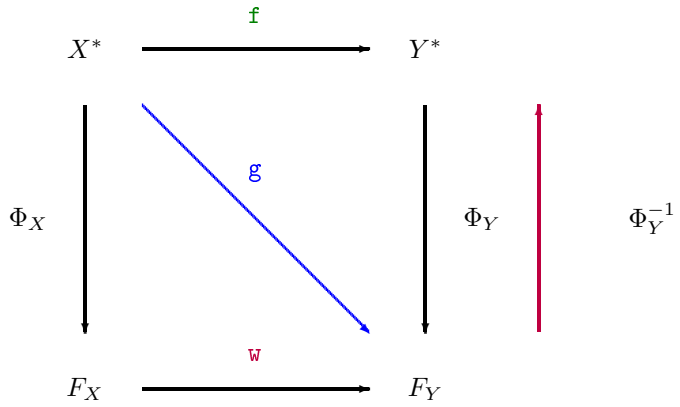

Figure 3.1: String-to-string mapping.

In Figure 3.1, **f** is the mapping we are trying to find, **w** is the mapping we learn after transforming the training set to a high dimensional feature space, and **g** is the mapping we can use, which also transforms the source to the new space in which the training data lives. The following are two of the main problems involved as depicted in Figure 3.1:

- **Regression problem:** The problem of learning $g : X^* \rightarrow F_Y$ that maps a given source sentence to a point in the feature space of the target language. Again, this is a many-to-many mapping.
- **Pre-image problem:** Given the features of the estimated target string sequence, $g(\mathbf{x}) = \Phi_Y(\hat{\mathbf{y}})$, find $\mathbf{y} \in Y^*$:

$$f(\mathbf{x}) = \arg\min_{\mathbf{y} \in Y^*} ||g(\mathbf{x}) - \Phi_Y(\mathbf{y})||^2,$$

  which approximates the target sentence when an exact pre-image does not exist ($\Phi_Y^{-1}(g(x)) = \emptyset$).

3.1 Regression problem

In general, regression estimation involves features in real space. We use feature mappings that transform input and output instances to real points in high dimensional spaces: $\mathbb{R}^{N_X}$ and $\mathbb{R}^{N_Y}$. A feature mapping could be the count of observed n-grams in a given sentence. We give the result of using n-gram counts as feature mappers for some example sentences in subsection 3.4.1. F_X and F_Y belong to N_X and N_Y dimensional real spaces. The feature mappings Φ_X and Φ_Y can be used to define positive definite symmetric kernels, k_X and k_Y, that correspond to inner products in Hilbert spaces F_X and F_Y such that for all $\mathbf{x}, \mathbf{x}' \in X^*$, $k_X(\mathbf{x}, \mathbf{x}') = \Phi_X(\mathbf{x}).\Phi_X(\mathbf{x}')$ and $k_Y(\mathbf{y}, \mathbf{y}') = \Phi_Y(\mathbf{y}).\Phi_Y(\mathbf{y}')$.

In a linear regression setting, the linear interpolation function $g(\mathbf{x}) : X^* \rightarrow F_Y$ is defined as follows:

$$g(\mathbf{x}) = \mathbf{W}\Phi_X(\mathbf{x}), \tag{3.1}$$

where $\mathbf{x}$ is an input string, $\mathbf{W} : F_X \rightarrow F_Y$ is a matrix of size $N_Y \times N_X$, and $\Phi_X(\mathbf{x}) \in \mathbb{R}^{N_X \times 1}$ is the representation of $\mathbf{x}$ in F_X. Similarly, $\Phi_Y(\mathbf{y}) \in \mathbb{R}^{N_Y \times 1}$ is the representation of $\mathbf{y}$ in F_Y. [1] Both Φ_X and Φ_Y transform source and target sentences respectively to numerical vectors in high dimensional space.

The regression problem tries to minimize the following regularized loss function:

$$\mathcal{J}(\mathbf{W}) = \sum_{i=1}^{m} \| \Phi_Y(\mathbf{y}_i) - \mathbf{W}\Phi_X(\mathbf{x}_i) \|^2 + \lambda \| \mathbf{W} \|_F^2, \tag{3.2}$$

for $\mathbf{W} \in \mathbb{R}^{N_Y \times N_X}$. We can think of the loss function as a Lagrangian function with λ being the Lagrange multiplier for the constraint, specifying that the norm of $\mathbf{W}$ will be close to zero. This penalty ensures that the learned coefficients do not have large values. Let $\mathbf{M}_X \in \mathbb{R}^{N_X \times m}$ and $\mathbf{M}_Y \in \mathbb{R}^{N_Y \times m}$ represent the training data in matrix form such that $\mathbf{M}_X = [\Phi_X(\mathbf{x}_1), \ldots, \Phi_X(\mathbf{x}_m)]$ and $\mathbf{M}_Y = [\Phi_Y(\mathbf{y}_1), \ldots, \Phi_Y(\mathbf{y}_m)]$.

[1] Appendix A gives an overview of the linear regression model and the least squares estimation.

Then, our regularized least squares regression problem can be rewritten as:

$$\mathbf{W} = \underset{\mathbf{W}' \in \mathbb{R}^{N_Y \times N_X}}{\arg\min} \underbrace{\| \mathbf{M}_Y - \mathbf{W}'\mathbf{M}_X \|_F^2}_{\text{training error}} + \underbrace{\lambda \| \mathbf{W}' \|_F^2}_{\text{regularization penalty}} . \quad (3.3)$$

Here, the norm becomes the Frobenius norm since we sum the squared norms of the individual errors where the Frobenius norm of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ is defined as follows (Trefethen and Bau, III, 1997, lecture 3):

$$\| \mathbf{A} \|_F^2 \triangleq \sum_{i=1}^{n} \sum_{j=1}^{m} A_{i,j}^2 = \langle \mathbf{A}, \mathbf{A} \rangle_F = \mathrm{tr}(\mathbf{A}^T\mathbf{A}) = \mathrm{tr}(\mathbf{A}\mathbf{A}^T) = \sum_{i=1}^{p} \sigma_i^2, \quad (3.4)$$

where $\langle \mathbf{A}, \mathbf{A} \rangle_F$ is the Frobenius inner product that induces the norm and σ_i corresponds to the i^{th} singular value of $\mathbf{A}$.

Proposition 1. *The solution to our cost function given in Equation 3.3 can be found by the following identities:*

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{M}_Y \boldsymbol{M}_X^T (\boldsymbol{M}_X \boldsymbol{M}_X^T + \lambda \boldsymbol{I}_{N_X})^{-1} && \textit{(primal)} \\ \boldsymbol{W} &= \boldsymbol{M}_Y (\boldsymbol{K}_X + \lambda \boldsymbol{I}_m)^{-1} \boldsymbol{M}_X^T && \textit{(dual)} \end{aligned} \quad (3.5)$$

where $\boldsymbol{K}_X = \boldsymbol{M}_X^T \boldsymbol{M}_X$ *is the* $m \times m$ *Gram matrix with* $\boldsymbol{K}_X(i,j) = k_X(\boldsymbol{x}_i, \boldsymbol{x}_j)$ *and* $k_X(\boldsymbol{x}_i, \boldsymbol{x}_j)$ *is the kernel function defined as* $k_X(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi_X(\boldsymbol{x}_i)^T \Phi_X(\boldsymbol{x}_j)$.

Proof. Since $\mathcal{J}(\mathbf{W})$ is convex and differentiable we can find the $\mathbf{W}$ that minimizes it by taking the derivative:

$$\begin{aligned} (\mathbf{M}_Y - \mathbf{W}\mathbf{M}_X)\mathbf{M}_X^T &= \lambda \mathbf{W} \\ \mathbf{M}_Y \mathbf{M}_X^T &= \mathbf{W}\mathbf{M}_X\mathbf{M}_X^T + \lambda \mathbf{W} \\ \mathbf{M}_Y \mathbf{M}_X^T &= \mathbf{W}(\mathbf{M}_X\mathbf{M}_X^T + \lambda \mathbf{I}_{N_X}) \\ \mathbf{W} &= \mathbf{M}_Y \mathbf{M}_X^T (\mathbf{M}_X\mathbf{M}_X^T + \lambda \mathbf{I}_{N_X})^{-1} \end{aligned} \quad (3.6)$$

This derivation is also presented by Cortes et al. (2007). Equation 3.6 becomes the primal solution. We can invert the primal solu-

tion given in Equation 3.6 by using the matrix inversion lemma (Equation A.34) to obtain the dual solution:

$$\begin{aligned}
\mathbf{W} &= \mathbf{M}_Y\mathbf{M}_X^T(\mathbf{M}_X\mathbf{M}_X^T + \lambda\mathbf{I}_{N_X})^{-1} \\
&= \mathbf{M}_Y\mathbf{M}_X^T\left[\lambda^{-1}\mathbf{I}_{N_X} - \lambda^{-1}\mathbf{M}_X(\mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)^{-1}\lambda^{-1}\mathbf{M}_X^T\right] \\
&= \mathbf{M}_Y\left[\lambda^{-1}\mathbf{M}_X^T - \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X(\mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)^{-1}\lambda^{-1}\mathbf{M}_X^T\right] \\
&= \mathbf{M}_Y\left[\mathbf{I}_m - \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X(\mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)^{-1}\right]\lambda^{-1}\mathbf{M}_X^T \\
&= \mathbf{M}_Y\left[\mathbf{I}_m - (-\mathbf{I}_m + \mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)(\mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)^{-1}\right]\lambda^{-1}\mathbf{M}_X^T \\
&= \mathbf{M}_Y(\mathbf{I}_m + \lambda^{-1}\mathbf{M}_X^T\mathbf{M}_X)^{-1}\lambda^{-1}\mathbf{M}_X^T \\
&= \mathbf{M}_Y(\lambda\mathbf{I}_m + \mathbf{M}_X^T\mathbf{M}_X)^{-1}\mathbf{M}_X^T \qquad (3.7) \\
&= \mathbf{M}_Y(\mathbf{K}_X + \lambda\mathbf{I}_m)^{-1}\mathbf{M}_X^T \qquad (3.8)
\end{aligned}$$

□

We see that the regularization term also prevents the Gram matrix, $\mathbf{K} = \mathbf{M}_X^T\mathbf{M}_X$, in the dual formulation or the covariance matrix, $\mathbf{C} = \mathbf{M}_X\mathbf{M}_X^T$, in the primal formulation from being singular. The regularization term prevents the normal equations to be singular (i.e. $\lambda\mathbf{I}$ term in $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ makes the whole matrix invertible (Taylor and Cristianini, 2004)) and prevents coefficients to have large values by diverging too far away from the diagonal matrix.

3.2 Pre-image problem

The pre-image problem in regression mapping model corresponds to the problem of decoding in machine translation where we try to find the translation from a target feature prediction vector. The pre-image problem involves predicting the target string $f(\mathbf{x}) \in Y^*$ for a given source string $\mathbf{x} \in X^*$ such that the corresponding translation is close: $\mathbf{y} \in Y^*$, $\Phi_Y(\mathbf{y}) \approx g(\mathbf{x}) = \mathbf{W}\Phi_X(\mathbf{x})$. We approximate the features of the target string when an exact pre-image does not exist ($\Phi_Y^{-1}(g(x)) =$

$\emptyset$):

$$
\begin{aligned}
f(\mathbf{x}) &= \arg\min_{\mathbf{y}\in Y^*} \| \Phi_Y(\mathbf{y}) - \mathbf{W}\Phi_X(\mathbf{x}) \|^2 \\
f(\mathbf{x}) &= \arg\min_{\mathbf{y}\in Y^*} (\Phi_Y(\mathbf{y})^T\Phi_Y(\mathbf{y}) - 2\Phi_Y(\mathbf{y})^T\mathbf{W}\Phi_X(\mathbf{x})) \\
f(\mathbf{x}) &= \arg\min_{\mathbf{y}\in Y^*} (\Phi_Y(\mathbf{y})^T\Phi_Y(\mathbf{y}) - 2\Phi_Y(\mathbf{y})^T\mathbf{M}_Y(\mathbf{K}_X + \lambda\mathbf{I}_m)^{-1}\mathbf{M}_X^T\Phi_X(\mathbf{x})) \\
f(\mathbf{x}) &= \arg\min_{\mathbf{y}\in Y^*} (k_Y(\mathbf{y},\mathbf{y}) - 2(\mathbf{K}_Y^{\mathbf{y}})^T(\mathbf{K}_X + \lambda\mathbf{I}_m)^{-1}\mathbf{K}_X^{\mathbf{x}}), \qquad (3.9)
\end{aligned}
$$

where $\mathbf{K}_Y^{\mathbf{y}} \in \mathbb{R}^{m\times 1}$ and $\mathbf{K}_X^{\mathbf{x}} \in \mathbb{R}^{m\times 1}$ are defined as follows:

$$
\mathbf{K}_Y^{\mathbf{y}} = \begin{bmatrix} k_Y(\mathbf{y},\mathbf{y}_1) \\ \dots \\ k_Y(\mathbf{y},\mathbf{y}_m) \end{bmatrix} \text{ and } \mathbf{K}_X^{\mathbf{x}} = \begin{bmatrix} k_X(\mathbf{x},\mathbf{x}_1) \\ \dots \\ k_X(\mathbf{x},\mathbf{x}_m) \end{bmatrix}. \qquad (3.10)
$$

As we can see, in the dual formulation, $\mathbf{W}$ need not be involved in the computations and kernel functions can be used instead and the term $(\mathbf{K}_X + \lambda\mathbf{I}_m)^{-1}$ can be calculated once and reused. When Φ_Y corresponds to polynomial kernels of odd degree, then Φ_Y becomes invertible and the pre-image problem becomes trivial (Cortes et al., 2007).

3.3 Related Work

Regression techniques can be used to model the relationship between strings (Cortes et al., 2007). Wang et al. (2007) applies a string-to-string mapping approach to machine translation by using ordinary least squares regression and n-gram string kernels on a small subset of the Europarl corpus. Later they use L_2 regularized least squares regression (Wang and Shawe-Taylor, 2008). Although the translation quality they achieve is not better than Moses (Koehn et al., 2007), which is accepted to be the state-of-the-art, they show the feasibility of the approach.

Serrano et al. (2009) use kernel regression to find translation mappings from source to target feature vectors and experiment with trans-

lating in a constrained hotel front desk requests domain. Ueffing et al. (2007) approaches the transductive learning problem for SMT by bootstrapping the training using the translations produced by the SMT system that have a scoring performance above some threshold as estimated by the SMT system itself. Transductive SVMs (Joachims, 1999) learn a hyperplane that separates both the labeled and the unlabeled data.

Online large margin training techniques are being used to incorporate millions of features for SMT (Watanabe et al., 2007). Structured learning techniques are proposed as another technique for SMT (Daumé III, 2006). In the context of regression, transduction can be considered similar to weighted mixed regression. Rao and Toutenburg (1999) define weighted mixed regression to handle missing values in the data where another regression model fills in the missing values, a process called imputation or repair. Locally weighted regression (LWR) solves separate weighted least squares problems for each instance (Hastie et al., 2009), weighted by a kernel similarity function.

We observe that L_1 reularized regression is effective in learning mappings between sparse features versus L_2 used in ridge regression on the word and phrase alignment task of machine translation (Biçici and Yuret, 2010a).

3.4 Practical Issues

In this section, we discuss techniques for making the RegMT approach more practical.

3.4.1 Kernel Implementation

Feature mapping can be done by using an n-spectrum word kernel defined as:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{n} \sum_{i=1}^{|\mathbf{x}|-p+1} \sum_{j=1}^{|\mathbf{x}'|-p+1} I(\mathbf{x}[i:i+p-1] = \mathbf{x}'[j:j+p-1]) \quad (3.11)$$

where $\mathbf{x}[i:j]$ denotes a substring of $\mathbf{x}$ composed of the words in the range $[i:j]$ and $I(.)$ is the indicator function. It considers all word sequences of order p. Note that the basis functions corresponding to this kernel are count vectors corresponding to all of the word n-grams that are found in the parallel training sentences. We give example feature vectors obtained using the 2-spectrum word kernel on two sample source sentences in Table 3.1.

$\mathbf{x}_1$ = "the book is on the table"

$\mathbf{x}_2$ = "I saw the man"

F_X	the	book	is	on	table	I	saw	man	the book	book is	is on	on the	the table	I saw	saw the	the man
$\phi_X(\mathbf{x}_1)$	2	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0
$\phi_X(\mathbf{x}_2)$	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Table 3.1: Example feature mapping on sample source sentences. Columns correspond to the features, rows correspond to the observation counts.

The first six columns in Table 3.1 correspond to unigram features and the remaining correspond to bigram features. We note that as new instances are added to the set, which can be the source side of the training set, the number of features in the feature vectors increase. The corresponding representation for a training set becomes sparser as new training instances are included. The size of the feature vectors can be proportional to the size of the vocabulary of the training set, $O(V)$ for vocabulary size V.

Another string kernel that can be used is the weighted word n-gram string kernel defined as:

$$k(\mathbf{x},\mathbf{x}') = \sum_{p=1}^{n} \sum_{i=1}^{|\mathbf{x}|-p+1} \sum_{j=1}^{|\mathbf{x}'|-p+1} I(\mathbf{x}[i:i+p-1] = \mathbf{x}'[j:j+p-1]) \exp(-\frac{\alpha}{p}), \tag{3.12}$$

which weighs the similarity based on the length of the string match

and α is a threshold of length below which the similarity is considered as atypical and a smaller similarity score is given. This kernel is similar to the weighted substring kernels defined by Vishwanathan and Smola (2003).

When operating in the Reproducing Kernel Hilbert Spaces (RKHS), the distance between two objects can be calculated by using kernel functions, which can be interpreted as dot products. The distance between two objects, $\mathbf{x}_i, \mathbf{x}_j \in X^*$ can be found as follows given that a kernel function k is already defined (Bakir et al., 2007):

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= \| \Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j) \|_{\mathcal{H}} \\ &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle + \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_j) \rangle \\ &= \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{x}_j, \mathbf{x}_j)}, \end{aligned} \tag{3.13}$$

where $\mathcal{H}$ is the RKHS in which the dot product can be evaluated by kernel functions. Thus, the distance can be defined as the distance between the images obtained after mapping the features by Φ.

3.4.2 Feature Engineering for Machine Translation

In this section, we discuss how to model features and create feature mappers that can benefit the machine translation task between source and target languages. One of the main questions we can ask is which feature space measures the similarities between the sentences better for higher SMT performance and therefore which kernel to choose? Can we automatically select the optimal kernel for a given learning task or automatically learn kernels from data? (KernelLearning, 2008) Can we combine multiple kernels for learning? (Bach et al., 2004) These are all relevant questions we might ask while choosing an appropriate space for learning.

Feature modeling for some languages like Turkish can be a challenge. In morphologically rich languages such as Turkish the stem and suffix forms bear their own regularities. Morphemes, the smallest unit in a given language carrying meaning (Mitkov, 2003), play an important

role in the syntactic dependency structure and they can have long-distance dependencies. The dependencies are not just between stems but also between stems and suffixes. If we use complete words as unit tokens, we will not be able to represent these sub-word dependencies.

Also, for less monotonic languages such as Turkish, the role of the language model can increase as the correct ordering improves BLEU performance. Language models in machine translation are generally used to measure the amount of fluency the translation sentence is expected to have. Language models are also used to constrain the search space used during decoding for selecting appropriate tokens or phrases for phrase based systems. Selection of features become important as a bigram may not model a given word having multiple suffixes. A better language model should be able to mirror the linguistic dependencies as the ones observable in (Hakkani-Tür et al., 2002) more effectively. Morpheme-aware language models such as FlexGrams (Yuret and Biçici, 2009) not only allow syntactic links to be made between different possible phrases that can be used to construct a sentence for phrase-based machine translation systems but can also help overcome the distortion cost used in SMT systems to sort out sentences that are highly irregular in their respective orderings relative to the source sentence.

Previous work shows that the effect of finding the correct morphological form for a given stem can increase the BLEU score by 8 when translating from English to Czech (Koehn et al., 2006). For translating into Turkish, word repair heuristic is being used to increase the BLEU for finding the correct morphological form (Oflazer, 2008). Error-tolerant finite state recognition (Oflazer, 1996) helps generate morphological alternatives for a word which can then be rescored using separate word and morpheme language models to pick the best alternative for the purpose of morphological disambiguation.

3.4.3 Decoding with the RegMT Model

The options we consider for decoding with the RegMT model include:

1. Re-ranking Moses (Koehn et al., 2007) outputs with the translation scores we obtain with the RegMT model. This option is discussed and explored in chapter 6 and in chapter 7.

2. Interpreting $\mathbf{W}$ as the phrase table and using Moses to perform decoding (discussed in chapter 8).

3. Using graph decoding (discussed in chapter 8).

3.5 Computational Complexity

In this section, we discuss computational complexity of the RegMT model and techniques for making the RegMT approach computationally more scalable. The computational complexity of the primal or dual solutions given in Theorem 1 depend largely on the matrix inversion, which is in $O(n^3)$ for n representing the dimension of the square matrix. If we use the primal solution, then it is in $O(N_X^3)$ and if we use the dual solution, it is in $O(m^3)$.

In the inductive learning setting where we build a model using all of the training set, primal solution can be used after a feature pruning step or using a low rank approximation. To reduce the computational complexity, Cortes et al. (2007) propose finding a matrix $\mathbf{L} \in \mathbb{R}^{m \times n}$, where $n \ll m$, such that $\mathbf{K}_X = \mathbf{L}\mathbf{L}^T$ by using incomplete Cholesky decomposition, which has $O(mn^2)$ complexity to speed up the training. When $\mathbf{K}_X \in \mathbb{R}^{m \times m}$ we can use the matrix inversion lemma (Equation A.34) as follows:

$$(\mathbf{L}\mathbf{L}^T + \lambda \mathbf{I}_m)^{-1} = \lambda^{-1} \left[\mathbf{I}_m - \mathbf{L}(\lambda \mathbf{I}_n + \mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\right]. \tag{3.14}$$

This reduces the cost of the matrix inversion in the dual solution from $O(m^3)$ to $O(n^3)$. More predictive techniques exist for low rank approximations (Bach and Jordan, 2005).

However, for some learning tasks where we have access to the test set beforehand, we need not induce a model from the full training set, which can be computationally demanding. In transductive learning, test instances are used to learn specific models tailored towards the test set. Machine translation fits the transductive learning paradigm well if we have access to the test set. Also, machine translation is usually performed at the sentence level and we can build a learning model specifically for a given test sentence by assuming that sentences in the test set are independent from each other.

3.5.1 Transductive Learning vs. Inductive Learning

Transduction is referred as the learning process by which we move from a set of training examples to points of interest directly without inducing a model from the training set and deducing the value of the points of interest by using the induced models (Vapnik, 2000). Figure 3.2 is from (Vapnik, 2000) and describes different types of inference that are being employed in machine learning. Vapnik argues that in cases where limited amount of information exist, we do not need to solve the more general problem of induction and then deduction.

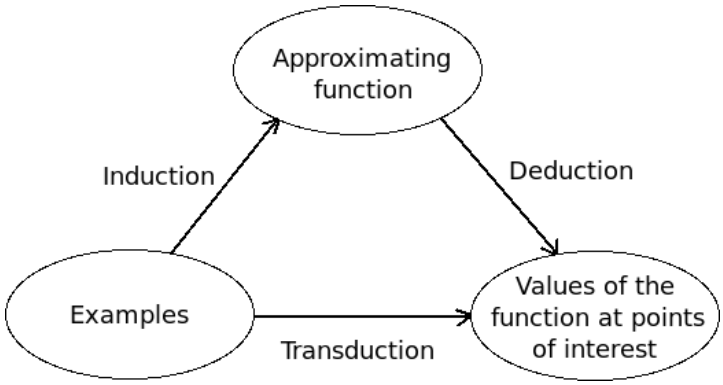

Figure 3.2: Different types of inference.

In our transductive learning setting, the transduced instances are selected from the training set and therefore we have access to their labels as well. In an approach closer to transduction, we can have the SMT

system generate possible translations and use those labels in training as well, a process referred as bootstrapping. The goal in transductive regression based machine translation (RegMT) is both reducing the computational burden of the regression approach by reducing the dimensionality and improving the translation quality by using transduction, which we know to improve the performance of the learning model.

3.6 Training Instance Selection per Source Feature

Proper selection of training instances plays an important role to learn feature mappings with limited computational resources accurately. In previous work (Wang and Shawe-Taylor, 2008), training sentences were selected using *tf-idf* [2] information retrieval technique. We transform test sentences to feature sets obtained by the kernel mapping before measuring their similarities and index the sentences based on the features found sorted by the length of the sentences, enabling us to find shorter matches faster. However, as we see in chapter 5, although selecting the shortest sentences is a good heuristic, it does not give us a training set with a target feature set coverage better than the instance selection methods that we develop in chapter 5: FDA and *dice*. We define coverage as the percentage of test source and target features found in a given training set.

Given a source sentence of length 20 tokens, its feature representation would have 57 total 1/2/3-gram features. If we only include the closest sentences to the set of test instances, then we may not have translations covering all of the features. But if we search for translations of each source feature, then we may have a higher chance of covering all the features found in the test sentence we are trying to translate. The index is used as a dictionary of source phrases storing training set entries whose source sentence match the given source phrase.

[2] tf-idf: term frequency - inverse document frequency.

The selected number of training instances per source feature, $\mathfrak{f}_x$, is chosen inversely proportional to the frequency of the source feature determined by the following formula:

$$\texttt{idfScore}(\mathfrak{f}_x) = \sum_{x \in Y(\mathfrak{f}_x)} \texttt{idf}(x) \tag{3.15}$$

$$\texttt{numTrain}(\mathfrak{f}_x) = \max\left(1, \lceil \frac{c}{\ln(1 + \alpha/\texttt{idfScore}(\mathfrak{f}_x))} \rceil\right), \tag{3.16}$$

where $\texttt{idfScore}(\mathfrak{f}_x)$ sums the *idf* (inverse document frequency) of the tokens in source feature $\mathfrak{f}_x$, $Y(.)$ lists the tokens of a given feature, α is an `idf` score threshold, and c is a small number. [3] `numTrain` tries to select more for rare features; selecting more for common features also works in some cases as it helps in disambiguating the features.

3.7 RegMT Work Flow

This section gives an overview of the RegMT work flow. We highlight the differences with the phrase-based SMT work flow given in Figure 2.4. After a sentence alignment step, we perform training instance selection over the parallel training sentences using the source sentences of the test set. We obtain the training data to be used during training following this procedure. Then we transform the sentences found in the training data to feature matrices using the corresponding feature mappers for source and target language sentences. RegMT learning is performed on the feature matrices to obtain the regression mapping, $\mathbf{W}$. The mapping becomes the translation model and it can be directly used as a phrase table and we see how this transformation is performed in chapter 8. Model tuning and translation generation or decoding steps follow.

[3] We use $\alpha = 9.0$, $c \geq 1$.

3.8 Summary

This chapter described the mathematical background for the RegMT model, the feature representation used, and the pre-image finding problem. We discussed related work and explained techniques for making the RegMT model more practical and computationally more scalable. In subsection 3.4.1, we give an example feature representation for a set of source sentences. The feature representation of a training set constructed with n-spectrum word kernels can become a sparse representation. However, in contrast, the coefficients matrix obtained from solving the problem in Equation 3.3 can still become a dense matrix. In machine translation, we would like to obtain sparser models as we describe in chapter 4, where we discuss techniques for obtaining sparser coefficients matrices that can also be used to create a phrase table for machine translation.

We described our transductive learning approach and how we select training instances per source feature found in each test source sentence. The next section discusses sparse regression models for statistical machine translation, which fit the machine translation task better.

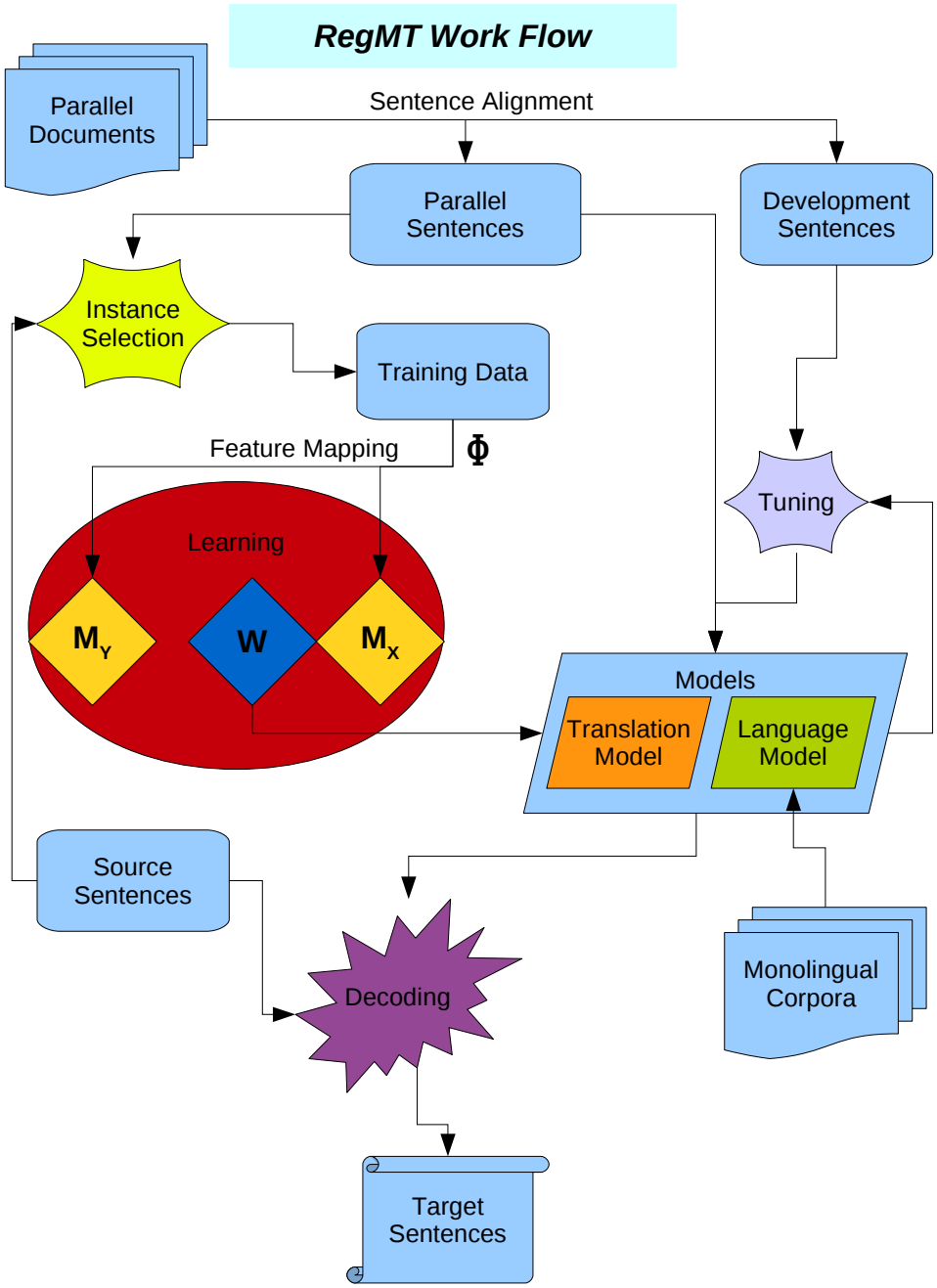

Figure 3.3: RegMT work flow.

Chapter 4

Sparse Regression for Statistical Machine Translation

This chapter introduces sparse regression techniques for statistical machine translation. The feature representations of source and target sentences belonging to a training set used for statistical machine translation can be sparse. Sparse feature representations can also cause the data matrices used to have more unknown features than the number of training instances.

The quadratic regularization penalty used with the L_2 model prevents the normal equations from becoming singular yet most of the coefficients remain non-zero and the L_2 model does not give us a sparse solution. When we perform machine translation, we would like to observe only a few nonzero target feature coefficients corresponding to a source feature in the coefficients matrix. Regularization with the L_1 norm helps us achieve sparse solutions to the mapping problem.

We present various techniques of regression leading to sparse coefficients matrices. We show the effectiveness of L_1 regularized regression or *lasso* to learn the mappings between sparsely observed feature sets versus L_2 regularized regression in an example word alignment scenario. Comparative results on the learning performance of the algorithms with other learning models is given in chapter 8.

Outline: In the first two sections we discuss the sparseness in the data and mappings when modeling source to target feature mappings

for statistical machine translation and show that L_1 norm regularization can help us achieve sparse solutions to the mapping problem. In section 4.3, we introduce the L_1 regularized regression techniques and techniques for L_1 minimization. Then in section 4.4 we give an example word alignment scenario which demonstrates the difference between word alignment matrices obtained using L_1 and L_2 regularized regression models.

4.1 Sparsity in Translation Mappings

In statistical machine translation, parallel training sentences, which contain translations of the same sentences in source and target languages, are used to estimate a likely target translation for a given source sentence based on the observed translations. String kernels lead to very sparse representations of the feature space and we examine the effectiveness of L_1 regularized regression to find the mappings between sparsely observed feature sets.

We would like to observe only a few nonzero target feature coefficients corresponding to a source feature in the coefficients matrix. If the coefficients matrix obtained resembles a dictionary or a phrase table, then it can be useful during machine translation. An example solution matrix representing a possible alignment between unigram source and target features could be the following:

W	x_1	x_2	x_3
y_1	1	1	
y_2		1	
y_3			1

Table 4.1: Sparsity in translation mappings.

Here x_i represents unigram source features and y_i represent unigram target features. x_1 and x_3 have unambiguous translations whereas x_2 is ambiguous. Even if unigram features lead to ambiguity, we expect

higher order features like bigrams and trigrams to help us resolve the ambiguity. Typical $\mathbf{W}$ matrices have thousands of features. L_1 regularization helps us achieve solutions close to permutation matrices by increasing sparsity (Bishop, 2006, page 145). In contrast, L_2 regularized solutions give us dense matrices and when we consider a phrase table, the correct solution is very sparse.

4.2 L_1 Norm Regularization

We describe the L_1 norm and its sparsity inducing characteristic in this section. The quadratic regularization penalty used with the L_2 model helps to ensure that the learned coefficients do not have large values by diverging too far away from the diagonal matrix and prevents the normal equations become singular. Although this term becomes easy to take the derivative, most of the coefficients remain non-zero and it does not give us a sparse solution. We are interested in penalizing the coefficients more effectively; zeroing the irrelevant ones and thereby leading to sparsification. L_1 norm behaves both as a feature selection technique and a method for reducing coefficient values.

We use the following definitions for L_1 vector norm (Trefethen and Bau, III, 1997, lecture 3) and L_1 matrix norm:

$$\| \mathbf{x} \|_1 = \sum_{i=1}^{N} |x_i| \qquad \| \mathbf{X} \|_1 = \sum_{i=1}^{I} \sum_{j=1}^{J} |X_{i,j}|. \tag{4.1}$$

L_1 is a sparsity inducing norm, making some of the coefficients zero (Bach et al., 2011). We show the effectiveness of using L_1 regularization versus L_2 used in ridge regression on the word and phrase alignment task of machine translation (Biçici and Yuret, 2010a). Figure 4.1 depicts an L_1 norm constrained objective function minimization problem in 2D. The figure shows why L_1 norm induces sparsity. L_1 norm constraint forces the function minimum to be found at an edge of the constraint square, which is likely to be at one of the corners of the square, enabling a sparse solution.

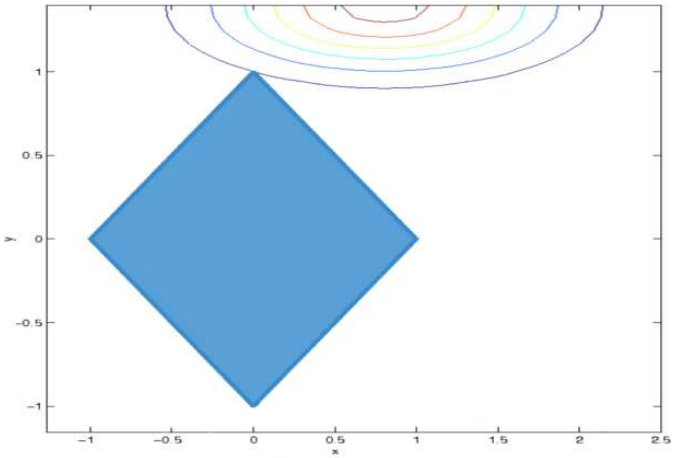

Figure 4.1: L_1 norm in 2D ($|x| + |y| = 1$)

L_1 norm regularized regression is also referred as *lasso* (least absolute shrinkage and selection operator) (Tibshirani, 1996) and the corresponding minimization problem is given below (Hastie et al., 2009):

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \sum_{i=1}^{N} (y_i - w_0 - \sum_{j=1}^{p} x_{ij} w_j)^2 + \lambda \sum_{j=1}^{p} |w_j| \right\} \quad (4.2)$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2 + \lambda \| \mathbf{w} \|_1 \quad (4.3)$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{y}, \mathbf{X}, \mathbf{w}) + \lambda \| \mathbf{w} \|_1 \quad (4.4)$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{y}, \mathbf{X}, \mathbf{w}) \text{ s.t. } \lambda \| \mathbf{w} \|_1 \leq C \quad (4.5)$$

where $\mathbf{w} \in \mathbb{R}^{p \times 1}$, $\mathbf{y} \in \mathbb{R}^{N \times 1}$, and $\mathbf{X} \in \mathbb{R}^{N \times p}$ for a given loss function $\mathcal{L}$. The formulation in Equation 4.5 is a quadratic programming criterion for each C (Hastie et al., 2006).

4.3 L_1 Regularized Regression Techniques

In this section, we give details of the various techniques of regression leading to sparse coefficients matrices. The problem we are trying to solve is learning mappings between *sparse* feature representations of text using a small number of training instances for computational

efficiency. The *lasso* (least absolute shrinkage and selection operator) regression problem is given below:

$$\mathbf{W}_{L_1} = \underset{\mathbf{W}\in\mathbb{R}^{N_Y\times N_X}}{\arg\min} \; \| \mathbf{M}_Y - \mathbf{W}\mathbf{M}_X \|_F^2 \; + \; \lambda \, \| \mathbf{W} \|_1 \, . \tag{4.6}$$

lasso has the property that as we increase the λ, some $W_{i,j}$ shrink towards zero, increasing the sparseness (Bishop, 2006). When the data matrix is sparse, L_1 regularized regression (*lasso*) is known to perform better than L_2 regularized regression in reducing the test set error and it can still perform better in dense settings (Bach, 2008).

We use two *lasso* techniques:

- Forward stagewise regression (FSR): Iteratively increases the weight of the variable most correlated with the residual by ϵ to approximate the *lasso* (Hastie et al., 2006).
- Quadratic programming (QP): Optimizes the *lasso* cost by using quadratic programming to reach the *lasso* solution (Mørup and Clemmensen, 2007).

In Equation 4.6, no analytical solution exists but we know that it is a convex problem and therefore it has a single global minimum. The L_2 regularized regression problem given in Equation 3.3 as well as the L_1 regularized problem given in Equation 4.6 belong to *convex optimization problems* where both the objective and the constraint functions are convex such that they satisfy the following inequality (Boyd and Vandenberghe, 2004):

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, $\alpha \geq 0$, and $f : \mathbb{R}^n \to \mathbb{R}$. Additionally, the affine combination or the sum of two convex functions, such as the training error term and the regularization penalty term, results in a convex function and the locally optimal points in convex optimization problems are also globally optimal (Boyd and Vandenberghe, 2004). Hence, *lasso* is a convex problem with a single global minimum and

we can use optimization to find the minimum. FSR and QP approach the same minimum solution to the cost function.

We also explore the following techniques for L_1 norm minimization, which are closely related:

- L_1 norm minimization using linear programming (LP): Faster than QP but optimizes a different cost than the *lasso*, interpreting the error term as the constraint to satisfy (Chen et al., 1998).
- L_1 norm minimization using iterative thresholding (*iter*-ε): Iterative technique for L_1 norm minimization (Herrity et al., 2006).

When working in high dimensional spaces, we may experience multicollinearity (i.e. some predictors being highly correlated) or redundancy of the coefficients. Therefore, we are interested in penalizing the coefficients more effectively; zeroing the irrelevant ones and thereby leading to sparsification. We discuss in subsection 4.3.5 that for problems with large number of correlated variables, FSR has smoother and monotone coefficient values and rate of increase in the coefficient values with increasing number of iterations whereas the QP solution to *lasso* has fluctuating coefficient values.

4.3.1 *lasso* with Forward Stagewise Regression (FSR):

The incremental forward stagewise regression (FSR) algorithm for approximating the *lasso* is given by Hastie et al. (2006). We use FSR, which approximates the *lasso* faster than quadratic programming. The incremental forward stagewise regression algorithm increases the weight of the predictor variable that is most correlated with the residual by a small amount, ϵ, multiplied with the sign of the correlation at each step. As $\epsilon \to 0$, the profile of the coefficients resemble the *lasso* (Hastie et al., 2009). The corresponding incremental forward stagewise regression algorithm for multivariate data is given in Algorithm 1. We can set $\epsilon = |\mathbf{c}_j|$ for $|\mathbf{c}_j|$ storing the magnitude of the largest current correlation value, which would eliminate the covariates that are correlated

with $\mathbf{x}_j$ (Efron et al., 2004). This would decrease multicollinearity and the approach is referred as forward selection.

In the multivariate case, we have multiple response variables and a regressor variable has varying degrees of correlation with each one. Turlach et al. (2005) view the value of each component of the weight matrix, $\mathbf{W}_{i,j}$ as the explanatory power that the j^{th} regressor variable has on the i^{th} response variable. Therefore, they propose to use $\mathbf{W}_{*,j}^{\max} = \max(|\mathbf{W}_{1,j}|, |\mathbf{W}_{2,j}|, \ldots, |\mathbf{W}_{N_Y,j}|)$ as a measure of the explanatory power of the j^{th} regressor on all the response variables. The corresponding norm is $l_{1,\infty}$:

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \| \mathbf{M}_Y - \mathbf{W}\mathbf{M}_X \|_F^2 + \lambda \| \mathbf{W} \|_{1,\infty} \tag{4.7}$$

This also makes sense in terms of the features obtained from word sequence kernels as we might prefer one translation corresponding to a word sequence above others.

In Algorithm 1, $\mathbf{M}_Y = \{\mathbf{M}_{Y_1}, \ldots, \mathbf{M}_{Y_{N_Y}}\}$ and $\mathbf{M}_X = \{\mathbf{M}_{X_1}, \ldots, \mathbf{M}_{X_{N_X}}\}$. We assume that each target feature is independent and for each of the target feature row vector as represented by $\mathbf{M}_{Y_i}$ we fill a row of $\mathbf{W}$. We repeat steps 6 to 9 until all predictors have a correlation value less than ϵ with $\mathbf{r}$; thus we continue while $maxcorr > \epsilon$. At each iteration, we have a matrix multiplication with a vector to find the correlations, which costs $O(N_X m)$. If the number of repetitions in the inner loop is M, the algorithm is in $O(N_Y N_X M m)$. If we consider the number of repetitions in the inner loop as constant, the algorithm is in $O(N_Y N_X m)$.

4.3.2 *lasso* with Quadratic Programming (QP):

We also use quadratic programming (QP) to find $\mathbf{W}_{L_1}$. We can pose *lasso* as a QP problem as follows (Mørup and Clemmensen, 2007). We assume that the rows of $\mathbf{M}_Y$ are independent and solve for each row i, $\mathbf{M}_{y_i} \in \mathbb{R}^{1\times m}$, using non-negative variables $\mathbf{w}_i^+, \mathbf{w}_i^- \in \mathbb{R}^{N_X \times 1}$ such that $\mathbf{w}_i = \mathbf{w}_i^+ - \mathbf{w}_i^-$:

Algorithm 1: Incremental Forward Stagewise Regression - FSR_ϵ - for Multivariate Data

Input: Source and target feature matrices, $\mathbf{M}_X$ and $\mathbf{M}_Y$, step size, ϵ.

Output: Coefficient matrix $\mathbf{W}$.

$\mathbf{W} \leftarrow \mathbf{0}$

for $i \leftarrow 1$ **to** N_Y **do**

  Let $\mathbf{r} = \mathbf{M}_{Y_i}^T \in \mathbb{R}^{m \times 1}$

  $maxcorr \leftarrow 1$

  while $maxcorr > \epsilon$ **do**

    Let $\mathbf{c} = \mathbf{M}_X \mathbf{r}$ store the correlations where $\mathbf{c}_{j'} = \text{corr}(\mathbf{r}, \mathbf{M}_{Xj'}) = \langle \mathbf{r}, \mathbf{M}_{Xj'} \rangle$. Then, $j = \arg\max_{j'} |\mathbf{c}_{j'}|$

    $maxcorr = |\mathbf{c}_j|$

    Update $\mathbf{W}_{i,j} \leftarrow \mathbf{W}_{i,j} + \delta_j$ for $\delta_j = \epsilon \cdot \text{sign}[\mathbf{c}_j]$

    Update $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{M}_{Xj}$

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \ \| \mathbf{M}_{y_i} - \mathbf{w}\, \mathbf{M}_X \|_F^2 \ + \lambda \sum_{k=1}^{N_X} |w_k|, \tag{4.8}$$

$$\hat{\mathbf{w}}_i = \arg\min_{\tilde{\mathbf{w}}_i} \frac{1}{2}\, \tilde{\mathbf{w}}_i\, \widetilde{\mathbf{M}_X} \widetilde{\mathbf{M}_X}^T \tilde{\mathbf{w}}_i{}^T - \tilde{\mathbf{w}}_i\, (\widetilde{\mathbf{M}_X} \mathbf{M}_{y_i}^T - \lambda \mathbf{1}), \tag{4.9}$$

$$\text{s.t.} \ \ \tilde{\mathbf{w}}_i > 0, \ \ \widetilde{\mathbf{M}_X} = \begin{bmatrix} \mathbf{M}_X \\ -\mathbf{M}_X \end{bmatrix}, \ \ \tilde{\mathbf{w}}_i = \begin{bmatrix} \mathbf{w}_i^{+T} & \mathbf{w}_i^{-T} \end{bmatrix}.$$

Due to using quadratic constraints, QP solution is slower than the FSR solution. Interior-point method for solving quadratic constraints is in $O(N_Y N_X^3 l)$ where l is the required number of bits to represent the data (Potra and Wright, 2000).

4.3.3 L_1 Minimization using Linear Programming (LP):

L_1 minimization can also be posed as a linear programming (LP) problem by interpreting the error term as the constraint (Chen et al., 1998) and solving for each row i.

Let the minimization problem be:

$$\min \| \mathbf{x} \|_1 \text{ subject to } A\mathbf{x} = \mathbf{b}. \tag{4.10}$$

The dual formulation of linear programming solves problems of the form:

$$\max -\mathbf{h}^T\mathbf{z} \text{ subject to } G^T\mathbf{z} + \mathbf{c} = 0 \text{ and } \mathbf{z} \geq 0. \tag{4.11}$$

Then, we can use linear programming by using non-negative variables $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{R}^{N_X \times 1}$ such that $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$ and using the following transformations:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}, \quad G = \begin{bmatrix} A^T \\ -A^T \end{bmatrix}, \quad \text{and} \quad \mathbf{c} = -\mathbf{b}. \tag{4.12}$$

For the multivariate case, we can assume that the rows of $\mathbf{M}_Y$ are independent and solve for each row i, $\mathbf{M}_{y_i} \in \mathbb{R}^{1 \times m}$, using non-negative variables $\mathbf{w}_i^+, \mathbf{w}_i^- \in \mathbb{R}^{N_X \times 1}$ such that $\mathbf{w}_i = \mathbf{w}_i^+ - \mathbf{w}_i^-$. The constraint given in Equation 4.10 may not be satisfiable, in which case we can use the L_2 regularized ridge regression solution.

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} \| \mathbf{w} \|_1 \text{ subject to } \mathbf{M}_{y_i} = \mathbf{w}\mathbf{M}_X, \tag{4.13}$$

which can again be solved using non-negative variables. This is a slightly different optimization problem and the results can be different but linear programming solvers offer computational advantages. We are selecting among the $\mathbf{w}$s that satisfies the constraint having the minimum L_1 norm. If the constraint is not satisfiable, we use the L_2 regularized regression solution.

4.3.4 L_1 Minimization using Iterative Thresholding (*iter*-ε):

When the size of the training matrices increase, the constraint given in Equation 4.10 can become harder to satisfy. In such cases, an iterative solution can be helpful. Iterative thresholding algorithms for sparse approximation iteratively update the coefficient vector after some thresholding of its values (Herrity et al., 2006).

Iterative thresholding assume that $\mathbf{x}$ is sparse with k nonzero entries and A is close to unitary such that $\mathbf{x}_{new} = \delta_k(A^T\mathbf{b}) = \delta_k(A^TA\mathbf{x})$, where $\delta_k(.)$ is a thresholding operator (Maleh, 2009) and n is the step number. The error becomes $\mathbf{y} \approxeq A^TA\mathbf{y} = A^TA(\mathbf{x} - \mathbf{x}_{[n]}) = A^T\mathbf{b} - A^TA\mathbf{x}_{[n]}$. This allows us to find a recurrence relation (Maleh, 2009):

$$\mathbf{x}_{[n]} = \delta_k\left(\mathbf{x}_{[n-1]} + \mu(A^T\mathbf{b} - A^TA\mathbf{x}_{[n-1]}))\right), \tag{4.14}$$

where μ is the step size used for better convergence and $0 < \mu \leq 1$. We select $\delta_k(.)$ such that it retains only the top k entries whose value is greater than 0.

Gradually decreasing the threshold as the number of iterations increase is also used (Drori, 2007). Instead of thresholding, we retain the largest k values, where k is estimated using the number of features found in the source sentence.

4.3.5 Regularization Parameter Optimization

lasso has the property that as we increase the λ, some $W_{i,j}$ shrink towards zero, increasing the sparseness (Bishop, 2006). For L_2 regularization, λ needs to be optimized.

The number of iterations and the ϵ value used for FSR corresponding to a given λ value is harder to calculate. Hastie et al. (2009) show that $M\epsilon \rightarrow \lambda \parallel \mathbf{W} \parallel_1$ where M is the number of iterations and $\epsilon > 0$ is the step size. As $\epsilon \rightarrow 0$, the incremental forward stagewise algorithm, FSR_ϵ, approximates the lasso. However, at each step FSR_ϵ and *lasso* optimize a different criteria. *lasso* optimally reduces the error for unit increase in the L_1 norm of the coefficients. FSR_ϵ is optimal for unit

increase in the L_1 arc length of the coefficients, which is the sum of the L_1 norms of the changes in the coefficient values (Hastie et al., 2009, page 74).

However, we don't know the value of $\| \mathbf{W} \|_1$ beforehand and therefore it is not easy to choose appropriate M and ϵ values that will also achieve fast computation. Instead, we choose the amount of L_1 arc length that we would like to distribute among the various features or we optimize M and ϵ values directly on the development set separately. For each source or target feature, we would like to observe only a few corresponding features in the coefficients matrix.

We also define the correlation used in FSR (Algorithm 1, line 6) as follows to make it scale invariant:

$$corr(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{length(\mathbf{x})} = \frac{\mathbf{x}^T \mathbf{y}}{length(\mathbf{x})}. \tag{4.15}$$

This allows us to use the same M and ϵ value for all sentences regardless of their length. We also limit the total L_1 arc length that is to be distributed by $M\epsilon$ and set its total to around 2.5. This allows us to select few target features for a given source feature.

Hastie et al. (2006) note that for problems with large number of correlated variables, FSR making L_1 arc length minimizing steps can be preferable to *lasso*, which makes L_1 norm minimizing steps. Hastie et al. (2006) show that (in Figure 7, Hastie et al. (2006)), in such cases, FSR's coefficient paths (paths that show the coefficient values and the rate of increase in the coefficient values) are monotone and smoother when compared with the *lasso*, which tend to fluctuate. FSR always makes a step in the direction of the largest correlation with the residual whereas *lasso* makes a step minimizing the L_1 norm, which need not be in the direction of the largest correlation. This explains the case for word sequence kernels where consecutive bigrams are related and unigrams are related to the their parent bigrams. We demonstrate an example word alignment scenario in the following section where FSR is preferable to *lasso* using QP optimization.

4.4 Example Word Alignment Scenario

In this section, we compare the effectiveness of L_1 regularized regression algorithms for the word alignment task on a sample test sentence. In the word alignment problem, we are interested in finding correspondences or alignments between source and target words. This example scenario can also be considered as a small, simplified machine translation exercise where we try to find which source words correspond to which target words. We are trying to align the following German-English sentence pair:

Source: `ich hoffe, dass wir dies hier im europäischen parlament tun werden .`

Target: `i hope that we will do that here in the european parliament .`

We select 1000 training sentences for training the corresponding RegMT models and remove the irrelevant features that are found in the training sets such that we retain only the unigram features found in the source and the target sentences. The coefficients matrix that we obtain using L_2 regularization is given in Figure 4.2. We use Hinton diagrams (Hinton et al., 1986) to depict the coefficients matrices where a white square corresponds to a positive coefficient value and a black square corresponds to a negative coefficient value. The area of each square drawn corresponds the coefficient value at that corresponding location.

As we expected, L_2 regularized regression is giving us a dense solution with most of the entries remaining nonzero. There exists some problems with the coefficients we obtain using the L_2 regularization. For instance the German source word `dass` is mapped to the word `the` on the English side as found by the largest corresponding coefficient value. Also, the word `europäischen` is mapped to the word `the`. From this word alignment matrix, we can obtain phrase pairs using

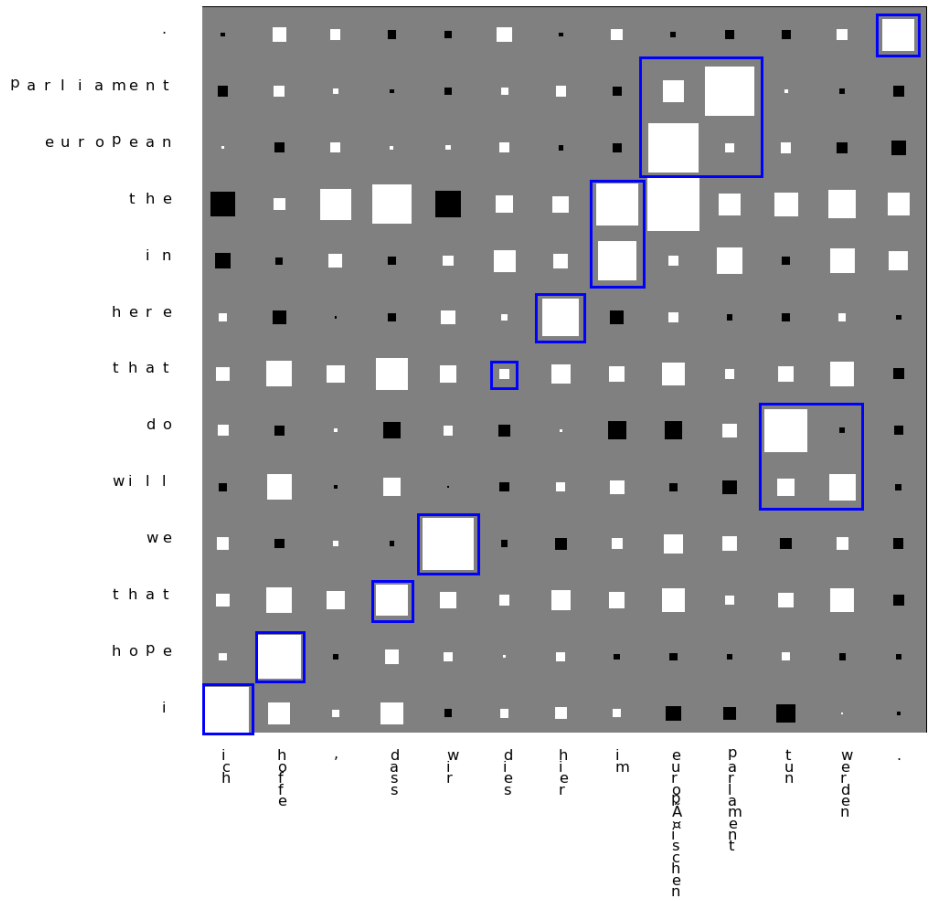

Figure 4.2: L_2 Regularized Regression for word alignment figure, 1000 training instances.

similar heuristics used in Moses (Koehn et al., 2007) such as the grow-diag-final heuristic, which grows the intersection of source to target and target to source alignments with additional alignment points. We form a blue rectangle around possible phrase pairs in Figure 4.2.

Figure 4.3 depicts the coefficients matrix we obtain using QP for L_1 regularized regression. We observe a sparser matrix than the L_2 regularized solution. However, as with the L_2 solution, we observe problems with the alignment in some cases. For instance, the German words `europäischen` and `dass` are mapped to `the` on the English side, which is a function word.

Figure 4.4 depicts the coefficients matrix we obtain using FSR for L_1 regularized regression. We again observe a sparser matrix than the

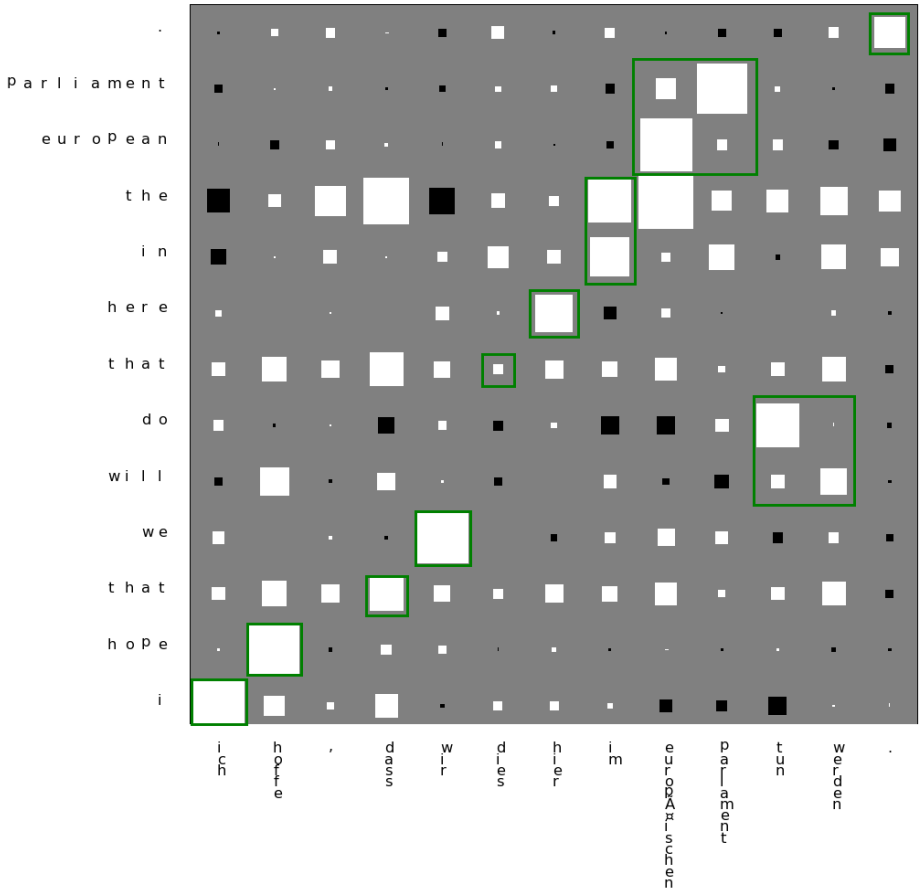

Figure 4.3: L_1 Regularized Regression for word alignment figure, QP, 1000 training instances.

L_2 regularized solution and we observe better word alignments. All of the alignment problems we mentioned are now solved with FSR. Using these alignments, we can generate a phrase table as depicted in Figure 4.5.

When we compare the figures, we see a clear advantage of using sparse regression techniques for learning the coefficients matrix. Sparser coefficients matrix corresponds to fewer translation alternatives for each phrase, which can reduce the computations during test time. However, selecting fewer alternatives can correspond to a higher chance of missing the correct translation as well.

When we compare the coefficients matrices obtained with QP and FSR, we see that FSR obtains a sparser solution whereas QP's coef-

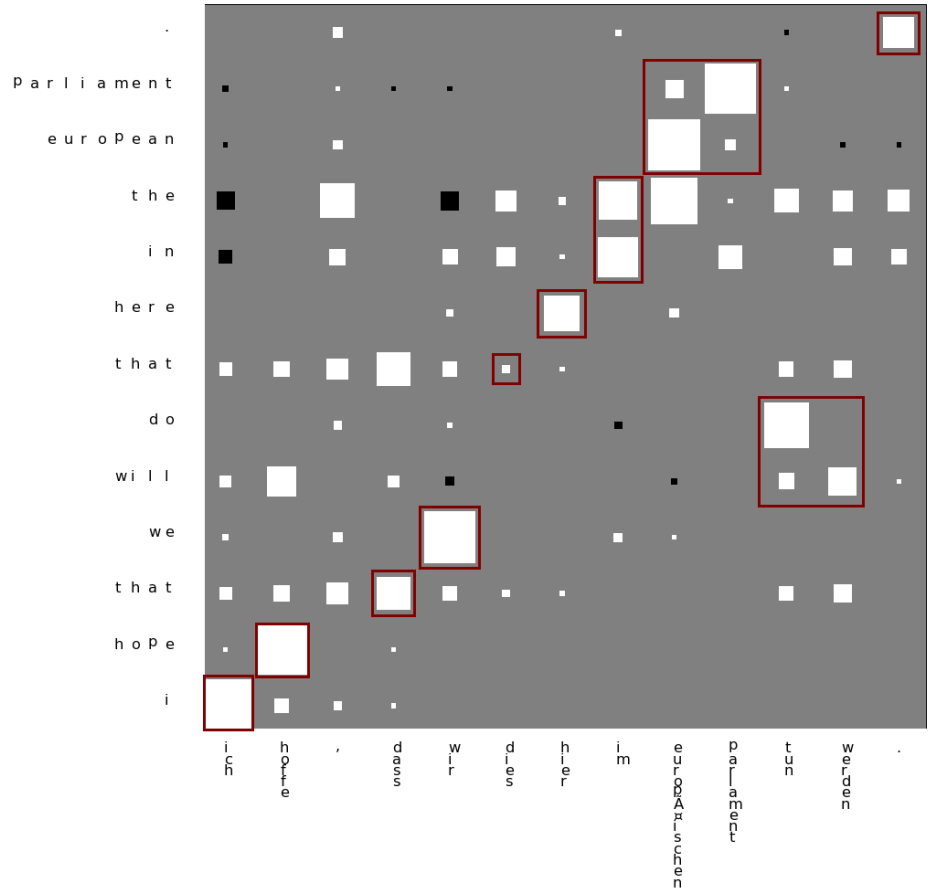

Figure 4.4: L_1 Regularized Regression for word alignment figure, FSR, 1000 training instances.

ficients matrix contain largely nonzero entries, most of which appear close to zero. QP solution can be made sparser by thresholding the values afterwards; but we still observe problems with the mappings obtained such as the source feature `dass` being mapped to the feature `the` on the English side.

We have seen that on this example word alignment scenario, sparse regression models can perform better than L_2 regularized regression and FSR performs better than QP in terms of obtaining a sparser coefficients matrix and better alignments. We discuss word alignment performance of RegMT on a sample test set in section 8.7 where we compare the performance of FSR to a larger number of learning algo-

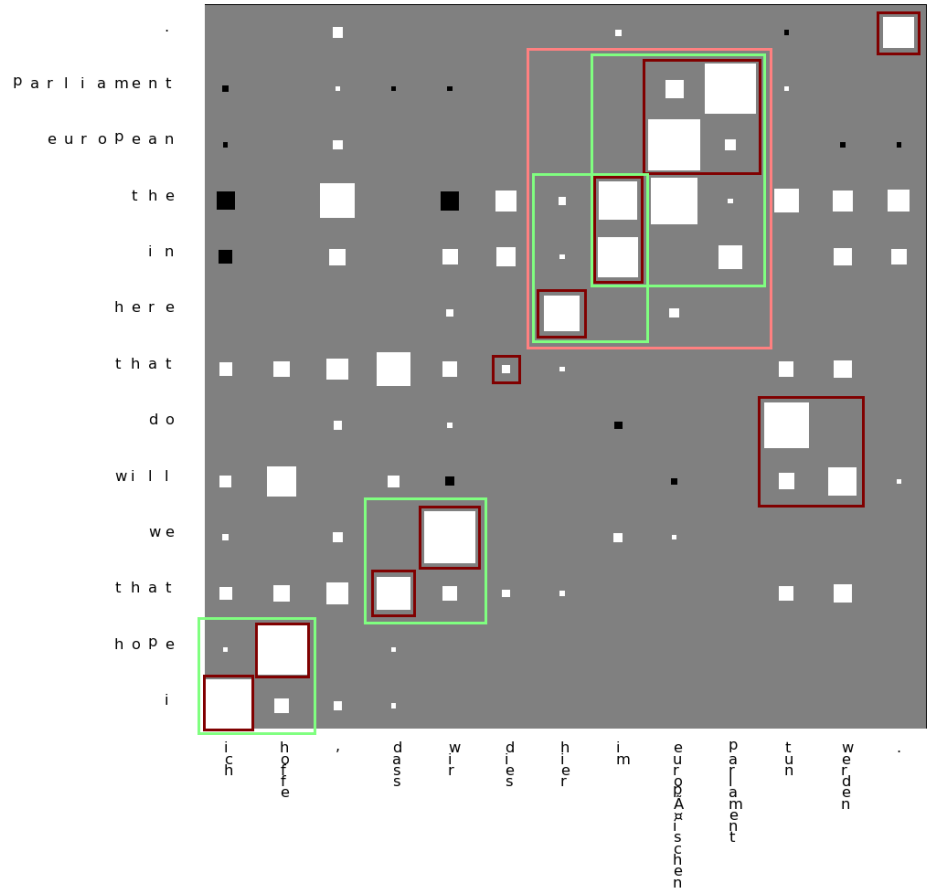

Figure 4.5: L_1 Regularized Regression for word alignment figure, FSR, with phrases identified, using 1000 training instances.

rithms and observe supporting results.

4.4.1 Evaluating Phrase Alignment Performance

In (Biçici and Yuret, 2010a), we show the effectiveness of using L_1 regularization versus L_2 used in ridge regression on the word and phrase alignment task of machine translation. We observe that after using about 200 instances for training, F_1 measure results as well as the squared error used start improving slowly, which suggests that we may obtain similar performance using a few hundred training instances for each test sentence. In chapter 5, we show how we can improve the

training instance selection methods we use by optimizing the coverage of source and target features of the test sentences.

4.5 RegMT W Cost Functions

We present the L_2 regularized regression problem and the *lasso* (least absolute shrinkage and selection operator) or the L_1 regularized regression problem again in this section.

L_2 regularized regression solution:

$$\mathbf{W}_{L_2} = \underset{\mathbf{W}\in\mathbb{R}^{N_Y\times N_X}}{\arg\min} \| \mathbf{M}_Y - \mathbf{W}\mathbf{M}_X \|_F^2 + \lambda \| \mathbf{W} \|_F^2 . \quad (4.16)$$

The *lasso* (least absolute shrinkage and selection operator) solution:

$$\mathbf{W}_{L_1} = \underset{\mathbf{W}\in\mathbb{R}^{N_Y\times N_X}}{\arg\min} \| \mathbf{M}_Y - \mathbf{W}\mathbf{M}_X \|_F^2 + \lambda \| \mathbf{W} \|_1 . \quad (4.17)$$

4.6 Summary

We introduced sparse regression techniques for statistical machine translation. We show that obtaining a few nonzero target feature coefficients corresponding to a source feature in the coefficients matrix can be preferable for machine translation.

We presented various techniques of regression returning sparse coefficients matrices. The effectiveness of *lasso* to learn the mappings between sparsely observed feature sets versus L_2 regularized regression is demonstrated in an example word alignment scenario.

The next section discusses our training instance selection techniques using feature decay algorithms. chapter 6 shows the results we obtain when we use the RegMT system scores to rerank alternative translations and pick the best one.

Chapter 5

Instance Selection for Machine Translation using Feature Decay Algorithms

We present an empirical study of instance selection techniques for machine translation. In an active learning setting, instance selection minimizes the human effort by identifying the most informative sentences for translation. In a transductive learning setting, selection of training instances relevant to the test set improves the final translation quality. After reviewing the state of the art in the field, we generalize the main ideas in a class of instance selection algorithms that use feature decay. Feature decay algorithms increase the diversity of the training set by devaluing features that are already included. We show that the feature decay rate has a very strong effect on the final translation quality whereas the initial feature values, inclusion of higher order features, or sentence length normalizations do not. We evaluate the best instance selection methods trained with a Moses baseline SMT system using the whole 1.6 million sentence English-German section of the Europarl corpus. We demonstrate that feature decay algorithms perform better than previous work both by selecting more relevant instances and by obtaining a higher BLEU performance. We show that selecting the best 3000 training sentences for a specific test sentence is sufficient to obtain a score within 1 BLEU of the baseline, using 5% of the training

data is sufficient to exceed the baseline, and a ~ 2 BLEU improvement over the baseline score is possible by using optimally selected subset of the training data. In out-of-domain translation, we are able to reduce the training set size to about 7% and achieve similar performance as the baseline. Part of this work is published as (Biçici and Yuret, 2011a).

5.1 Introduction

Statistical machine translation (SMT) makes use of a large number of parallel sentences, sentences whose translations are known in the target language, to derive translation tables, estimate parameters, and generate the actual translation. Not all of the parallel training sentences nor the translation table that is generated is used during decoding a given set of test sentences and filtering is usually performed for computational advantage (Koehn et al., 2007). Some recent regression-based statistical machine translation systems rely on a small sized training data to learn the mappings between source and target features (Biçici and Yuret, 2010b; Serrano et al., 2009; Wang and Shawe-Taylor, 2008). Regression has some computational disadvantages when scaling to large number of training instances.

Previous work shows that the more the training data, the better the translations become (Koehn, 2006). However, with the increased size of the parallel training sentences there is also the added noise, making relevant instance selection important. Phrase-based SMT systems rely heavily on accurately learning word alignments from the given parallel training sentences. Proper instance selection plays an important role in obtaining a small sized training set with which correct alignments can be learned. Word-level translation accuracy is also affected by the number of times a word occurs in the parallel training sentences (Koehn and Knight, 2001). Koehn and Knight find that about 50 examples per word are required to achieve a performance close to using a bilingual lexicon in their experiments. Translation performance can improve

as we include multiple possible translations for a given word, which increases the diversity of the training set.

Transduction uses test instances, which can sometimes be accessible at training time, to learn specific models tailored towards the test set which also reduces computation by not using the full training set. Transductive retrieval selects training data close to the test set given a parallel training sentences and a test set. This work shows that *transductive retrieval* of the training set for statistical machine translation allows us to achieve a performance better than using all of the parallel training sentences. When selecting training data, we seek to maximize the coverage or the percentage of test source and target features (i.e. n-grams) found in the training set using minimal number of target training features and a fixed number of training instances. Diversifying the set of training sentences can help us increase the coverage. We show that target coverage bounds the achievable BLEU score with a given training set and small increases can result in large increases on this BLEU bound.

We develop the feature decay algorithms (FDA) that aim to maximize the coverage of the target language features and achieve significant gains in translation performance. We find that decaying feature weights has significant effect on the performance. We improve the BLEU score by $\sim$2 using about 20% of the available training data in terms of target words and by $\sim$1 with only about 5%. We show that selecting 3000 instances for a test sentence is sufficient to obtain a score within 1 BLEU of the baseline. In the out-of-domain translation task, we are able to reduce the training set size to its 7% to achieve similar performance as the baseline.

The next section reviews related previous work. We discuss the FDA in section 5.3. section 5.4 presents our coverage and translation results both in and out-of-domain and includes an instance selection method also designed for improving word alignment results. We list our contributions in the last section.

5.2 Related Work

Transductive learning makes use of test instances, which can sometimes be accessible at training time, to learn specific models tailored towards the test set. Selection of training instances relevant to the test set improves the final translation quality as in transductive learning and decreases human effort by identifying the most informative sentences for translation as in active learning. Instance selection in a transductive learning framework selects the best instances for a given test set (Lü et al., 2007). *Active learning* selects training samples that will benefit the learning algorithm the most over the unlabeled dataset $\mathcal{U}$ from a labeled training set $\mathcal{L}$ or from $\mathcal{U}$ itself after labeling (Banko and Brill, 2001). Active learning in SMT selects which instances to add to the training set to improve the performance of a baseline system (Ananthakrishnan et al., 2010; Haffari et al., 2009). Recent work involves selecting sentence or phrase translation tasks for external human effort (Bloodgood and Callison-Burch, 2010). Below we present examples of both with a label indicating whether they follow an approach close to active learning [AL] or transductive learning [TL] and in our experiments we use the transductive framework.

TF-IDF [TL]: Lü et al. (2007) use tf-idf information retrieval technique based cosine score to select a subset of the parallel training sentences close to the test set for SMT training. They outperform the baseline system when the top 500 training instances per test sentence are selected. The terms used in their tf-idf measure correspond to words where this work focuses on bigram feature coverage. When the combination of the top N selected sentences are used as the training set, they show increase in the performance at the beginning and decrease when 2000 sentences are selected for each test sentence.

N-gram coverage [AL]: Eck et al. (2005) use n-gram feature coverage to select training instances:

$$\phi_{NGRAM}(S) = \frac{\sum_{i=1}^{n} \sum_{\text{unseen } x \in X_i(S)} C(x)}{|S|}, \tag{5.1}$$

for sentence S with $X_i(S)$ storing the i-grams found in S and $C(x)$ returning the count of x in the parallel training sentences. ϕ_{NGRAM} score sums over unseen n-grams to increase the coverage of the training set. The denominator involving the length of the sentence takes the translation cost of the sentence into account. Eck et al. (2005) also note that longer sentences are more difficult for training SMT models. In their experiments, they are not able to reach a performance above the baseline system's BLEU score, which is using all of the parallel training sentences, but they achieve close performance by using about 15% of the parallel training sentences.

DWDS [AL]: Density weighted diversity sampling ($DWDS$) (Ambati et al., 2010) score tries to select sentences containing the n-gram features in the unlabeled dataset $\mathcal{U}$ while increasing the diversity among the sentences selected, $\mathcal{L}$ (labeled). $DWDS$ increases the score of a sentence with increasing frequency of its n-grams found in $\mathcal{U}$ and decreases with increasing frequency in the already selected set of sentences, $\mathcal{L}$, in favor of diversity. Let $P_{\mathcal{U}}(x)$ denote the probability of feature x in $\mathcal{U}$ and $C_{\mathcal{L}}(x)$ denote its count in $\mathcal{L}$. Then:

$$d(S) = \frac{\sum_{x \in X(S)} P_{\mathcal{U}}(x) e^{-\alpha C_{\mathcal{L}}(x)}}{|X(S)|} \tag{5.2}$$

$$u(S) = \frac{\sum_{x \in X(S)} I(x \notin X(\mathcal{L}))}{|X(S)|} \tag{5.3}$$

$$\phi_{DWDS}(S) = \frac{2d(S)u(S)}{d(S) + u(S)}, \tag{5.4}$$

where $X(S)$ stores the features of S and α is a decay parameter. $d(S)$ denotes the density of S proportional to the probability of its features in $\mathcal{U}$ and inversely proportional to their counts in $\mathcal{L}$ and $u(S)$ its uncertainty, measuring the percentage of new features in S. These two

scores are combined using harmonic mean. *DWDS* tries to select sentences containing similar features in $\mathcal{U}$ with high diversity. In their active learning experiments, they selected 1000 training instances in each iteration and retrained the SMT system.

Log-probability ratios [AL]: Haffari et al. (2009) develop sentence selection scores using feature counts in $\mathcal{L}$ and $\mathcal{U}$, increasing for frequent features in $\mathcal{U}$ and decreasing for frequent features in $\mathcal{L}$. They use geometric and arithmetic averages of log-probability ratios in an active learning setting where 200 sentences from $\mathcal{U}$ are selected and added to $\mathcal{L}$ with their translations for 25 iterations (Haffari et al., 2009). Later, Haffari and Sarkar (2009) distinguish between features found in the phrase table, x_{reg}, and features not found, x_{oov}. OOV features are segmented into subfeatures (i.e. feature "go to school" is segmented as: (go to school), (go)(to school), (go to)(school), (go)(to)(school)). *Expected log probability ratio (ELPR)* score is used:

$$\begin{aligned}\phi_{ELPR}(S) = & \frac{0.4}{|X_{reg}(S)|} \sum_{x \in X_{reg}(S)} \log \frac{P_{\mathcal{U}}(x)}{P_{\mathcal{L}}(x)} \\ & + \frac{0.6}{|X_{oov}(S)|} \sum_{x \in X_{oov}(S)} \sum_{h \in H(x)} \frac{1}{|H(x)|} \sum_{y \in Y_h(x)} \log \frac{P_{\mathcal{U}}(y)}{P_{\mathcal{L}}(y)},\end{aligned} \tag{5.5}$$

where $H(x)$ return the segmentations of x and $Y_h(x)$ return the features found in segment h. ϕ_{ELPR} performs better than geometric average in their experiments (Haffari and Sarkar, 2009).

Perplexity [AL & TL]: Perplexity of the training instance as well as inter-SMT-system disagreement are also used to select training data for translation models (Mandal et al., 2008). The increased difficulty in translating a parallel sentence or its novelty as found by the perplexity adds to its importance for improving the SMT model's performance. A sentence having high perplexity (a rare sentence) in $\mathcal{L}$ and low perplexity (a common sentence) in $\mathcal{U}$ is considered as a candidate for addition. They are able to improve the performance of a baseline system trained on some initial parallel training sentences together with additional par-

allel sentences data using the initial parallel training sentences and part of the additional data.

Alignment [TL]: Uszkoreit et al. (2010) mine parallel text to improve the performance of a baseline translation model on some initial document translation tasks. They retrieve similar documents using inverse document frequency weighted cosine similarity. Then, they filter nonparallel sentences using their word alignment performance, which is estimated using the following score:

$$\text{score}(A) = \sum_{(s,t)\in A} \ln \frac{p(s,t)}{p(s)p(t)}, \tag{5.6}$$

where A stands for an alignment between source and target words and the probabilities are estimated using a word aligned parallel training sentences. The produced parallel data is used to expand a baseline parallel training sentences and shown to improve the translation performance of machine translation systems.

5.3 Instance Selection with Feature Decay

In this section we will describe a class of instance selection algorithms for machine translation that use feature decay, i.e. increase the diversity of the training set by devaluing features that have already been included. Our abstraction makes three components of such algorithms explicit permitting experimentation with their alternatives:

- The value of a candidate training sentence as a function of its features.
- The initial value of a feature.
- The update of the feature value as instances are added to the training set.

A feature decay algorithm (FDA) aims to maximize the coverage of the target language features (such as words, bigrams, and phrases) for

the test set. A target language feature that does not appear in the selected training instances will be difficult to produce regardless of the decoding algorithm (impossible for unigram features). In general we do not know the target language features, only the source language side of the test set is available. Unfortunately, selecting a training instance with a particular source language feature does not guarantee the coverage of the desired target language feature. There may be multiple translations of a feature appropriate for different senses or different contexts. For each source language feature in the test set, FDA tries to find as many training instances as possible to increase the chances of covering the appropriate target language feature. It does this by reducing the value of the features that are already included after picking each training instance. Algorithm 2 gives the pseudo-code for FDA.

The input to the algorithm is parallel training sentences, the number of desired training instances, and the source language features of the test set. We use unigram and bigram features; adding trigram features does not seem to significantly affect the results. The user has the option of running the algorithm for each test sentence separately, then possibly combining the resulting training sets. We will present results with these variations in section 5.4.

The first foreach loop initializes the value of each test set feature. We experimented with initial feature values that are constant, proportional to the length of the n-gram, or log-inverse of the corpus frequency. We have observed that the initial value does not have a significant effect on the quality of the training instances selected. The feature decay rule dominates the behavior of the algorithm after the first few iterations. However, we prefer the log-inverse values because they lead to fewer score ties among candidate instances and result in faster running times.

The second foreach loop initializes the score for each candidate training sentence and pushes them onto a priority queue. The score is calculated as the sum of the feature values. Note that as we change the feature values, the sentence scores in the priority queue will no longer

Algorithm 2: The Feature Decay Algorithm

Input: Parallel training sentences $\mathcal{U}$, test set features $\mathcal{F}$, and desired number of training instances N.

Data: A priority queue $\mathcal{Q}$, sentence scores `score`, feature values `fvalue`.

Output: Subset of the parallel sentences to be used as the training data $\mathcal{L} \subseteq \mathcal{U}$.

```
1  foreach f ∈ F do
2      fvalue(f) ← init(f,U)
3  foreach S ∈ U do
4      score(S) ← Σ_{f∈features(S)} fvalue(f)
5      push(Q, S, score(S))
6  while |L| < N do
7      S ← pop(Q)
8      score(S) ← Σ_{f∈features(S)} fvalue(f)
9      if score(S) ≥ topval(Q) then
10         L ← L ∪ {S}
11         foreach f ∈ features(S) do
12             fvalue(f) ← decay(f,U,L)
13     else
14         push(Q, S, score(S))
```

be correct. However they will still be valid upper bounds because the feature values only get smaller. Features that do not appear in the test set are considered to have zero value. This observation can be used to speed up the initialization by using a feature index and only iterating over the sentences that have features in common with the test set.

Finally the while loop populates the training set by picking candidate sentences with the highest scores. This is done by popping the top scoring candidate sentence S from the priority queue at each iteration. We recalculate its score because the values of its features may have changed. We compare the recalculated score of S with the score of the

next best candidate. If the score of S is equal or better we are sure that it is the top candidate because the scores in the priority queue are upper bounds. In this case we place S in our training set and decay the values of its features. Otherwise we push S back into the priority queue with its updated score.

The feature decay function (Equation 5.8) is the heart of the algorithm. Unlike the choice of features (bigram vs trigram) or their initial values (constant vs log–inverse–frequency) the rate of decay has a significant effect on the performance. We found it is optimal to reduce feature values at a rate of $1/n$ where n is the current training set count of the feature. The results get significantly worse with no feature decay. They also get worse with faster, exponential feature decay, e.g. $1/2^n$. Table 5.1 presents the experimental results that support these conclusions where the obtained coverage values for the source and target language bigrams, *scov* and *tcov* respectively, are given. We use the following settings for the experiments in section 5.4:

$$\texttt{init}(f,\mathcal{U}) = 1 \text{ or } \log(|\mathcal{U}|/\texttt{cnt}(f,\mathcal{U})) \tag{5.7}$$

$$\texttt{decay}(f,\mathcal{U},\mathcal{L}) = \frac{\texttt{init}(f,\mathcal{U})}{1+\texttt{cnt}(f,\mathcal{L})} \text{ or } \frac{\texttt{init}(f,\mathcal{U})}{1+2^{\texttt{cnt}(f,\mathcal{L})}} \tag{5.8}$$

		en→de		de→en	
init	decay	*scov*	*tcov*	*scov*	*tcov*
1	none	.761	.484	.698	.556
$\log(1/f)$	none	.855	.516	.801	.604
1	$1/n$	**.967**	**.575**	**.928**	**.664**
$\log(1/f)$	$1/n$	.967	.570	.928	.656
1	$1/2^n$	.967	.553	.928	.653
$\log(1/f)$	$1/2^n$	.967	.557	.928	.651

Table 5.1: FDA parameter selection experiments. The first two columns give the initial value and decay formula used for features. f is the corpus frequency of a feature and n is its count in selected instances. The next four columns give the expected coverage of the source and target language bigrams, *scov* and *tcov* respectively, of a test sentence when 100 training sentences are selected.

5.4 Experiments

We perform translation experiments on the English-German (*en-de*) language pair using the parallel training sentences provided by Callison-Burch et al. (2010) (WMT'10). The English-German section of the Europarl corpus contains about 1.6 million sentences. We perform *in-domain* experiments to discriminate among different instance selection techniques better in a setting with low out-of-vocabulary rate. We randomly select the test set *test* with 2,588 target words (100 sentences) and separate development set *dev* with 26,178 target words (1000 sentences). We use the language model corpus provided in WMT'10 to build a 5-gram model.

We use target language *bigram* coverage, *tcov*, as a quality measure for a given training set, which measures the percentage of the target bigram features of the test sentence found in a given training set. We compare the achieved *tcov* and the translation performance of FDA with related work. We also perform small scale SMT experiments where only a couple of thousand training instances are used for each test sentence.

5.4.1 The Effect of Coverage on Translation

BLEU (Papineni et al., 2001) is a precision based measure and uses n-gram match counts up to order n to determine the quality of a given translation. The absence of a given word or translating it as another word interrupts the continuity of the translation and decreases the BLEU score even if the order among the words is determined correctly. Therefore, the target coverage of an out-of-domain test set whose translation features are not found in the training set bounds the translation performance of an SMT system.

We estimate this translation performance bound from target coverage by assuming that the missing tokens can appear randomly at any location of a given sentence where sentence lengths are normally distributed with mean 25.6 and standard deviation 14.1. This is close to

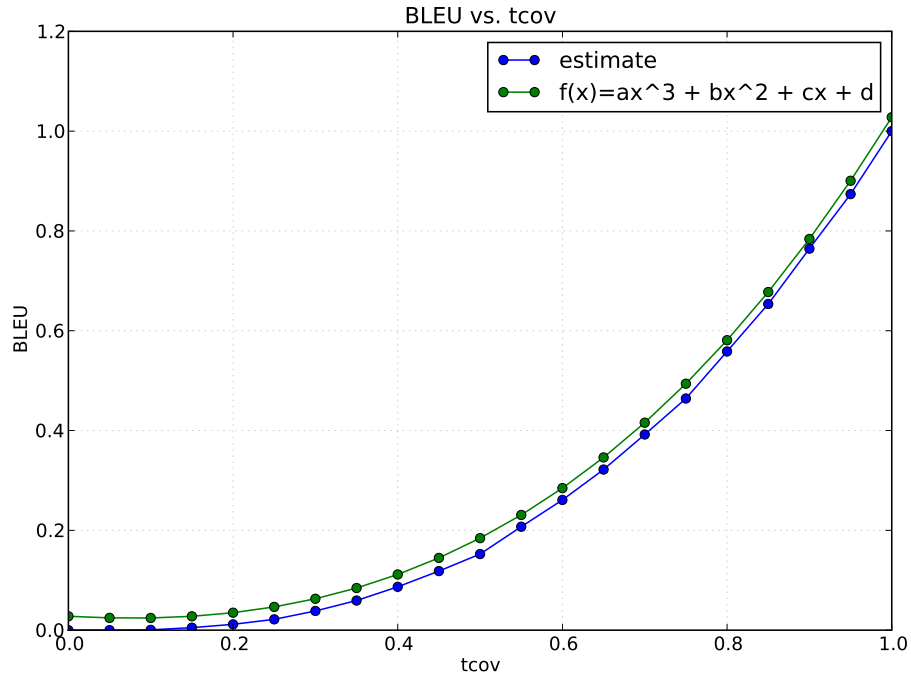

Figure 5.1: Effect of coverage on translation performance. BLEU bound is a third-order function of target coverage. High coverage → High BLEU.

the sentence length statistics of the German side Europarl corpus used in (Callison-Burch et al., 2010) (WMT'10). We replace all unknown words found with an `UNK` token and calculate the BLEU score. We perform this experiment for $10,000$ instances and repeat for 10 times. The sentences tested has the form: $S_i \approxeq a\ b$ `UNK` $d\ e$, for tokens a, b, d, e.

The obtained BLEU scores for target coverage values is plotted in Figure 5.1 with the label *estimate*. We also fit a third order polynomial function of target coverage 0.025 BLEU scores above the *estimate* values to show the similarity with the BLEU scores bound estimated, whose parameters are found to be $[0.56, 0.53, -0.09, 0.003]$ with a least-squares fit. Thus, in this setting, given a translation for a test set, $\mathbf{T}$, and its *tcov* value, we can estimate the BLEU performance without performing the translation and calculating BLEU with

reference set $\mathbf{R}$:

$$\begin{aligned} \text{BLEU}(\mathbf{T}, \mathbf{R}) &\approx \text{BLEU}(\mathbf{T}, tcov) \\ &= 0.56 * tcov^3 + 0.53 * tcov^2 - 0.09 * tcov + 0.003. \end{aligned} \tag{5.9}$$

Figure 5.1 shows that the BLEU score bound obtained has a third-order polynomial relationship with target coverage and small increases in the target coverage can result in large increases on this BLEU bound.

5.4.2 Coverage Results

We select N training instances per test sentence using FDA (Algorithm 2), *TF-IDF* with bigram features, *NGRAM* scoring (Equation 5.1), *DWDS* (Equation 5.4), and *ELPR* (Equation 5.5) techniques from previous work. For the active learning algorithms, source side test corpus becomes $\mathcal{U}$ and the selected training set $\mathcal{L}$. For all the techniques, we compute 1-grams and 2-grams as the features used in calculating the scores and add only one sentence to the training set at each iteration except for *TF-IDF*. We set α parameter of *DWDS* to 1 as given in their paper. We adaptively select the top scoring instance at each step from the set of possible sentences $\mathcal{U}$ with a given scorer $\phi(.)$ and add the instance to the training set, $\mathcal{L}$, until the size of $\mathcal{L}$ reaches N for the related work other than *TF-IDF*. We test all of the algorithms in this transductive learning setting.

We measure the *bigram* coverage when all of the training sentences selected for each test sentence are combined. The results are presented in Figure 5.2 where the x-axis is the number of words of the training set and y-axis is the target coverage obtained. FDA has a steep slope in its increase and it is able to reach target coverage of $\sim$0.84. *DWDS* performs worse initially but its target coverage improve after a number of instances are selected due to its exponential feature decay procedure. *TF-IDF* performs worse than *DWDS* and it provides a fast alternative to FDA instance selection but with some decrease in coverage. *ELPR* and *NGRAM* instance selection techniques per-

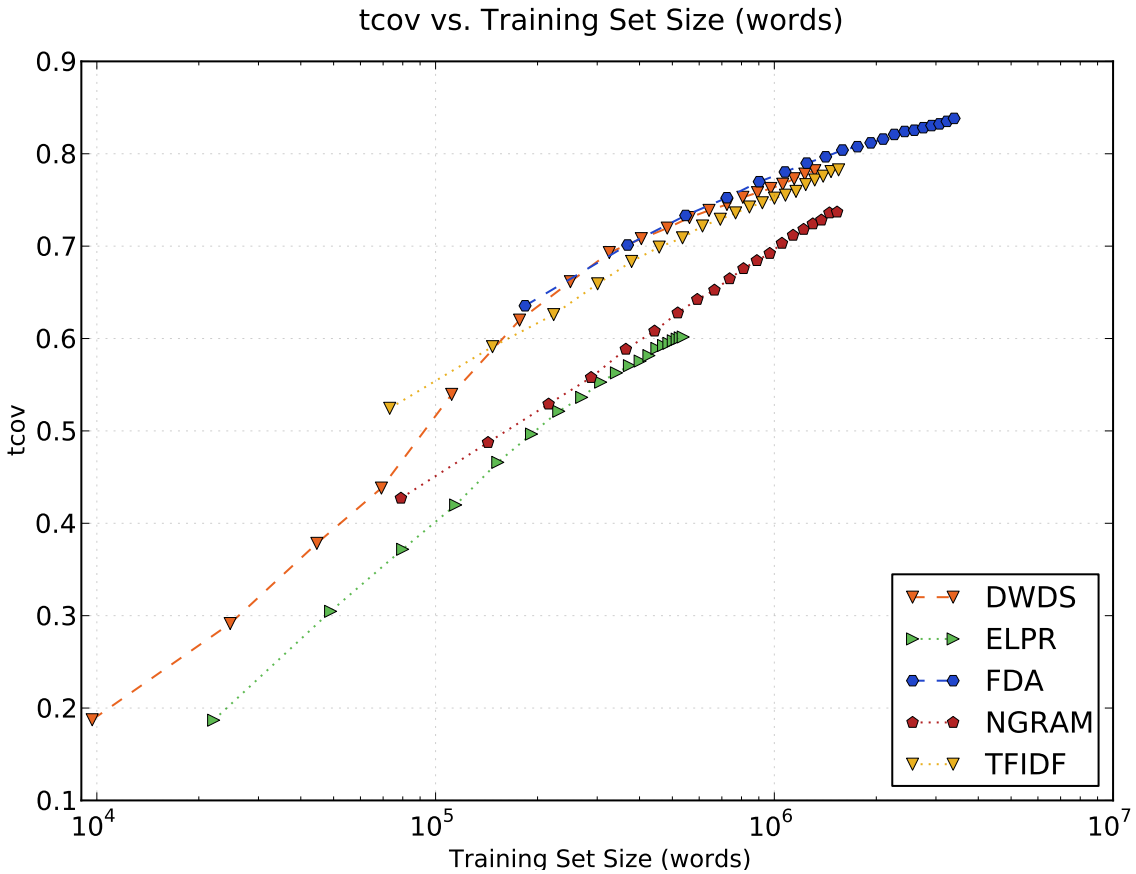

Figure 5.2: Target coverage curve comparison with previous work. Figure shows the rate of increase in *tcov* as the size of $\mathcal{L}$ increase.

form worse. *NGRAM* achieves better coverage than *ELPR*, although it lacks a decay procedure.

When we compare the sentences selected, we observe that FDA prefers longer sentences due to summing feature weights and it achieves larger target coverage value. *NGRAM* is not able to discriminate between sentences well and a large number of sentences of the same length get the same score when the unseen n-grams belong to the same frequency class. The statistics of $\mathcal{L}$ obtained with the instance selection techniques differ from each other as given in Table 5.2, where $N = 1000$ training instances selected per test sentence. We observe that *DWDS* has fewer unique target bigram features than *TF-IDF* although it selects longer target sentences. *NGRAM* obtains a large

number of unique target bigrams although its selected target sentences have similar lengths with *DWDS* and *ELPR* prefers short sentences.

Technique	Unique bigrams	Words per sent	*tcov*
FDA	827,928	35.8	.74
DWDS	412,719	16.7	.67
TF-IDF	475,247	16.2	.65
NGRAM	626,136	16.6	.55
ELPR	172,703	10.9	.35

Table 5.2: Statistics of the obtained target $\mathcal{L}$ for $N = 1000$.

5.4.3 Translation Results

We develop separate phrase-based SMT models using Moses (Koehn et al., 2007) using default settings with maximum sentence length set to 80 and obtained baseline system score as 0.3577 BLEU. We use the training instances selected by FDA in three learning settings:

$\mathcal{L}_{\cup}$: $\mathcal{L}$ is the union of the instances selected for each test sentence.

$\mathcal{L}_{\cup_{\mathcal{F}}}$: $\mathcal{L}$ is selected using all of the features found in the test set.

$\mathcal{L}_{\mathcal{I}}$: $\mathcal{L}$ is the set of instances selected for each test sentence.

We develop separate Moses systems with each training set. $\mathcal{L}_{\cup}$ results are plotted in Figure 5.3 where we increasingly select $N \in \{100, 200, 500, 1000, 2000, 3000, 5000, 10000\}$ instances for each test sentence for training. The improvements over the baseline are statistically significant with paired bootstrap resampling using 1000 samples (Koehn, 2004). We give the details of statistical significance testing in general and for SMT in Appendix B. As we select more instances, the performance of the SMT system increases as expected and we start to see a decrease in the performance after selecting $\sim 10^7$ target words. We observe that a $\sim$2 BLEU improvement over the baseline is possible by optimally selected subset of the training data. We obtain comparable results for *de-en* direction. The performance increase is likely to be due to the reduction in the number of noisy or irrele-

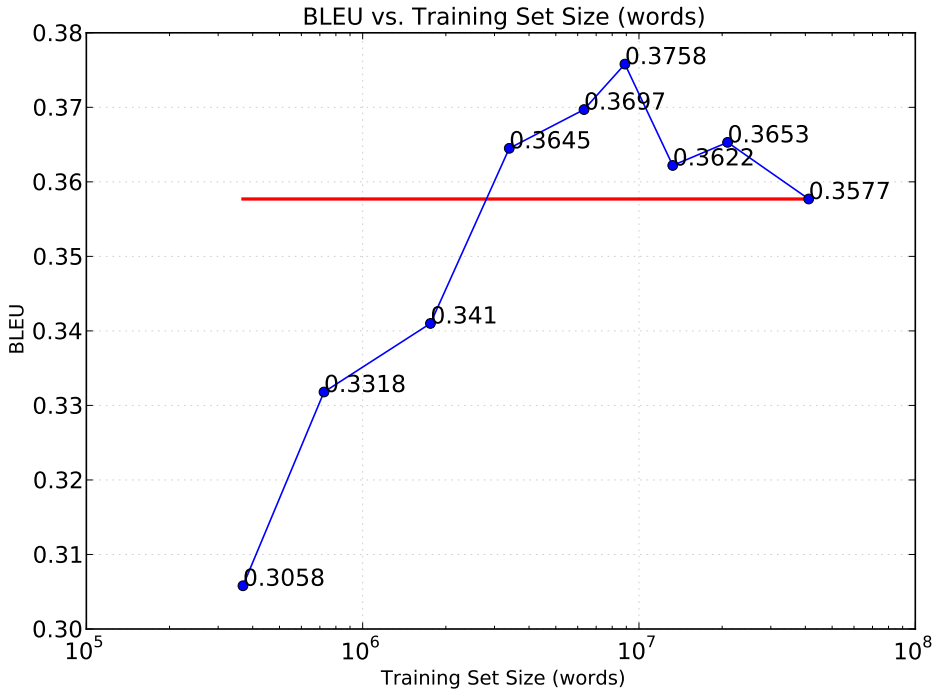

Figure 5.3: BLEU vs. the number of target words in $\mathcal{L}_{\cup}$.

vant training instances and the increased precision in the probability estimates in the generated phrase tables.

$\mathcal{L}_{\cup_{\mathcal{F}}}$ results given in Table 5.3 show that we can achieve within 1 BLEU performance using about 3% of the parallel training sentences target words (30,000 instances) and better performance using only about 5% (50,000 instances).

# sent	# target words	BLEU	NIST
10,000	449,116	0.3197	5.7788
20,000	869,908	0.3417	6.0053
30,000	1,285,096	0.3492	6.0246
50,000	**2,089,403**	**0.3711**	**6.1561**
100,000	**4,016,124**	**0.3648**	**6.1331**
ALL	41,135,754	0.3577	6.0653

Table 5.3: Performance for *en-de* using $\mathcal{L}_{\cup_{\mathcal{F}}}$. ALL corresponds to the baseline system using all of the parallel training sentences. **bold** correspond to statistically significant improvement over the baseline result.

The results with $\mathcal{L}_{\mathcal{I}}$ when building an individual Moses model for

each test sentence are given in Table 5.4. Individual SMT training and translation can be preferable due to smaller computational costs and high parallelizability. As we translate a single sentence with each SMT system, tuning weights becomes important. We experiment three settings: (1) using 100 sentences for tuning, which are randomly selected from *dev*, (2) using the mean of the weights obtained in (1), and (3) using the weights obtained in the union learning setting ($\mathcal{L}_{\cup}$). We observe that we can obtain a performance within 2 BLEU difference to the baseline system by training on 3000 instances per sentence (underlined) using the mean weights and 1 BLEU difference using the union weights. We also experimented with increasing the N-best list size used during MERT optimization (Hasan et al., 2007), with increased computational cost, and observed some increase in the performance.

N	100 *dev* sents	Mean	$\mathcal{L}_{\cup}$
1000	0.3149	0.3242	0.3354
2000	0.3258	0.3352	0.3395
3000	0.3270	0.3374	0.3501
5000	0.3217	0.3303	0.3458

Table 5.4: $\mathcal{L}_{\mathcal{I}}$ performance for *en-de* using 100 sentences for tuning or mean of the weights from $\mathcal{L}_{\mathcal{I}}$ or *dev* weights obtained for $\mathcal{L}_{\cup}$.

Comparison with related work: Table 5.5 presents the translation results compared with previous work using $N = 1000$. We observe that coverage and translation performance are correlated. Although the coverage increase of *DWDS* and FDA appear similar, due to the third-order polynomial growth of BLEU with respect to coverage, we achieve large BLEU gains in translation. We observe increased BLEU gains when compared with the results of *TF-IDF*, *NGRAM*, and *ELPR* in order.

FDA	*DWDS*	*TF-IDF*	*NGRAM*	*ELPR*
0.3645	0.3547	0.3405	0.2572	0.2268

Table 5.5: BLEU results using different techniques with $N = 1000$. High coverage $\rightarrow$ High BLEU.

We note that *DWDS* originally selects instances using the whole test

corpus to estimate $P_{\mathcal{U}}(x)$ and selects 1000 instances at each iteration. We experimented with both of these settings and obtained 0.3058 and 0.3029 BLEU respectively. Lower performance suggest the importance of updating weights after each instance selection step.

5.4.4 Instance Selection for Alignment

We have shown that high coverage is an integral part of training sets for achieving high BLEU performance. SMT systems also heavily rely on the word alignment of the parallel training sentences to derive a phrase table that can be used for translation. GIZA++ (Och and Ney, 2003) is commonly used for word alignment and phrase table generation, which is prone to making more errors as the length of the training sentence increase (Ravi and Knight, 2010). Therefore, we analyze instance selection techniques that optimize coverage and word alignment performance and at the same time do not produce very long sentences. Too few words per sentence may miss the phrasal structure, whereas too many words per sentence may miss the actual word alignment for the features we are interested. We are also trying to retrieve relevant training sentences for a given test sentence to increase the feature alignment performance.

Shortest: A baseline strategy that can minimize the training feature set's size involves selecting the shortest translations containing each feature.

Co-occurrence: We use *co-occurrence* of words in the parallel training sentences to retrieve sentences containing co-occurring items. Dice's coefficient (Dice, 1945) is used as a heuristic word alignment technique giving an association score for each pair of word positions (Och and Ney, 2003). We define Dice's coefficient score as:

$$dice(x, y) = \frac{2C(x, y)}{C(x)C(y)}, \tag{5.10}$$

where $C(x, y)$ is the number of times x and y co-occur and $C(x)$ is the number of times x appears in the selected training set. Given a

	$\mathcal{L}_{\mathcal{U}}$			BLEU		
	Unique bigrams	Words per sent	*tcov*	$\mathcal{L}_{\mathcal{U}}$	$\mathcal{L}_{\mathcal{I}}$ 100 dev	$\mathcal{L}_{\mathcal{I}}$ Mean
FDA	827,928	35.8	.74	.3645	.3149	.3242
dice	707,802	26.2	.73	.3635	.3144	.3171

Table 5.6: We compare the translation performance of *dice* with FDA as well as the statistics of the $\mathcal{L}_{\mathcal{U}}$ obtained.

test source sentence, $S_{\mathcal{U}}$, we can estimate the goodness of a training sentence pair, (S, T), by the sum of the alignment scores:

$$\phi_{dice}(S_{\mathcal{U}}, S, T) = \frac{1}{|T| \log |S|} \sum_{x \in X(S_{\mathcal{U}})} \sum_{j=1}^{|T|} \sum_{y \in Y(x)} dice(y, T_j), \qquad (5.11)$$

where $X(S_{\mathcal{U}})$ stores the features of $S_{\mathcal{U}}$ and $Y(x)$ lists the tokens in feature x. The difficulty of word aligning a pair of training sentences, (S, T), can be approximated by $|S|^{|T|}$. We use a normalization factor proportional to $|T| \log |S|$.

The average target words per sentence using ϕ_{dice} drops to 26.2 compared to 35.8 of FDA. We still obtain a better performance than the baseline *en-de* system with the union of 1000 training instances per sentence with 0.3635 BLEU and 6.1676 NIST scores. Coverage comparison with FDA shows slight improvement with lower number of target bigrams and similar trend for others (Figure 5.4). We note that shortest strategy achieves better performance than both *ELPR* and *NGRAM*. We obtain 0.3144 BLEU and 5.5 NIST scores in the individual translation task with 1000 training instances per sentence and 0.3171 BLEU and 5.4662 NIST scores when the mean of the weights is used. Comparison of *dice* with FDA is given in Table 5.6.

5.4.5 Out-of-domain Translation Results

We have used FDA and *dice* algorithms to select training sets for the out-of-domain challenge test sets used in (Callison-Burch et al., 2011) (WMT'11). The parallel training sentences contain about 1.9 million training sentences and the test set contain 3003 sentences. We built

		en-de	*de-en*	*en-es*	*es-en*
BLEU	ALL	0.1376	0.2074	0.2829	0.2919
	FDA	0.1363	0.2055	0.2824	0.2892
	dice	0.1374	0.2061	0.2834	0.2857
# target words $\times 10^6$	ALL	47.4	49.6	52.8	50.4
	FDA	7.9	8.0	8.7	8.2
	dice	6.9	7.0	3.9	3.6
% of ALL	FDA	17	16	16	16
	dice	14	14	7.4	7.1

Table 5.7: Performance for the out-of-domain translation task of Callison-Burch et al. (2011). ALL corresponds to the baseline system using all of the parallel training sentences.

separate Moses systems using all of the parallel training sentences for the language pairs *en-de*, *de-en*, *en-es*, and *es-en*. We created training sets using all of the features of the test set to select training instances. The results given in Table 5.7 show that we can achieve similar BLEU performance using about 7% of the parallel training sentences target words (200,000 instances) using *dice* and about 16% using FDA. In the out-of-domain translation task, we are able to reduce the training set size to achieve a performance close to the baseline. The sample points presented in the table is chosen proportional to the relative sizes of the parallel training sentences sizes of WMT'10 and WMT'11 datasets and the training set size of the peak in Figure 5.3. We may be able to achieve better performance in the out-of-domain task as well. The sample points in Table 5.7 may be on either side of the peak.

5.5 Contributions

We have introduced the feature decay algorithms (FDA), a class of instance selection algorithms that use feature decay, which achieve better target coverage than previous work and achieve significant gains in translation performance. We find that decaying feature weights has significant effect on the performance. We demonstrate that target coverage and translation performance are correlated, showing that target coverage is also a good indicator of BLEU performance. We have shown that target coverage provides an upper bound on the translation

performance with a given training set.

We are able to improve the BLEU score by ∼2 using about 20% of the available training data in terms of target words with FDA and by ∼1 with only about 5%. We have also shown that by training on only 3000 instances per sentence we can reach within 1 BLEU difference to the baseline system. As the length of the parallel sentences decrease, word alignment task becomes easier. We develop *dice* sentence selection technique that optimizes coverage and word alignment performance and at the same time reduces the average length of the selected sentences. In the out-of-domain translation task, we are able to reduce the training set size to about 7% and achieve similar performance with the baseline using FDA. *dice* method reduces the training size further.

Our results demonstrate that SMT systems can improve their performance by transductive training set selection. The results show that making relevant instance selection is important and the performance increase is likely to be due to the reduction in the number of noisy or irrelevant training instances and the increased precision in the probability estimates in the generated phrase tables. We have shown how to select instances and achieved significant performance improvements.

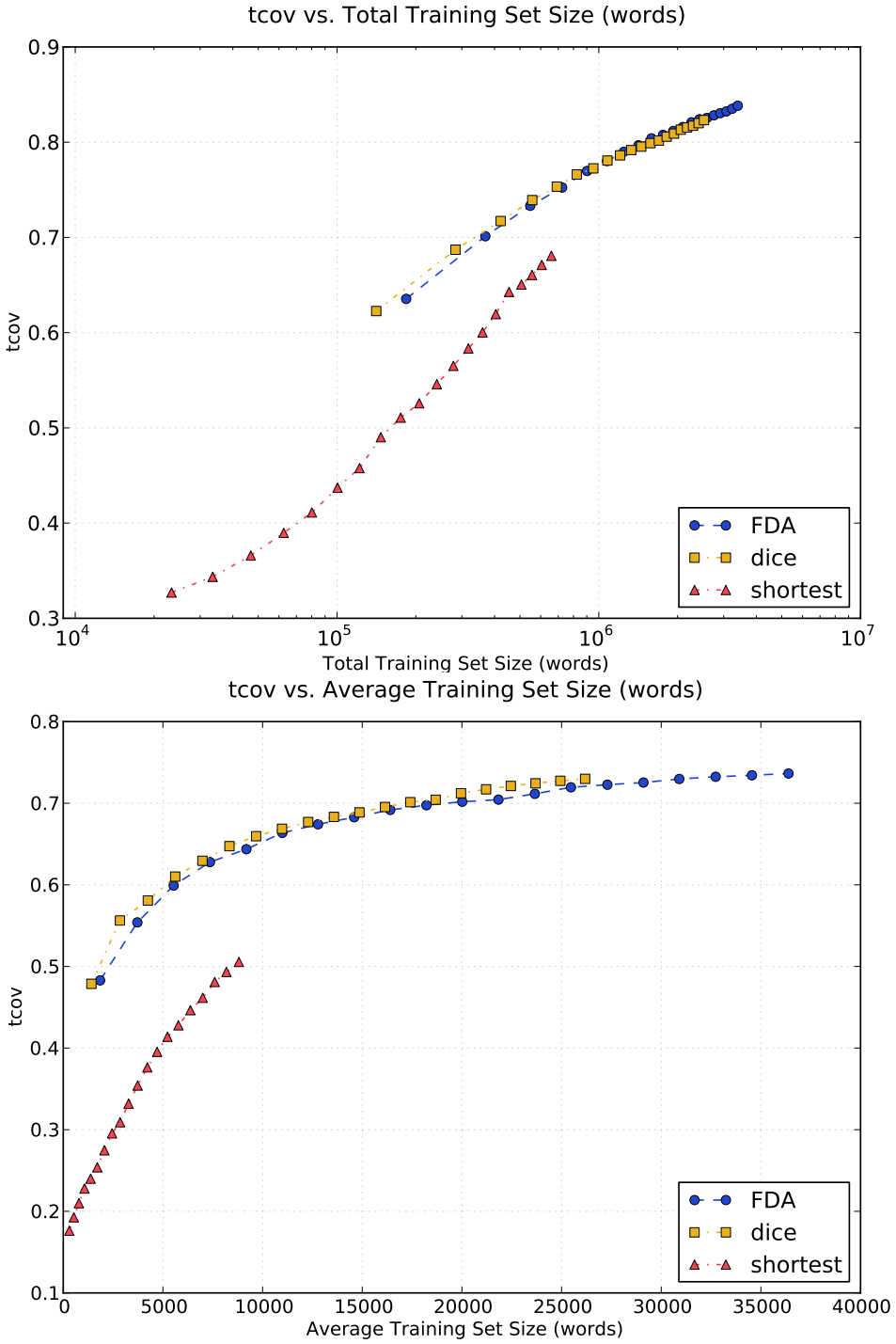

Figure 5.4: Target coverage per target words comparison. Figure shows the rate of increase in *tcov* as the size of $\mathcal{L}$ increase. Target coverage curves for total training set size is given on the left plot and for average training set size per test sentence on the right plot.

Chapter 6

L_1 Regularized Regression for Reranking and System Combination in Machine Translation

This chapter shows that regression mapping score can be used to improve over a baseline SMT system by reranking the N-best lists generated by it. We use L_1 regularized transductive regression to learn mappings between source and target features of the training sets derived for each test sentence and use these mappings to rerank translation outputs. We compare the effectiveness of L_1 regularization techniques for regression to learn mappings between features given in a sparse feature matrix. The results show the effectiveness of using L_1 regularization versus L_2 used in ridge regression. We show that regression mapping is effective in reranking translation outputs and in selecting the best system combinations with encouraging results on different language pairs. Part of this work is published as (Biçici and Yuret, 2010b).

6.1 Introduction

Regression can be used to find mappings between the source and the target feature sets derived from given parallel training sentences. Transductive learning uses a subset of the training examples that are closely related to the test set without using the model induced by the

full training set. In the context of SMT, we select a few training instances for each test instance to guide the translation process. This also gives us a computational advantage when considering the high dimensionality of the problem. The goal in transductive regression based machine translation (RegMT) is both reducing the computational burden of the regression approach by reducing the dimensionality of the training set and its feature representation and also improving the translation quality by using transduction. Transductive regression is shown to achieve higher accuracy than L_2 regularized ridge regression on some machine learning benchmark datasets (Chapelle et al., 1999).

In an idealized feature mapping matrix where features are word sequences, we would like to observe few target features for each source feature derived from a source sentence. In this setting, we can think of feature mappings being close to permutation matrices with few nonzero item for each column. L_1 regularization helps us achieve solutions close to the permutation matrices by increasing sparsity.

We show that L_1 regularized regression mapping is effective in reranking translation outputs and present encouraging results on different language pairs in the translation task of Callison-Burch et al. (2010) (WMT'10). In the system combination task, different translation outputs of different translation systems are combined to find a better translation. We model system combination task as a reranking problem among the competing translation models and obtain statistically significant improvements over the baseline system results with the RegMT system.

Outline: In section 6.2 we present our translation experiments where we discuss how we add the brevity penalty to the RegMT score and combine other sources of information about translations for reranking translation outputs. We achieve significant improvements over the baseline system output. We then discuss results on system combination in section 6.3 using reranking on both WMT'10 and WMT'11 challenges. The last section presents our contributions.

6.2 Translation Experiments

We perform experiments on the translation tasks of English-German (*en-de*), German-English (*de-en*), English-French (*en-fr*), English-Spanish (*en-es*), and English-Czech (*en-cz*) (in source language-target language format) using the paired training corpora provided by Callison-Burch et al. (2010) (WMT'10). We discuss the datasets and the baseline systems developed, the training instance selection technique used, how we incorporate the brevity penalty into the score, and the results we obtain using reranking.

6.2.1 Datasets and Baseline

We developed separate SMT models using Moses (Koehn et al., 2007) with the default settings including a maximum sentence length limit of 80 tokens and a 5-gram target language model and obtained distinct 100-best lists for the test sets. All systems were tuned with 2051 sentences and tested with 2525 sentences. We have randomly picked 100 instances from the development set to be used in tuning the regression experiments (*dev.100*). The translation challenge test set contains 2489 sentences. Number of sentences in the training set of each system and baseline performances for uncased output (test set BLEU, challenge test set BLEU) are given in Table 6.1.

Corpus	# **sent**	**BLEU**	**BLEU** Challenge
en-de	1,609,988	.1471	.1309
de-en	1,609,988	.1943	.1556
en-fr	1,728,965	.2281	.2049
en-es	1,715,158	.2237	.2106
en-cz	7,320,238	.1452	.1145

Table 6.1: Initial uncased performances of the translation systems for WMT'10 where the third and fourth columns list the results on the test set with 2525 sentences and the challenge test set with 2489 sentences respectively.

Feature mappers used are 3-spectrum weighted word kernel (Equation 3.11), which considers all n-grams up to order 3 weighted by the number of tokens in the feature. We segment sentences using some of

the punctuation for managing the feature set better and do not consider n-grams that cross segments. We use BLEU (Papineni et al., 2001) and NIST (Doddington, 2002) evaluation metrics for measuring the performance of translations automatically.

6.2.2 Instance Selection for Transductive Regression

Transduction uses test instances, which can sometimes be accessible at training time, to learn specific models tailored towards the test set. Transduction has computational advantages by not using the full training set and by having to satisfy a smaller set of constraints as defined by the training instances and their features. For each test sentence, we pick a limited number of training instances designed to improve the coverage of correct features to build a regression model. section 3.6 details the instance selection method that we use.

6.2.3 Addition of the Brevity Penalty

Detailed analysis of the results shows that RegMT scored best translation achieves better n-gram match percentages than the Moses translation but suffers from the brevity penalty due to selecting shorter translations. Due to using a cost function that minimizes the squared loss, RegMT score tends to select shorter translations when the coverage is low. We also observe that we are able to achieve higher scores for NIST, which suggests the addition of a brevity penalty to the score.

Precision based BLEU scoring divides n-gram match counts to n-gram counts found in the translation and this gives an advantage to shorter translations. Therefore, a brevity penalty (BP) is added to penalize short translations:

$$\texttt{BP} = \min(1 - \frac{\text{ref-length}}{\text{trans-length}}, 0) \tag{6.1}$$

$$BLEU = \exp\left(\log(\texttt{ngram}_{prec}) + \texttt{BP}\right) \tag{6.2}$$

where $\texttt{ngram}_{prec}$ represent the sum of n-gram precisions. We give more details about the BLEU score in section 2.4. Moses rarely incurs BP

as it has a word penalty parameter optimized against BLEU which penalizes translations that are too long or too short. For instance, Moses 1-best translation for *en-de* system achieves 0.1309 BLEU versus 0.1320 BLEU without BP.

We handle short translations in two ways. We optimize the λ parameter of QP, which manages the sparsity of the solution (larger λ values correspond to sparser solutions) against BLEU score rather than the squared loss. Optimization yields $\lambda = 20.744$. We alternatively add a BP cost to the squared loss:

$$\mathtt{BP} = \exp\left(\min(1 - \frac{|\Phi_Y(\mathbf{y})|}{|\ulcorner \mathbf{W}\Phi_X(\mathbf{x}) + \alpha_{\mathtt{BP}} \urcorner|}, 0)\right) \tag{6.3}$$

$$f(\mathbf{x}) = \arg\min_{\mathbf{y} \in Y^*} \| \Phi_Y(\mathbf{y}) - \mathbf{W}\Phi_X(\mathbf{x}) \|^2 + \lambda_{\mathtt{BP}}\mathtt{BP} \tag{6.4}$$

where $|.|$ denotes the length of the feature vector, $\ulcorner . \urcorner$ rounds feature weights to integers, $\alpha_{\mathtt{BP}}$ is a constant weight added to the estimation, and $\lambda_{\mathtt{BP}}$ is the weight given for the BP cost. $|\ulcorner \mathbf{W}\Phi_X(\mathbf{x}) + \alpha_{\mathtt{BP}} \urcorner|$ represents an estimate of the length of the reference as found by the RegMT system. This BP cost estimate is similar to the cost used in (Serrano et al., 2009) normalized by the length of the reference. We found $\alpha_{\mathtt{BP}} = 0.1316$ and $\lambda_{\mathtt{BP}} = -13.68$ when optimized on the *en-de* system. We add a BP penalty to all of the reranking results given in the next section and QP results also use optimized λ.

6.2.4 Reranking Experiments

We rerank N-best lists by using linear combinations of the following scoring functions:

1. RegMT: Transductive regression based machine translation scores as found by Equation 3.9.
2. TM: Translation model scores we obtain from the baseline SMT system that is used to generate the N-best lists.

3. LM: 5-gram language model scores that the baseline SMT system uses when calculating the translation model scores.

The training set we obtain may not contain all of the features of the reference target due to low coverage. Therefore, when performing reranking, we also add the cost coming from the features of $\Phi_Y(\mathbf{y})$ that are not represented in the training set to the squared loss as in:

$$\| \Phi_Y(\mathbf{y}) \setminus F_Y \|^2 + \| \Phi_Y(\mathbf{y}) - \mathbf{W}\Phi_X(\mathbf{x}) \|^2, \tag{6.5}$$

where $\Phi_Y(\mathbf{y}) \setminus F_Y$ represent the features of $\mathbf{y}$ not represented in the training set.

We note that RegMT score only contains ordering information as present in the 2/3-gram features in the training set. Therefore, the addition of a 5-gram LM score as well as the TM score, which also incorporates the LM score in itself, improves the performance. We are not able to improve the BLEU score when we use the RegMT score by itself however we are able to achieve improvements in the NIST and 1-WER scores. The performance increase is important for two reasons. First of all, we are able to improve the performance using blended spectrum 3-gram features against translations obtained with 5-gram language model and higher order features. Outperforming higher order n-gram models is known to be a difficult task (Galley and Manning, 2009). Secondly, increasing the performance with reranking itself is a hard task since possible translations are already constrained by the ones observed in N-best lists. Therefore, an increase in the N-best list size may increase the score gaps.

Table 6.2 [1] presents reranking results on all of the language pairs we considered, using RegMT, TM, and LM scores with the combination weights learned on the development set. We are able to achieve better BLEU and NIST scores on all of the listed systems. We are able to see up to 0.38 increase in the BLEU score for the *en-es* pair. Baseline row corresponds to the baseline system results where the top sentence from the N-best list as returned by the SMT system is used as the

[1] We give the details of statistical significance testing in general and for SMT in Appendix B.

	en-de		*de-en*		*en-fr*		*en-es*		*en-cz*	
Model	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**
Baseline	.1309	5.1417	.1556	5.4164	.2049	6.3194	.2106	6.3611	.1145	4.5008
Oracle	.1811	6.0252	.2101	6.2103	.2683	7.2409	.2770	7.3190	.1628	5.4501
L2	**.1319**	**5.1680**	.1555	**5.4344**	.2044	**6.3370**	**.2132**	**6.4093**	.1148	**4.5187**
FSR	*.1317**	**5.1639**	.1559	**5.4383**	.2053	**6.3458**	**.2144**	**6.4168**	.1150	**4.5172**
LP	.1317	**5.1695**	.1561	**5.4304**	.2048	6.3245	.2109	**6.4176**	.1124	4.5143
QP	.1309	**5.1664**	.1550	**5.4553**	.2033	*6.3354**	**.2121**	**6.4271**	.1150	**4.5264**

Table 6.2: Reranking results on the translation task using RegMT, TM, and LM scores on the WMT'10 datasets. We use approximate randomization test (Riezler and Maxwell, 2005) with 1000 repetitions to determine score difference significance: results in **bold** are significant with $p \leq 0.01$ and *italic* results with (*) are significant with $p \leq .05$. The difference of the remaining from the baseline are not statistically significant.

translation. Oracle row corresponds to picking the best translation using reranking with the BLEU scoring metric as evaluated with the reference translations at the sentence level.

If we used only the TM and LM scores when reranking with the *en-de* system, then we would obtain 0.1309 BLEU and 5.1472 NIST scores. We only see a minor increase in the NIST score and no change in the BLEU score with this setting when compared with the baseline given in Table 6.2. Therefore, we see that RegMT score improves the results when used in combination with the TM and LM scores.

Due to computational reasons, we do not use the same number of instances to train different models. In our experiments, we used $n = 3$ for L2, $n = 1.5$ for FSR, and $n = 1.2$ for QP and LP solutions to select the number of instances in Equation 3.16. The average number of instances used per sentence in training corresponding to these choices are approximately 140, 74, and 61. Even with these decreased number of training instances, L_1 regularized regression techniques are able to achieve comparable scores to L_2 regularized regression model or better in Table 6.2.

6.3 System Combination Experiments

We perform experiments on the system combination task for the English-German, German-English, English-French, English-Spanish, and English-

Model	*en-de*		*de-en*		*en-fr*		*en-es*		*en-cz*	
	BLEU	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**
Random	.1490	5.6555	.2088	6.4886	.2415	6.8948	.2648	7.2563	.1283	4.9238
Best	.1658	5.9610	.2408	6.9861	.2864	7.5272	.3047	7.7559	.1576	5.4480
L2	.1694	5.9974	.2336	6.9398	.2948	7.7037	.3036	7.8120	.1657	5.5654
FSR	.1689	5.9638	.2357	6.9254	.2947	7.7107	.3049	7.8156	.1657	5.5632
LP	.1694	5.9954	.2368	6.8850	.2928	7.7157	.3027	7.7838	.1659	5.5680
QP	.1692	5.9983	.2368	6.9172	.2913	7.6949	.3040	7.8086	.1662	5.5785

Table 6.3: Reranking results on the system combination task using RegMT, TM, and LM scores on the WMT'10 datasets. underlined correspond to the best score in each rectangle of scores.

Czech language pairs using the training corpus provided in WMT'10.

6.3.1 WMT'10 Datasets

We use the training set provided in WMT'10 to index and select transductive instances from. The challenge split the test set for the translation task of 2489 sentences into a tuning set of 455 sentences and a test set with the remaining 2034 sentences. Translation outputs for each system is given in a separate file and the number of system outputs per translation pair varies. We have tokenized and lowercased each of the system outputs and combined these in a single N-best file per language pair. We also segment sentences using some of the punctuation for managing the feature set better. We use these N-best lists for RegMT reranking to select the best translation model. Feature mappers used are again 3-spectrum weighted word kernels (Equation 3.11).

6.3.2 WMT'10 Experiments

We rerank N-best lists by using combinations of the following scoring functions:

1. RegMT: Transductive regression based machine translation scores as found by Equation 3.9.
2. CBLEU: Comparative BLEU scores we obtain by measuring the average BLEU performance of each translation relative to the other systems' translations in the N-best list.

3. LM: We calculate 5-gram language model scores for each translation using the language model trained over the target corpus provided in the translation task.

Since we do not have access to the reference translations nor to the translation model scores each system obtained when generating each translation, we estimate translation model performance (CBLEU) by measuring the average BLEU performance of each translation relative to the other translations in the N-best list. Thus, each possible translation in the N-best list is BLEU scored against other translations and the average of these scores is selected as the TM score for the sentence. Sentence level BLEU score calculation avoids singularities in n-gram precisions by taking the maximum of the match count and $\frac{1}{2|s_i|}$ for $|s_i|$ denoting the length of the source sentence s_i as used in (Macherey and Och, 2007).

Table 6.3 presents reranking results on all of the language pairs we considered, using RegMT, CBLEU, and LM scores with the same combination weights as above. Random model score lists the random model performance selected among the competing translations randomly and it is used as a baseline. Best model score lists the performance of the best model performance. We are able to achieve better BLEU and NIST scores in all of the listed systems except for the *de-en* language pair when compared with the performance of the best competing translation system. The lower performance in the *de-en* language pair may be due to having a single best translation system that outperforms others significantly. The difference between the best model performance and the mean as well as the variance of the scores in the *de-en* language pair is about twice their counterparts in the *en-de* language pair.

Due to computational reasons, we do not use the same number of instances to train different models. In our experiments, we used $n = 4$ for L2, $n = 1.5$ for FSR, and $n = 1.2$ for QP and LP solutions to select the number of instances in Equation 3.16. The average number of instances used per sentence in training corresponding to these choices are approximately 189, 78, and 64.

BLEU	*en-de*	*de-en*	*en-es*	*es-en*	*ht-en*
Min	.1064	.1572	.2174	.1976	.2281
Max	.1727	.2413	.3375	.3009	.3708
2^{nd} best	.1572	.2302	.3301	.2973	.3288
Average	.1416	.1997	.292	.2579	.2993
Oracle	.2529	.3305	.4265	.4233	.4336
RegMT	.1631	.2322	.3311	.3052	.3234

Table 6.4: System combination results on the WMT'11 datasets.

6.3.3 WMT'11 Results

We perform experiments on the system combination task for the English-German, German-English, English-Spanish, Spanish-English (*es-en*), and Haitian Creole-English (*ht-en*) language pairs using the training corpus provided in WMT'11 (Callison-Burch et al., 2011). We have tokenized and lowercased each of the system outputs and combined these in a single N-best file per language pair. We use these N-best lists for reranking by RegMT to select the best translation model. Feature mappers used are 2-spectrum counting word kernels (Equation 3.11).

Table 6.4 presents reranking results on all of the language pairs we considered, using RegMT, CBLEU, and LM scores with the same combination weights as above. We also list the performance of the best model (Max) as well as the worst (Min). We are able to achieve close or better BLEU scores in all of the listed systems when compared with the performance of the best translation system except for the *ht-en* language pair. The lower performance in the *ht-en* language pair may be due to having a single best translation system that outperforms others significantly. This happens for instance when an unconstrained model use external resources to achieve a significantly better performance than the second best model. 2^{nd} best in Table 6.4 lists the second best model's performance to estimate how much the best model's performance is better than the rest.

RegMT model may prefer sentences with lower BLEU, which can sometimes cause it to achieve a lower BLEU performance than the best model. This is clearly the case for *en-de* with 1.6 BLEU points

difference with the second best model performance and for *de-en* task with 1.11 BLEU points difference. Also this observation holds for *en-es* with 0.74 BLEU points difference and for *ht-en* with 4.2 BLEU points difference. For *es-en* task, there is 0.36 BLEU points difference with the second best model and these models likely to complement each other.

System combination models may segment alternative translations to generate a new one. Reranking approach can achieve a better performance only when competing models exist in a case similar to the XOR truth value: it performs good when one of them performs better than the other on different instances. If a model performs much better than the others, then it makes it harder to achieve a performance better than the best model.

The existence of complementing SMT models is important for the reranking approach to achieve a performance better than the best model, as there is a need for the existence of a model performing better than the best model on some test sentences. We can use the competitive SMT model to achieve the performance of the best with a guarantee even when a single model is dominating the rest (Biçici and Kozat, 2010). For competing translation systems in an online machine translation setting, adaptively learning of model weights can be performed based on the previous translation performance. We discuss these techniques in chapter 7.

6.4 Contributions

We use transductive regression to learn mappings between source and target features of given parallel training sentences and use these mappings to rerank translation outputs. We compare the effectiveness of L_1 regularization techniques for regression. RegMT score has a tendency to select shorter translations when the coverage is low. We incorporate a brevity penalty to the squared loss and optimize the λ parameter of QP to tackle this problem and further improve the performance of the

system.

The results show the effectiveness of using L_1 regularization versus L_2 used in ridge regression. Proper selection of training instances plays an important role to learn correct feature mappings with limited computational resources accurately. We notice that we may improve on the training instance selection technique that we use. We investigate and use better instance selection methods that optimize for coverage of the test set features in chapter 5.

Reranking approach shows that RegMT score is valuable in selecting the good translations among the alternatives in an N-best list. We show that RegMT system is useful with reranking system outputs and with system combination results. We can also incorporate a decoder to actually generate translation results. We experiment with graph decoding and decoding with Moses in chapter 8.

Chapter 7

Adaptive Model Weighting and Transductive Regression for Reranking Machine Translation Outputs

In this chapter, we analyze adaptive model weighting techniques for reranking using candidate translation scores obtained by L_1 regularized transductive regression for translation outputs generated by different translation models. Competitive statistical machine translation is an online learning technique for sequential translation tasks where we try to select the best among competing statistical machine translators. The competitive predictor assigns a probability per model weighted by the sequential performance. We define additive, multiplicative, and loss-based weight updates with exponential loss functions for competitive statistical machine translation. Without any pre-knowledge of the performance of the translation models, we succeed in achieving the performance of the best model in all translation experiments and surpass their performance in most of the language pairs we considered. Part of this work is published as (Biçici and Kozat, 2010).

7.1 Introduction

In the system combination task of machine translation, multiple translation system outputs are combined to achieve a better performance than the translations of all the competing translation models. When we view each sentence to be translated as independent instances, system combination task can be solved with a sequential learning algorithm. Online learning algorithms enable us to benefit from previous good model choices to estimate the next best model. Online SMT receives test source sentences one by one and generates a translation for each instance, which can be modeled with the reranking approach given an N-best list. We use transductive regression based machine translation model to estimate the scores for each sentence.

We analyze adaptive model weighting techniques for system combination when the competing translators are statistical machine translation (SMT) models. We use separate model weights weighted by the sequential performance. We use additive, multiplicative, or loss based techniques to update model weights. Without any pre-knowledge of the performance of the translation models, we show that we can achieve the performance of the best model in all translation experiments and we can surpass its performance as well as the regression based machine translation's performance in some cases.

We provide empirical results on the performance of various class of algorithms including results with some possible variations for a range of translation language pairs. We experiment in two main settings: *simulated online learning* setting where only the scores obtained from the regression model are used and *online learning* setting where reference translations are used to estimate the performance for each instance.

Outline: The next section presents competitive statistical machine translation model for solving sequential translation tasks with competing translation models. We present techniques for estimating the best performing translation model and discuss how we handle model selection and adjust learning rate. Then we discuss the transductive

regression approach for machine translation, which we use to obtain instance scores. section 7.4 presents our experimental setup and the algorithm we use for predicting the best translation model. In section 7.5 we give the results on the test sets. The last section gives a summary of our contributions.

7.2 Adaptive Model Weighting for Statistical Machine Translation

We develop the adaptive model weighting techniques for statistical machine translation for sequential translation tasks when the statistical machine translators compete to achieve a better overall performance. The resulting competitive SMT framework uses the output of different translation models to achieve a translation performance that surpasses the translation performance of all of the component models or achieves the performance of the best.

Competitive SMT (CSMT) uses online learning to update the weights used for estimating the best performing translation model. Competitive predictor assigns a weight per model estimated by their sequential performance. The sequential prediction problem we investigate is in the following setting. Given a sequence of observations, $\mathbf{x}[n] = (x[1], \ldots, x[n])$ and their previous outcomes, $\mathbf{y}[n-1] = (y[1], \ldots, y[n-1])$, the observer predicts $y[n]$. In machine translation, the observations, $\mathbf{x}[i]$, represent the source sentences and the outcomes, $\mathbf{y}[i]$, represent the reference target sentences. At each step, M component translation models are executed in parallel over the input source sentence sequence and the loss $l_p[n]$ of model p at observation n is calculated by comparing the desired data $y[n]$ with the output of model p, $\hat{y_p}[n]$. Individual performances of the models at each observation are also stored in $\mathbf{x}[n]$. CSMT selects a model based on the weights and the performance of the models and adaptively updates the weights given for each model. This corresponds to learning in *full information setting* where we have access to the loss for each action (Blum and

Mansour, 2007).

CSMT learning involves two main steps: *estimation* and *weight update*:

$$\begin{aligned} \hat{y}_c[n] &= E(\mathbf{w}[n], \mathbf{x}[n]), && (\texttt{estimation}) \\ l_p[n] &= y[n] - \hat{y}_p[n], && (\texttt{instance loss}) \\ \mathcal{L}_p[n] &= \textstyle\sum_{i=1}^{n} l_p[i]^2, && (\texttt{cumulative loss}) \\ \mathbf{w}[n+1] &= U(\mathbf{w}[n], \hat{y}_c[n], \boldsymbol{\mathcal{L}}[n]), && (\texttt{update}) \end{aligned} \tag{7.1}$$

where $\mathbf{w}[n] = (w_1[n], \ldots, w_M[n])$ stores the weights for M models, $\mathcal{L}_p$ is the cumulative squared loss of model p, $\boldsymbol{\mathcal{L}}[n]$ stores cumulative and instance losses, and $\hat{y}_c[n]$ is the competitive model estimated for instance n. The learning problem is finding an adaptive $\mathbf{w}$ that minimizes the cumulative squared error with appropriate estimation and update methods.

7.2.1 Related Work

Multistage adaptive filtering (Kozat and Singer, 2002) combines the output of multiple adaptive filters to outperform the best among them where the first stage executes models in parallel and the second stage updates parameters using the performance of the combined prediction, $\hat{y}_c[n]$. Macherey and Och (2007) investigate different approaches for system combination including candidate selection that maximize a weighted combination of BLEU scores among different system outputs. Their system uses a fixed weight vector trained on the development set to be multiplied with instance BLEU scores.

7.2.2 Estimating the Best Performing Translation Model

We use additive, multiplicative, or loss based updates to estimate model weights. We measure instance loss with $\texttt{trLoss}(y[i], \hat{y}_p[i])$, which is a function that returns the translation performance of the

output translation of model p with respect to the reference translation at instance i. 1-BLEU (Papineni et al., 2001) is one such function with outputs in the range $[0, 1]$. Cumulative squared loss of the p-th translation model is defined as:

$$\mathcal{L}_p[n] = \sum_{i=1}^{n} \texttt{trLoss}(y[i], \hat{y}_p[i])^2. \tag{7.2}$$

We use *exponentially re-weighted prediction* to estimate model performances, which uses exponentially re-weighted losses based on the outputs of the m different translation models.

We define the *additive* exponential weight update as follows:

$$w_p[n+1] = \frac{w_p[n] + e^{-\eta[n]\, l_p[n]}}{\sum_{k=1}^{M} \left(w_k[n] + e^{-\eta[n]\, l_k[n]} \right)}, \tag{7.3}$$

where $\eta > 0$ is the learning rate and the denominator is used for normalization. The update amount, $e^{-\eta[n]\, l_p[n]}$ is 1 when $l_p[n] = 0$ and it approaches zero with increasing instance loss. Perceptrons, gradient descent, and Widrow-Huff learning have additive weight updates.

We define the *multiplicative* exponential weight update as follows:

$$w_p[n+1] = w_p[n] \times \frac{e^{-\eta[n]\, l_p[n]^2}}{\sum_{k=1}^{M} w_k[n]\; e^{-\eta[n]\, l_k[n]^2}}, \tag{7.4}$$

where we use the squared instance loss. Equation 7.4 is similar to the update of Weighted Majority Algorithm (Littlestone and Warmuth, 1992) where the weights of the models that make a mistake are multiplied by a fixed β such that $0 \leq \beta < 1$.

The *Winnow* algorithm (Littlestone, 1988) makes multiplicative updates, however, performs an update only if there is a mistake with a fixed update amount. We present a Winnow-type algorithm that uses

instance loss with positive exponential update and fixed η:

$$w_p[n+1] \;=\; w_p[n] \times \frac{e^{\eta[n]\, l_p[n]}}{\sum_{k=1}^{M} w_k[n]\; e^{\eta[n]\, l_k[n]}}. \tag{7.5}$$

We use *Bayesian Information Criterion (BIC)* as a *loss based* re-weighting technique. Assuming that instance losses are normally distributed with variance σ^2, BIC score is obtained as (Hastie et al., 2009):

$$\mathrm{BIC}_p[n] = \frac{\mathcal{L}_p[n]}{\sigma^2} + d_p \log(n), \tag{7.6}$$

where σ^2 is estimated by the average of model sample variances of squared instance loss and d_p is the number of parameters used in model p which we assume to be the same for all models; therefore we can discard the second term. The model with the minimum BIC value becomes the one with the highest posterior probability where the posterior probability of model p can be estimated as (Hastie et al., 2009):

$$w_p[n+1] = \frac{e^{-\frac{1}{2}\mathrm{BIC}_p[n]}}{\sum_{k=1}^{M} e^{-\frac{1}{2}\mathrm{BIC}_k[n]}}. \tag{7.7}$$

The posterior probabilities become model weights and we basically forget about the previous weights, whose information is presumably contained in the cumulative loss, $\mathcal{L}_p$. We define multiplicative re-weighting with BIC scores as follows:

$$w_p[n+1] = w_p[n] \times \frac{e^{-\frac{1}{2}\mathrm{BIC}_p}}{\sum_{k=1}^{M} w_k[n]\; e^{-\frac{1}{2}\mathrm{BIC}_k}}. \tag{7.8}$$

7.2.3 Model selection

We use *stochastic* or *deterministic* selection to choose the competitive model for each instance. Deterministic choice randomly selects among the maximum scoring models with minimum translation length to reduce ties whereas stochastic choice draws model p with probability proportional to $w_p[n]$. Randomization with the stochastic model selection decreases expected mistake bounds in the weighted majority algorithm (Blum, 1996; Littlestone and Warmuth, 1992).

7.2.4 Variations

We can obtain a number of variations of the presented online learning algorithms.

Learning rate: We can choose the learning rate adaptively based on the incurred loss so far instead of using a constant value. Auer et al. (2002) show that optimal fixed learning rate for the weighted majority algorithm is found as $\eta[n] = \sqrt{M/\mathcal{L}_*[n]}$ where $\mathcal{L}_*[n] = \min_{1 \leq i \leq M} \mathcal{L}_i[n]$, which requires prior knowledge of the cumulative losses.

Averaging: Averaging is used in averaged perceptron to improve the performance (Collins, 2002). We perform averaging as follows: the weights obtained at step $n+1$ is averaged over the last k weights: $\mathbf{w}[n+1] = \sum_{j=0}^{k-1} \mathbf{w}[n-j] \;/\; k$.

Update when correct: Especially the multiplicative updates may suffer from the "winner takes all" phenomenon where the best model's weight becomes dominant over others. To prevent such cases, we also test with updating only when the selected model differs from the best model as it is done in the Winnow algorithm.

7.3 Transductive Regression Based Translation Scoring

Transduction uses test instances, which can sometimes be accessible at training time, to learn specific models tailored towards the test set. Transduction has computational advantages since we are not using the full training set and a smaller set of constraints exist to satisfy. Transductive regression based machine translation (RegMT) aims to reduce the computational burden of the regression approach by reducing the dimensionality of the training set and the feature set and also improve the translation quality by using transduction. We select a subset of the training set for each test sentence to train regression models. For each test sentence, we pick a limited number of training instances designed to improve the coverage of correct features to build a regression model. We use the instance selection technique given in section 3.6. We use L_1 regularized regression to deal with the sparsity of the features derived from a small training set.

7.4 Experimental Setup

We give empirical results and analysis for the performance of the models considered in reranking machine translation outputs coming from different systems. We perform experiments on the system combination task for the English-German (*en-de*), German-English (*de-en*), English-French (*en-fr*), English-Spanish (*en-es*), and English-Czech (*en-cz*) language pairs using the translation outputs for all the competing systems provided in (Callison-Burch et al., 2010) (WMT'10).

We experiment in two main settings: *simulated online learning*, where only the scores obtained from the RegMT system are used, and *online learning*, where reference translations are used to estimate instance losses. We do not use reference translations in measuring instance performance in the simulated online learning setting for the results we obtain be in line with system combination challenge's goals.

We use model weights, instance scores as estimated by the RegMT

model, and the instance losses found by the RegMT model scores or reference translations to achieve performances better than the competing models. For the learning rate, we use $\eta = \sqrt{M/(0.05n)}$ when η is constant and set $\eta[n] = \sqrt{M/(\mathcal{L}_*[n] + c)}$ when it is adaptive with constant $c = 3$.

7.4.1 Datasets

We use the training set provided in WMT'10 to index and select transductive instances from. The challenge split the test set for the translation task of 2489 sentences into a tuning set of 455 sentences and a test set with the remaining 2034 sentences. Translation outputs for each system is given in a separate file and the number of system outputs per translation pair varies. We have tokenized and lowercased each of the system outputs and combined these in a single N-best file per language pair to be used in reranking. We use BLEU (Papineni et al., 2001) automatic evaluation metric for measuring the performance of the translations automatically.

7.4.2 Reranking Scores

The problem we are solving is online learning with prior information, which comes from the comparative BLEU scores, LM scores, and RegMT scores at each step n. Scoring functions are explained below:

1. RegMT: Transductive regression based machine translation scores as found by Equation 3.9. We use the RegMT scores obtained by the FSR model (subsection 4.3.1).
2. CBLEU: Comparative BLEU scores we obtain by measuring the average BLEU performance of each translation relative to the other systems' translations in the N-best list.
3. LM: We calculate 5-gram language model scores for each translation using the language model trained over the target corpus provided in the translation task.

To make things simpler, we use a single prior RegMT system score representing the linear combination of the three scores mentioned with weights learned on the tuning set. The overall RegMT system score for instance n, model i is referred as $\texttt{RegScore}_i[n]$.

Since we do not have access to the reference translations nor to the translation model scores each system obtained for each sentence, we estimate translation model performance by measuring the average BLEU performance of each translation relative to other translations in the N-best list. Thus, each possible translation in the N-best list is BLEU scored against other translations and the average of these scores is selected as the CBLEU score for the sentence. Sentence level BLEU score calculation avoids singularities in n-gram precisions by taking the maximum of the match count and $\frac{1}{2|s_i|}$ for $|s_i|$ denoting the length of the source sentence s_i as used in (Macherey and Och, 2007).

7.4.3 Adaptive Model Weighting for Predicting Best Translations

The generic adaptive model weighting algorithm is given in Algorithm 3. The algorithm has an additional update step, which estimates the model after incorporating the current instance's scores in the estimation without changing the losses. If averaging is performed with a window of the last N weight vectors, then it is applied after weight updates, where $\mathbf{w}[j-N:j]$ represent the last N weight vectors obtained. Model selection is another parameter affecting the model estimation function: $E(\mathbf{w}[j], \texttt{Model.selection})$. For simulated online learning, $y[n]$ are the model with the highest RegMT score and for online learning, they are the model with the closest translation to the reference as measured by the BLEU scoring metric. We initialize the weights to $1/M$ for all models.

Table 7.1 presents the simulated online learning performances of the weight update algorithms on the *en-de* development set. Add. stands for additive, Mul. for multiplicative, and BICW for BIC with weighting weight update algorithms. We have measured their performances with

Algorithm 3: Best Translation Estimation with Adaptive Model Weighting

$\mathbf{w}_i[1] \leftarrow 1/M \quad \forall i, 1 \leq i \leq M$
$\eta \leftarrow \sqrt{M/(0.05n)}$
$c \leftarrow 3$
$N \in \{1, 5, 10, 50\}$
for $j \leftarrow 1$ ***to*** n **do**
    if *Mixture.weighting* **then**
        $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot \texttt{RegScore}[j]$
    $\hat{y}_c[j] \leftarrow E(\mathbf{w}[j], \texttt{Model.selection})$
    $\mathbf{w}[j] \leftarrow U(\mathbf{w}[j], \hat{y}_c[j], \boldsymbol{\mathcal{L}}[j])$ `// Initial update`
    $\hat{y}_c[j] \leftarrow E(\mathbf{w}[j], \texttt{Model.selection})$
    if ***not*** *Update.when.correct* ***and*** $y[j] \neq \hat{y}_c[j]$ **then**
        $(\mathbf{w}[j+1], \boldsymbol{\mathcal{L}}[j+1]) \leftarrow U(\mathbf{w}[j], \hat{y}_c[j], \boldsymbol{\mathcal{L}}[j])$
    else
        $(\mathbf{w}[j+1], \boldsymbol{\mathcal{L}}[j+1]) \leftarrow U(\mathbf{w}[j], \hat{y}_c[j], \boldsymbol{\mathcal{L}}[j])$
    if *Averaging* **then**
        $\mathbf{w}[j+1] \leftarrow \texttt{Average}(\mathbf{w}[j-N:j])$
    if *Dynamic.*η **then**
        $\eta[j] = \sqrt{M/(\mathcal{L}_*[j] + c)}$ where $\mathcal{L}_*[j] = \min_{1 \leq i \leq M} \mathcal{L}_i[j]$

four different setting combinations: (1) Model selection: S: Stochastic or D: Deterministic, (2) Mixture model: W: Weights, M: Mixture, (3) Update when different: T: True, F: False, (4) η update: static, dynamic. W mixture model use only the current weights while selecting the model. Mixture weights consider instance scores as well and are obtained by: $w_i[n] = w_i[n]$ $\texttt{RegScore}_i[n]$, for instance n, model i, which is equivalent to $\mathbf{w}[n] = \mathbf{w}[n] \cdot \texttt{RegScore}[n]$ for $\cdot$ denoting the dot product and $\texttt{RegScore}[n]$ the vector of RegMT scores of instance n. We found the variance for the stochastic model choice to be ≤ 0.006 for 100 repetitions. Therefore, we do not list the variance in the results.

Baseline performance obtained with random selection has 0.1407 BLEU score on the *en-de* development set. RegMT model score itself

BLEU Setting				Add.	BIC	Multiplicative BICW	Mul.	Winnow
S	M	T	static	.1421	.1433	.1579	.1535	.1525
			dynamic	.1422	.1431	.1573	.1550	.1523
		F	static	.1421	.1432	.1569	.1529	.1525
			dynamic	.1423	.1433	.1576	.1544	.1523
	W	T	static	.1419	.1439	.1563	.1523	.1526
			dynamic	.1426	.1440	.1576	.1526	.1522
		F	static	.1417	.1439	.1562	.1526	.1524
			dynamic	.1430	.1440	.1560	.1526	.1526
D	M	T	static	.1642	.1529	.1531	.1535	.1522
			dynamic	.1641	.1530	.1531	.1520	.1528
		F	static	.1643	.1530	.1507	.1536	.1521
			dynamic	.1643	.1530	.1507	.1519	.1526
	W	T	static	.1643	.1638	.1646	.1638	.1522
			dynamic	.1643	.1638	.1646	.1647	.1525
		F	static	.1644	.1638	.1646	.1638	.1522
			dynamic	.1644	.1638	.1646	.1647	.1524

Table 7.1: Simulated online learning BLEU score performances of the algorithms on the *en-de* development set over 100 repetitions. Setting has four parts: (1) Model selection: S: Stochastic or D: Deterministic, (2) Mixture model: W: Weights, M: Mixture, (3) Update when different: T: True, F: False, (4) η update: static, dynamic.

obtains a performance of 0.1661 BLEU with reranking. The best model performance among the 12 *en-de* translation models has 0.1644 BLEU score. Therefore, by just using the RegMT score, we are able to achieve better scores. Deterministic model selection results in Table 7.1 almost always achieve the performance of the best translation model.

Not all of the settings are meaningful. For instance, stochastic model selection is used for algorithms having multiplicative weight updates. This is reflected in the Table 7.1 by the low performance on the additive and BIC models. Similarly, using mixture weights may not result in better scores for algorithms with multiplicative updates, which resulted in decreased performance in Table 7.1. Also, $\eta = 0.5$ for BIC (Equation 7.7) and BIC with weighting (Equation 7.8), therefore η update has no effect. Decreased performance for BIC with deterministic model selection hints that we may use other techniques for mixture weights.

We do not observe a significant effect of η selection except for mul-

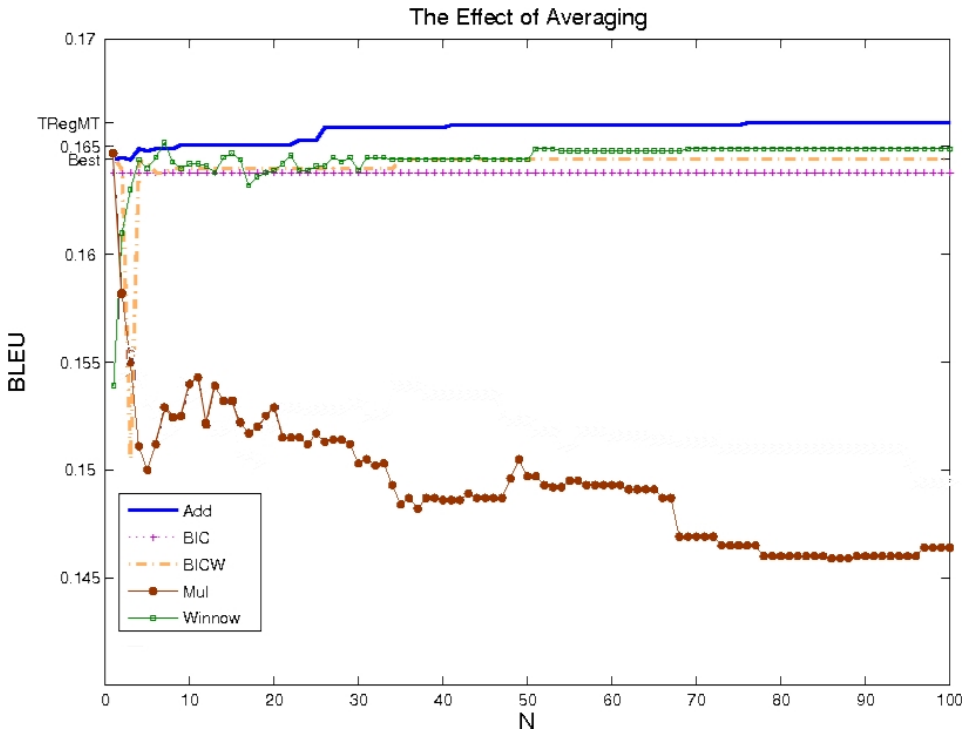

Figure 7.1: BLEU performance for different selection methods using *averaging* for increasing N.

tiplicative updates where dynamic selection performs slightly better. Therefore, we choose dynamic η on the test sets. We also do not observe significant difference between update when different selections (**not** `Update.when.correct`) and we choose to update at each instance. This is different than how Winnow algorithm performs updates. We consider only the deterministic model selection for additive and loss based weight updates. We select two best settings for each algorithm to be used in obtaining test results.

Averaging: We present averaging results for each learning algorithm in Figure 7.1. For multiplicative updates, averaging with N decreases the performance. In general, for other algorithms, averaging improves the performance. Winnow algorithm achieves significant gains with a small sized window.

BLEU	*en-de*	*de-en*	*en-fr*	*en-es*	*en-cz*
Random	.1490	.2088	.2415	.2648	.1283
Best model	.1658	.2408	.2864	.3047	.1576
RegMT	.1689	.2357	.2947	.3049	.1657
Add-D-M-F	**.1697**	**.2353**	**.2948**	**.3044**	**.1641**
Add-D-W-F	**.1697**	**.2354**	**.2948**	**.3044**	**.1642**
BIC-D-M-F	.1580	.2141	.2791	.2876	**.1577**
BIC-D-W-F	.1613	**.2407**	**.2841**	.2785	**.1621**
BICW-D-W-T	**.1647**	**.2357**	.2807	.2785	.1511
BICW-S-W-T	.1621	.2247	.2796	.2882	**.1568**
Mul-D-W-F	.1590	.2267	.2797	.2842	**.1613**
Mul-S-W-F	.1568	.2094	.2810	.2920	**.1611**
Winnow-D-W-T	.1567	.2139	.2792	.2879	**.1576**
Winnow-S-W-T	.1567	.2140	.2792	.2877	**.1576**
Challenge	.1567	**.2394**	.2758	**.3047**	**.1641**

Table 7.2: CSMT BLEU results with *simulated online learning* where **bold** corresponds to scores better than or close to the best model. Underlined scores are better than both the RegMT model and the best model.

7.5 Test Results

This section presents the results we obtained on the test sets.

7.5.1 Simulated Online Learning Results

Table 7.2 presents reranking results on all of the language pairs we considered with the random, RegMT, and CSMT models. Random model score lists the random model performance selected among the competing translations randomly and it can be used as a baseline. Best model score lists the performance of the best model performance. CSMT models are named with the weighting model used, model selection technique, mixtures model, and update when different with hyphens in between. We submitted part of the initial simulated online results to the WMT'10 system combination challenge (Callison-Burch et al., 2010).

We have presented scores that are better than or close to the best model in **bold**. We observe that the additive model performs the best by achieving the performance of the best competing translation model and performing better than the best in most of the language pairs.

BLEU	*en-de*	*de-en*	*en-fr*	*en-es*	*en-cz*
Random	.1490	.2088	.2415	.2648	.1283
Best model	.1658	.2408	.2864	.3047	.1576
RegMT	.1689	.2357	.2947	.3049	.1657
Add-D-M-F	**.1689**	**.2360**	**.2948**	**.3049**	**.1657**
Add-D-W-F	**.1689**	**.2360**	**.2950**	**.3049**	**.1657**
BIC-D-M-F	**.1636**	.2141	.2790	.2876	**.1578**
BIC-D-W-F	.1620	**.2374**	.2340	.2918	**.1574**
BICW-D-W-T	.1497	**.2419**	.2807	.2785	.1405
BICW-S-W-T	.1461	.2121	.2541	.2677	.1282
Mul-D-W-F	.1611	**.2373**	.2360	.2918	**.1574**
Mul-S-W-F	.1585	.2054	.2543	.2729	.1271
Winnow-D-W-T	.1595	.2142	.2792	.2877	**.1576**
Winnow-S-W-T	.1592	.2143	.2792	.2877	**.1576**
Challenge	.1567	**.2394**	.2758	**.3047**	**.1641**

Table 7.3: CSMT BLEU results with *online learning* where **bold** corresponds to scores better than or close to the best model. Underlined scores are better than both the RegMT model and the best model.

For the *en-de* language pair, additive model score achieves even better than the RegMT model, which is used for evaluating instance scores.

Another observation is that with the adaptive weighting model, we can achieve scores better than the best model and the RegMT model. In cases where the RegMT model score is not better than the best model, CSMT is able to achieve the performance of the best model as in the *de-en* system.

7.5.2 Online Learning Results

We also experiment with the online learning setting where reference translations are used to estimate instance losses. We still use RegMT scores for mixture weights and the initial weight update. The results are given in Table 7.3. In general, we see some increase in the scores with the online learning setup. We did not in general see significant differences between the results obtained with the simulated online learning and online learning. This may indicate that RegMT scores can effectively select the best model. Also, the performance differences between the oracle best model and the best model according to the RegMT score may be low. Since we measure the performance with

sentence level BLEU scores, rather than binary losses, the differences may be negligible.

7.6 Contributions

We have analyzed adaptive model weighting techniques for system combination when the competing translators are statistical machine translation models. We defined additive, multiplicative, and loss-based weight updates with exponential loss functions for the competitive statistical machine translation learning framework.

We applied these weight update models in simulated online learning and online learning settings. We observe that with the adaptive weighting model, we can achieve scores better than the best model and the RegMT model. In cases where the RegMT model score is not better than the best model, CSMT is able to achieve the performance of the best model. By following the combination techniques we pursued, we may achieve performances with worst case guarantees such as the performance of the best among the competing models.

Competitive SMT via adaptive weighting of various translation models is shown to be a powerful technique for sequential translation tasks. We have demonstrated its use in the system combination task by using the instance scores obtained by the RegMT model. Without any pre-knowledge of the performance of the translation models, we have been able to achieve the performance of the best model in all translation experiments and we are able to surpass its performance as well as RegMT's performance with some of the weight update models we considered.

Chapter 8

Prediction, Evaluation, and Decoding with RegMT

In this chapter, we use the RegMT model as a stand alone machine for translation. We present our target feature prediction results in comparison with other learning techniques, evaluation metrics that we develop and use, and decoding results with the RegMT outputs including graph decoding and decoding with Moses.

We show that L_1 regularized regression performs better than L_2 regularized regression and other learning models we compared when predicting target language features, estimating word alignments, creating phrase tables, and generating translation outputs.

We discuss the F_1 measure, which performs good when evaluating translations into English according to human judgments. Weighted F_1 measure allows us to evaluate the performance of the RegMT model using the target feature prediction vectors or the coefficients matrices learned. We can also evaluate a given SMT model's performance using its phrase table without performing the decoding step.

We present encouraging results when translating from German to English (*de-en*) and Spanish to English (*es-en*) using graph decoding. We demonstrate that sparse L_1 regularized regression performs better than L_2 regularized regression in the *de-en* translation task sampled from the WMT'10 datasets and in the *es-en* translation task when using small sized training sets. Graph based decoding can provide an

alternative to phrase-based decoding systems in translation domains having low vocabulary size.

Outline: We describe the datasets we use in the next section. In section 8.2, we introduce the F_1 measure, which correlates with human judgments better than BLEU for evaluating both the predictions and the phrase table. We compare performances of various learning models with *lasso* and find that *lasso* achieves better performance. We also give results about using F_1 as an evaluation metric to measure the translation quality. In section 8.3, we show how we can convert the learned coefficients matrix, $\mathbf{W}$, to a phrase table that can be used by the Moses decoder to generate translations. Later in section 8.4, we develop an evaluation metric to determine the quality of a given phrase table for comparing phrase table quality. section 8.5 demonstrates that weighted F_1 score correlates better with the BLEU scores obtained than a phrase table evaluation measure that we develop and therefore it is a better measure for evaluating translation performance. We give evaluation results with weighted F_1 in section 8.6 where we compare various learning algorithms. In section 8.7, we present word alignment results comparing the learning algorithms and in section 8.8 we give decoding results using both graph decoding and the Moses decoder. The last section lists our contributions.

8.1 Datasets

We use the German-English (*de-en*) parallel training sentences of size about 1.6 million sentences from WMT'10 (Callison-Burch et al., 2010) to select training and in-domain test instances. For development and test sentences, we select randomly among the sentences whose length are in the range $[10, 20]$ and target language bigram coverage in the range $[0.6, 1]$. We select 20 random instances from each target coverage decimal fraction (i.e. 20 from $[0.5, 0.6]$, 20 from $[0.6, 0.7]$, etc.) to obtain sets of 100 sentences. Sampling from training instances having high target coverage leads to the creation of in-domain datasets and

selecting instances belonging to varying coverage ranges allows us to measure the performance on sentences with different translation difficulties. We create in-domain *dev*, *dev2*, and *test* sets following this procedure and make sure that test set source sentences do not have exact matches in the training set. We use n-spectrum weighted word kernel (Equation 3.11) as feature mappers which considers all n-grams up to order n. Features used are 1-grams, 2-grams, or 1&2-grams.

Our selection of in-domain development and test sets enable us to evaluate the performance of different learning algorithms better with less concerns about whether a given translation exists in the training set or not. This choice does not limit the applicability of the RegMT system to only in-domain test sets.

8.2 F_1 Measure for Evaluating the Estimates and the Phrase Table

F_1 is a commonly used information retrieval measure defined as the harmonic mean of *precision* and *recall* values. The following table gives the types of errors for binary classification:

	Prediction	
Truth	Class 0	Class 1
Class 0	TN	FP
Class 1	FN	TP

Let TP be the true positive, TN the true negative, FP the false positive, and FN the false negative rates, then the measures are defined as:

$$\text{prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{BER} = \left(\frac{\text{FP}}{\text{TN} + \text{FP}} + \frac{\text{FN}}{\text{TP} + \text{FN}}\right)/2 \tag{8.1}$$

$$\text{rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad F_\beta = (1 + \beta^2)\frac{\text{prec} \times \text{rec}}{\beta^2 \text{prec} + \text{rec}} \tag{8.2}$$

where BER is the balanced error rate, prec is precision, and rec is recall. For $\beta = 1$, *precision* and *recall* values are equally weighted in F_β and for $\beta > 1$ *recall* is weighted more than *precision*.

$\phi_Y(\mathbf{y})$	1	1	2	1	1	1	1	1	2	1	1	1			
$\hat{\phi_Y}(\mathbf{y})$					0.9	1.2	0.4	0.1	1.6	0.3	0.2	1.1	0.8	1.5	0.3	0.4
$\sigma(\hat{\phi_Y}(\mathbf{y}))$					1	1	1	0	1	0	0	1	1	1	0	1
Δ	a				b								c			

$$\text{prec} = \frac{b}{b+c} \qquad \text{rec} = \frac{b}{a+b} \tag{8.3}$$

Table 8.1: Precision and recall diagram shows the feature vectors for the reference, $\phi_Y(\mathbf{y})$, its prediction, $\hat{\phi_Y}(\mathbf{y})$, the thresholded prediction, $\sigma(\hat{\phi_Y}(\mathbf{y}))$, the element-wise absolute difference, $\Delta = |\phi_Y(\mathbf{y}) - \sigma(\hat{\phi_Y}(\mathbf{y}))|$, and the corresponding precision and recall calculation over Δ.

Given a feature prediction vector, we can use a threshold for determining which features can be considered significant to exist in a given prediction. After classifying the features in the two categories, 0 (absent) or 1 (present), we can use binary classification tests to evaluate the performance. The evaluation techniques measure the effectiveness of the learning models on distributing the weights and in identifying the features of the target sentence while making minimal error and increasing the performance of the decoder and its translation quality. BER has the problem that if the estimate is only zeros, then BER becomes 0.5; however, F_1 is 0 in this case, which is more acceptable. If a learning model optimizes its parameters with BER, there is a chance that it will prefer an empty output than making some error and correcting it later. Thus, it may get stuck in the pit of BER.

We can determine whether a feature is present in the estimation by using the 0/1-class predictions obtained after thresholding $\Phi_Y(\hat{\mathbf{y}})$, the target features prediction vector. The thresholds are used to map real feature values to 0/1-class predictions and they are also optimized using the F_1 measure on *dev*. We obtain optimized feature thresholds in subsection 8.6.1. Table 8.1 depicts the calculation of precision and recall on sample reference and prediction feature vectors. The

top row represents the reference sentence in feature counts vector and the second row corresponds to the feature prediction vector obtained. The third row corresponds to the transformed prediction vector after thresholding with a thresholding function $\sigma(.)$. The element-wise absolute difference vector $\Delta = |\phi_Y(\mathbf{y}) - \sigma(\hat{\phi_Y}(\mathbf{y}))|$ is given in the last row. The first part of the prediction corresponds to features that are not covered in the training set and its sum is represented with a in the Δ row. The second part is the covered features with cost b and the third part is the extra feature predictions over the features in the training set that are neither covered nor observed in the reference sentence with cost c.

We can also incorporate the weights in the prediction to calculate weighted binary classification performance and weighted *precision* and *recall* values to be used for measuring the weighted F_1 and weighted BER scores. After thresholding, instead of adding a 1 to the binary classification test errors, we increase them by the weight obtained. We find that weighted F_1 score correlates well with the BLEU scores obtained (section 8.5) and therefore it is a good measure for evaluating translation performance, which also considers the estimate weights. We refer to the weighted F_1 measure when we use F_1. The model parameters such as the regularization λ and the number of iterations for FSR are optimized on *dev* using the weighted F_1 measure. Slightly better results can also be obtained by optimizing F_1's β against the sentence level correlation with BLEU.

8.2.1 Target F_1 as a Performance Evaluation Metric

We use target sentence F_1 measure over the target features as a translation performance evaluation metric. We use gapped word sequence kernels (Taylor and Cristianini, 2004) when using F_1 for evaluating translations since a given translation system may not be able to translate a given word but can correctly identify the surrounding phrase.

		Format	BLEU	F_1		
Ref:	`a sound compromise has been reached`	a b c d e f	4-grams	3-grams	4-grams	5-grams
Trans_1:	`a sound agreement has been reached`	a b x d e f	.2427	.6111	.5417	.5
Trans_2:	`a compromise has reached`	a c d f	.137	.44	.3492	.3188
Trans_3:	`a sound agreement is reached`	a b x y f	.1029	.2	.1558	.1429
Trans_4:	`a compromise is reached`	a c y f	.0758	.2	.1587	.1449
Trans_5:	`a good compromise is reached`	a z c y f	.0579	.1667	.1299	.119
Trans_6:	`a good compromise is been`	a z c y e	.0579	.2	.1558	.1429

Table 8.2: BLEU vs. F_1 on sample sentence translation task.

For instance, let the reference translation be the following sentence:

`a sound compromise has been reached`

Some possible translations for the reference are given in Table 8.2 together with their BLEU (Papineni et al., 2001) and F_1 scores for comparison. F_1 score does not have a brevity penalty but a brief translation is penalized by a low recall value. We use up to 3 tokens as gaps. F_1 measure is able to increase the ranking of Trans_4 by using a gapped sequence kernel, which can be preferable to Trans_3.

We note that a missing token results in an increasing loss in the n-gram precision used in the BLEU score proportional to n. A sentence containing m tokens has m 1-grams, $m-1$ 2-grams, and $m-n+1$ n-grams. A missing token degrades the performance more in higher order n-gram precision values. A missing token decreases 1-gram precision by $\frac{1}{m}$ and by $\frac{n}{m-n+1}$ for n-grams. Based on this observation, we use F_1 measure with gapped word sequence kernels to evaluate translations. Gapped features allows us to consider the surrounding phrase for a missing token as present in the translation.

Let the reference sentence be represented with `a b c d e f` where a-f, x, y, z correspond to tokens in the sentences. Then, Trans_3 has the form `a b x y f`, and Trans_4 has the form `a c y f`. Then, F_1 ranks Trans_4 higher than Trans_3 for orders greater than 3 as there are two consecutive word errors in Trans_3. F_1 can also prefer a missing token rather than a word error as we see by comparing Trans_4 and Trans_5 and it can still prefer contiguity over a gapped sequence as we see by

comparing Trans_5 and Trans_6 in Table 8.2.

We calculate the correlation of F_1 with BLEU on the *en-de* development set. We use 5-grams with the F_1 measure as this increases the correlation with 4-gram BLEU. Table 8.3 gives the correlation results using both Pearson's correlation score and Spearman's correlation score. Spearman's correlation score is a better metric for comparing the relative orderings.

Metric	No gaps (F_1)	Gaps (F_1^{g3})
Pearson	.8793	.7879
Spearman	.9068	.8144

Table 8.3: F_1 correlation with 4-gram BLEU using blended 5-gram gapped word sequence features on the development set.

We refer to F_1 measure with features using up to 3 tokens as gaps as F_1^{g3}, which we compare in Table 8.3. Although F_1^{g3} has lower correlation scores with BLEU compared to F_1, we prefer to use F_1^{g3} as an evaluation metric due to its better discriminative power as exemplified in Table 8.3 with different translations. We have submitted the results obtained using both of the measures to the WMT'11 automatic evaluation metrics challenge, which we discuss in the next section.

8.2.2 WMT'11 Automatic Evaluation Metrics Challenge Results

We participated in the WMT'11 automatic evaluation metrics challenge, which examines automatic evaluation metrics and calculates the correlation of their rankings with human judgments. We participated in the challenge with the F_1 measure with up to 5-gram features (F15) without gapped features and with the F_1 measure with up to 5-gram features with up to 3 tokens as gaps (F15g3). System-level correlation comparison results are given in (Callison-Burch et al., 2011, Table 12) for translations out of English and (Callison-Burch et al., 2011, Table 13) lists translations into English.

We observe that in Table 13 both F15 and F15g3 perform better than BLEU in evaluating the overall translation quality of different

translation systems. However, we observe lower performance in Table 12. The lower performance compared to BLEU for translations out of English given in (Callison-Burch et al., 2011, Table 12) is likely to be due to the increased reorderings involved in the target languages. For a highly non-monotonic language, some gapped features may become less meaningful, making those features less useful in the evaluation or even degrading the performance. This is observed for the *en-cz* and the *en-de* language pairs in Table 12. We also observe better performance for the *en-es* direction, where Spanish is considered to be a more monotonic language than German. When evaluating non-monotonic languages, we can optimize the maximum size of the gap accordingly to improve the performance. With no gaps involved, the features used become the n-gram features used in BLEU.

Overall, the good performance of F_1 measure in the evaluation task demonstrates that we can also use F_1 to evaluate the quality of translations using their feature representations without actually performing decoding (see section 8.5). We can estimate the set of target features using the RegMT system and use F_1 measure to evaluate the performance. F_1 measure can be a good alternative to BLEU when working with machine learning algorithms optimizing the prediction of target feature vectors.

8.3 Phrase Table Generation from W and Moses Integration

We can obtain a phrase table (PT) such as the one Moses is using from a given $\mathbf{W}$, the coefficients matrix. After performing this conversion, we can use Moses decoder for translation, which has built in reordering features, lexical weights, and phrase translation probabilities in both translation directions. We interpret the coefficients matrix $\mathbf{W}$ as the phrase table, which enables us to perform experiments with Moses.

In our transductive learning framework, a $\mathbf{W}$ matrix is created for each test sentence, which replaces the phrase table. Moses loads the phrase table once in the beginning and uses it to translate test sen-

tences. We modify Moses such that it reloads the phrase table before translating each test sentence.

Moses is using 5 scores in the generated phrase table:

- $\varphi(\mathfrak{f}_x|\mathfrak{f}_y)$: inverse phrase translation probability.
- $\text{lex}(\mathfrak{f}_x|\mathfrak{f}_y)$: inverse lexical weighting.
- $\varphi(\mathfrak{f}_y|\mathfrak{f}_x)$: direct phrase translation probability.
- $\text{lex}(\mathfrak{f}_y|\mathfrak{f}_x)$: direct lexical weighting.
- phrase penalty: $e = 2.718$

for source phrase $\mathfrak{f}_x$ and target phrase $\mathfrak{f}_y$. Phrase translation probabilities are obtained using the phrase counts as follows:

$$\varphi(\mathfrak{f}_x|\mathfrak{f}_y) = \frac{\text{count}(\mathfrak{f}_x, \mathfrak{f}_y)}{\text{count}(\mathfrak{f}_y)} \quad , \quad \varphi(\mathfrak{f}_y|\mathfrak{f}_x) = \frac{\text{count}(\mathfrak{f}_x, \mathfrak{f}_y)}{\text{count}(\mathfrak{f}_x)} \tag{8.4}$$

Moses obtains phrase counts after a word alignment step applied on the training data using GIZA++ (Och and Ney, 2003). Lexical weights, $\text{lex}(\mathfrak{f}_x|\mathfrak{f}_y)$ and $\text{lex}(\mathfrak{f}_y|\mathfrak{f}_x)$, are obtained using word translation probabilities $w(y|x)$ and $w(x|y)$ for source word x and target word y and estimated using the word alignments (Koehn et al., 2003):

$$w(y|x) = \frac{\text{count}(x, y)}{\sum_{y'} \text{count}(x, y')} \tag{8.5}$$

$$\text{lex}(\mathfrak{f}_x|\mathfrak{f}_y, \mathbf{a}) = \prod_{i=1}^{n} \frac{1}{|\{j|(i,j) \in \mathbf{a}\}|} \sum_{\forall(i,j)\in\mathbf{a}} w(x_i|y_j) \tag{8.6}$$

$$\text{lex}(\mathfrak{f}_x|\mathfrak{f}_y) = \max_{\mathbf{a}} \text{lex}(\mathfrak{f}_x|\mathfrak{f}_y, \mathbf{a}) \tag{8.7}$$

where $\mathbf{a}$ stores the word alignments found in $\mathfrak{f}_x$ and $\mathfrak{f}_y$ and i stores the source word positions from 1 to n. In case multiple alignments exist maximum lexical weight is chosen as the lexical weight.

We obtain direct and indirect translation probabilities after retaining the top N scoring target phrase entries for each source phrase. Moses use a default translation table limit (`--ttable-limit`) of 20

entries per source phrase, which limits the number of translation options considered for each source phrase. Accordingly, for each source feature, we choose the top $N = 20$ target features to be in line with the translation table limit of Moses. We use $\mathfrak{f}_x$ and $\mathfrak{f}_y$ to refer to a source phrase and a target phrase respectively. We obtain the scores for a given source test sentence $\mathbf{x}$ as follows [1]:

$$p(\mathfrak{f}_y|\mathfrak{f}_x) = \frac{\mathbf{W}[\mathfrak{f}_y, \mathfrak{f}_x]}{\sum_{\mathfrak{f}_y' \in \text{top}_N(\mathbf{W}, F_Y, \mathfrak{f}_x)} \mathbf{W}[\mathfrak{f}_y', \mathfrak{f}_x]}, \tag{8.8}$$

$$p(\mathfrak{f}_x|\mathfrak{f}_y) = \frac{\mathbf{W}[\mathfrak{f}_y, \mathfrak{f}_x]}{\sum_{\mathfrak{f}_x' \in \Phi_X(\mathbf{x})} \mathbf{W}[\mathfrak{f}_y, \mathfrak{f}_x']} \tag{8.9}$$

where $\Phi_X(\mathbf{x})$ return the source features of source test sentence $\mathbf{x}$, and $\text{top}_N(\mathbf{W}, F_Y, \mathfrak{f}_x)$ return the top N target features found in $\mathbf{W}$ for given $\mathfrak{f}_x$. $\mathfrak{f}_x$ are chosen from the feature set of the source test sentence and $\mathfrak{f}_y$ from where $\mathbf{W}[\mathfrak{f}_y, \mathfrak{f}_x] > 0$. $\mathbf{W}[\mathfrak{f}_y, \mathfrak{f}_x]$ corresponds to the regression coefficient mapping $\mathfrak{f}_x$ to $\mathfrak{f}_y$. This choice of summing over source features found in the source test sentence helps us discriminate among target feature alternatives better than summing over all source features found in the training set. We also find that by summing over all of the positive entries for features selected from F_Y rather than over the selected top N aligned target features, we increase the precision in the phrase translation scores by increasing the denominator for frequently observed phrases. We then renormalize using the top N target feature entries.

After this conversion, we obtain 3 scores for each phrase table entry: $p(\mathfrak{f}_x|\mathfrak{f}_y)$, $p(\mathfrak{f}_y|\mathfrak{f}_x)$, 2.718, where the last score is used for the phrase penalty. To compare the performance with Moses, we use the option `-score-options '--NoLex'` during training, which removes the scores coming from lexical weights in the phrase table entries, leaving 3 scores similar to the scores used in the RegMT phrase table. Instead

[1] $p(\mathfrak{f}_y|\mathfrak{f}_x)$ is the direct phrase translation probability and corresponds to $p(e|f)$ in Moses. Similarly, $p(\mathfrak{f}_x|\mathfrak{f}_y)$ is the inverse phrase translation probability and corresponds to $p(f|e)$ in Moses.

of adding extra lexical weights to the RegMT phrase tables, we remove them to measure the difference between the phrase tables better.

8.4 Phrase Table Quality Measure: $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$

In this section, we develop a phrase table quality evaluation measure to be able to compare RegMT phrase tables with Moses phrase tables. Phrase table is a very important translation model used during decoding and provides the vocabulary of translation. Our goal is to be able to evaluate the quality of a phrase table and maybe estimate the BLEU performance achievable before actually performing the computationally demanding process of decoding.

We have mentioned in subsection 4.3.1 that $\mathbf{W}^{\max}_{*,j} = \max(|\mathbf{W}_{1,j}|, |\mathbf{W}_{2,j}|, \ldots, |\mathbf{W}_{N_Y,j}|)$ can be used as a measure of the explanatory power of the j^{th} regressor on all the response variables. Similarly, we use $\mathbf{W}^{\max}_{i,*} = \max(\mathbf{W}_{i,1}, \mathbf{W}_{i,2}, \ldots, \mathbf{W}_{i,N_X})$ as the maximum achievable explanation available for the i^{th} response variable. When we model translation features, $\mathbf{W}_{i,j}$ correspond to the explanatory power that the j^{th} source feature, $\mathfrak{f}_{x_j}$, has on the i^{th} target feature, $\mathfrak{f}_{y_i}$. If we have access to the target features, we can use $\mathbf{W}^{\max}_{i,*}$ to obtain a quality metric that measures how well target features are explained by $\mathbf{W}$.

In the transductive learning setting, we do not need to consider all of the source features $\mathfrak{f}_{x_j}$ for $j \in \{1, \ldots, N_X\}$ and consider only the ones that appear in the source sentence. After obtaining a phrase table, PT, from $\mathbf{W}$ following section 8.3, we have scores for each source feature mapping to a number of target features that they translate to with their $p(\mathfrak{f}_x|\mathfrak{f}_y)$ and $p(\mathfrak{f}_y|\mathfrak{f}_x)$ values. We define $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$ for a source sentence $\mathbf{x}$ as the maximum phrase translation probability for $\mathfrak{f}_y$ achievable given the alternative source features:

$$p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x})) = \max_{\mathfrak{f}_x \in \Phi_X(\mathbf{x})} p(\mathfrak{f}_y|\mathfrak{f}_x). \tag{8.10}$$

$p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$ makes sense when we are interested in distribut-

ing the probability of the source feature to few target translations that are correct and it also makes sense when we interpret translation as a classification problem where we are only interested in the correct label. We will visit the interpretation of translation as classification in subsection 8.6.2. Although it favors *precision* rather than *recall*, $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$ is helpful in determining the quality of the phrase translation probability estimates in a given PT. We are interested in distributing the probability to only the correct target features.

We define the sum $p^{\Sigma}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$ given the translation, $\mathbf{y}$, as follows:

$$p^{\Sigma}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x})) = \sum_{\mathfrak{f}_y \in \Phi_Y(\mathbf{y})} p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x})) = \sum_{\mathfrak{f}_y \in \Phi_Y(\mathbf{y})} \max_{\mathfrak{f}_x \in \Phi_X(\mathbf{x})} p(\mathfrak{f}_y|\mathfrak{f}_x). \tag{8.11}$$

Using $p^{\Sigma}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$, we identify how well a given PT performs in explaining the features of a target sentence, $\mathbf{y}$, starting from the features of the source sentence, $\mathbf{x}$.

We define the mean value, $p^{\mu}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$, as follows:

$$p^{\mu}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x})) = p^{\Sigma}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x})) \;/\; |\Phi_Y(\mathbf{y})|. \tag{8.12}$$

In the presence of correlated variables, $p^{\max}(\mathfrak{f}_y \mid \Phi_X(\mathbf{x}))$ may be distributed among multiple variables correlated with $\mathfrak{f}_y$. In such cases, as we normalize such that $\sum_{\mathfrak{f}_{y_i} \in \Phi_Y(\mathbf{y})} p(\mathfrak{f}_{y_i}|\mathfrak{f}_x) = 1$, $p^{\max}(\mathfrak{f}_y \mid \Phi_X(\mathbf{x}))$ may be lower than expected.

For a given test set of sentences, $T = (\mathbf{X}, \mathbf{Y})$, we define:

$$p^{\mu}(T) = \frac{\sum_{(\mathbf{x},\mathbf{y}) \in T} p^{\Sigma}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x})))}{|\bigcup_{\mathbf{y} \in T} \Phi_Y(\mathbf{y})|}. \tag{8.13}$$

for the mean value of a test set.

8.4.1 Optimization with $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$

A sharp distribution with only a single translation for a given source feature $\mathfrak{f}_x$ corresponds to having $p(\mathfrak{f}_x|\mathfrak{f}_y) = 1$. This makes the inverse phrase translation probability, $p(\mathfrak{f}_x|\mathfrak{f}_y)$, useless when discriminating among features. A sharp distribution is useful with respect to estimating the target features with few estimates however this reduces recall. We can use $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$ to measure the performance of a phrase table created from a distribution regardless of it is made sharper by modifying its parameters. We optimize the λ using $p^{\max}(\mathfrak{f}_y|\Phi_X(\mathbf{x}))$ to obtain phrase tables having sharper distributions and fewer phrase table entries.

8.5 An Evaluation Metric for Translation Quality Before Obtaining the Translations

We are interested in obtaining an evaluation metric for machine translation before actually performing decoding. The question we ask is given as follows:

> *Can we estimate the performance of a machine translation model without going through the computationally complex process of decoding?*

The answer is YES with the RegMT model, to some degree. We can estimate the performance using the two measures that we defined:

- by looking at either the phrase table performance as measured by $p^{\mu}(T)$ relative to the $p^{\mu}(T)$ value of the phrase table of a known BLEU result, or
- by looking at the target sentence estimates obtained from the learning model or the derived phrase table with the F_1 measure and comparing with a known result.

We compare the performance of both $p^{\mu}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$ and F_1 measures with respect to their correlation with the sentence level BLEU

	Corr with BLEU Sent		Scores		
Model	$p^{\mu}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$	F_1	$p^{\mu}(T)$	F_1	**BLEU**
Moses	0.5291	0.5102	0.3440	0.51	0.3021
L_2	0.5186	0.5742	0.1020	0.36	0.2547
FSR	0.5718	0.6112	0.2616	0.47	0.2511
QP	0.5594	0.5963	0.2976	0.47	0.2296
logreg	0.5303	0.5488	0.2928	0.44	0.1900

Table 8.4: Phrase table performance scores and their Pearson's correlation coefficient score with sentence level BLEU scores of the translations obtained. F_1 is obtained by using the weighted precision and recall scores.

scores. Table 8.4 shows that a good performance in these measures is highly correlated with sentence level BLEU performance but each model is correlated with a different correlation score. We obtain the BLEU results for the learning models using Moses after replacing the phrase table with the phrase table obtained from $\mathbf{W}$ following section 8.3. We use weighted *precision* and *recall* values to obtain weighted F_1 score where the weight is the estimation itself. Note that we use the probability values for the *logreg* model and Moses and for others we use the estimate values themselves as the weights after bounding their values in the range $[0, 1]$.

We observe that both of these measures provide good estimations of the BLEU performance at the sentence level as can be seen from the second and third columns of Table 8.4. F_1 correlates more with BLEU than $p^{\mu}(\Phi_Y(\mathbf{y}) \mid \Phi_X(\mathbf{x}))$ at the sentence level, reaching a correlation of 0.61 using the translations obtained with FSR. When we look at the fourth and fifth columns, we observe that the $p^{\mu}(T)$ score obtained is less correlated with the BLEU scores obtained (corr=0.0588) than F_1 (corr=0.3186) at the model level. Therefore, F_1 achieves better correlation with BLEU than $p^{\mu}(T)$ and we can use it, to some degree, to evaluate the translation quality before actually performing decoding. We can also obtain a translation and optimize on weighted F_1 to achieve a better translation before the decoding step.

When we compare the performance of sparse regression results with L_2 regularized regression in Table 8.4, we observe that sparse regression models achieve higher F_1 score but close or lower BLEU score. When

Model	# entries	# words	# words per entry
Moses	23,728	68,684	2.89
L_2	52,780	162,599	3.08
FSR	28,979	80,342	2.77
QP	52,780	160,190	3.04
logreg	29,078	86,805	2.98

Table 8.5: Phrase table comparison for various learning models by the number of entries, the total number of words, and the average number of words in each entry.

creating the translation, other knowledge sources such as the language model are also incorporated and having more target phrase entries for each source phrase in the phrase table can compensate for lower weighted F_1 score and result with similar BLEU performance. We cannot claim that this is due to L_2 having higher weighted recall as we see in Table 8.8 where FSR-*lasso* performs better than L_2 in all measures used.

We observe that using a phrase-based decoder like Moses, with having more entries in the phrase table and caring less about the accuracy of the phrase translation probabilities present in the phrase table, we can still achieve similar BLEU with a model having higher weighted F_1 score. If we have a decoder that has a performance close to optimal, then recall can be a more important factor even though the phrase translation probabilities of the entries are very low. With an optimal decoder, recall is the important, which determines the vocabulary. Therefore, having more entries in the phrase table can compensate for worse prediction performance as we see with L_2 regularized solution than having more accurate phrase translations as with the *lasso* solution.

In Figure 8.1 we observe similar performance overall for *lasso* and L_2 and small improvement over L_2 using *lasso* for small training set sizes. In subsection 8.8.2, we show that with graph decoding on the target feature prediction vectors, sparse regression can achieve better performance than L_2 regularized regression.

We compare the statistics of the phrase tables obtained in Table 8.5. We give the number of entries, the total number of words, and the av-

erage number of words in each entry of the phrase tables. The number of words per entry is the sum of the number of words in the source and the target phrase of each entry. We observe that FSR's phrase tables are the closest to Moses' phrase tables among the different learning models' obtained phrase tables.

8.6 F_1 Experiments

This section presents our experimental results when we use F_1 as an evaluation metric when predicting target features and when evaluating the phrae table. We experiment with different regression techniques and learning models and compare their performance to FSR-*lasso*. We show that FSR is able to identify features better than the other techniques we compared while making less error. We first perform optimization on the *dev* set to find the parameters of the learning models that maximize F_1 value (subsection 8.6.1). Then we interpret target feature estimation as a classification problem in subsection 8.6.2 estimating the target features which are present rather than estimating the values of the target features as in regression. We test our learning models on the *test* set in subsection 8.6.3.

8.6.1 Optimization with F_1

We compare the performance of L_2 regularized ridge regression with L_1 regularized regression (*lasso*) as well as other regression techniques. We use FSR when we use the term *lasso*. We also compare the results we obtain with support vector regression (SVR) using *rbf* (radial basis functions) kernel and iterative thresholding (*iter*-ε) for L_1 minimization (subsection 4.3.4). As classification methods, we also use logistic regression (*logreg*) and support vector classification (SVC) to determine target features.

We perform parameter optimization for the machine learning models we use to estimate the target vector on the *dev* set. Table 8.6 lists the feature thresholds found optimizing weighted F_1 over *dev*. For

	Regression					Classification		
	L_2	FSR	QP	*iter-ε*	SVR	*logreg*	SVC	Moses
F_1 Threshold	0.2079	0.2599	0.236	0.9615	0.2195	0.0127	0.1201	0.2404

Table 8.6: Feature thresholds are optimized to maximize weighted F_1 value over *dev*. For *logreg* and SVC models, we use the probability of class 1 estimates for thresholding.

logreg and SVC models, we use the probability of class 1 estimates for thresholding (section 8.2).

We also estimate the threshold for Moses by transforming the phrase table to a prediction vector. For instance, in the individual translation setting [2], we obtain target feature vector predictions for each test sentence after transforming the phrase tables of individual Moses systems to prediction vectors by appending the target phrase entries for each source phrase one after the other. The ordering of the features is not important when measuring the F_1 score with respect to the reference translation feature vector.

We obtain results on *dev2* set using these feature thresholds. The coverage as measured by the percentage of test bigrams found in the training set is *scov* and *tcov* for source and target coverage (see section 5.3 for details). We measure the *scov* and *tcov* values for *dev2* when using 100 training instances per test sentence to be $(1.0, 0.96)$, $(0.94, 0.74)$, and $(0.97, 0.85)$ for 1-grams, 2-grams, and 1&2-grams respectively. Table 8.7 presents the results on *dev2*, listing weighted BER, precision, recall, and F_1 (see section 8.2 for definitions) when using 1-grams, 2-grams, or both. The reason for lower performance with bigrams is likely to be due to lower counts of observing them in the training set. Also, the presence of correlation between features increase with increasing order of n-grams used due to increased overlap with each other.

We compare the performance with individual Moses systems' phrase tables obtained using maximum phrase length set to 2. We can evaluate a phrase table with the weighted F_1 measure by first converting

[2] See subsection 5.4.3 for the description of individual translation models developed with Moses for each test sentence.

	BER	*Prec*	*Rec*	F_1
		1-grams		
L_2	0.41	0.55	0.43	0.48
lasso	**0.37**	**0.71**	**0.51**	**0.59**
SVR	0.35	0.52	0.41	0.46
iter-ε	0.37	0.37	0.68	0.48
logreg	0.72	0.71	0.50	0.59
SVC	0.42	0.54	0.37	0.44
Moses	0.52	0.76	0.48	0.59
		2-grams		
L_2	0.48	0.37	0.13	0.19
lasso	**0.47**	**0.64**	**0.22**	**0.32**
SVR	0.44	0.34	0.14	0.19
iter-ε	0.50	0.34	0.08	0.13
logreg	0.80	0.52	0.19	0.28
SVC	0.48	0.35	0.12	0.17
Moses	0.70	0.35	0.22	0.27
		1&2-grams		
L_2	0.43	0.55	0.29	0.38
lasso	**0.41**	**0.68**	**0.36**	**0.47**
SVR	0.39	0.53	0.25	0.34
iter-ε	0.43	0.38	0.48	0.42
logreg	0.76	0.71	0.33	0.45
SVC	0.43	0.56	0.23	0.32
Moses	0.53	0.65	0.45	0.53

Table 8.7: Comparison of learning models for the prediction of target features using weighted scores on *dev2* with 100 instances used for each test sentence using *dice* selection. Dimensions are $N_X \times N_Y \approx 846.97 \times 818.66$ for 1-grams, 1678.63×1803.85 for 2-grams, and 2594.55×2684.58 for 1/2-grams. Thresholds used are given in Table 8.6.

it to a target estimate vector. We obtain Moses target vectors from the target phrase table entries for each source test sentence feature and optimize the threshold for achieving the maximum weighted F_1 value on *dev*. This threshold is found to be 0.2404. Moses achieves $(1.0, 0.98)$, $(0.90, 0.66)$, and $(0.94, 0.92)$ coverage values for 1-grams, 2-grams, and 1&2-grams. These coverage values are higher for 1-grams and 1&2-grams, which is an indication that Moses retains only the likely entries in the phrase table. The results on *dev2* set are given in Table 8.7 separated with lines.

FSR-*lasso* is able to achieve better performance than other learning

techniques in terms of the weighted BER, precision, recall, and F_1 values. *logreg* achieves similar F_1 on 1-grams with worse BER score but performs worse on 2-grams and 1&2-grams. This may be due to the dependencies among different higher order n-grams. An evaluation with the logistic regression model shows that in the presence of many correlated features, some relevant features may get very low or negative correlations (Hastie et al., 2009, section 4.4.2). This may be a result of using an iterative procedure such as iterative reweighted least squares during the optimization of the weights (Hastie et al., 2009, section 4.4.1). As we have seen in subsection 4.3.1, setting ϵ to the magnitude of the largest current correlation value eliminates features that are correlated with the current feature. FSR_ϵ may be exempt from this error appearing in the presence of correlated variables by making $\epsilon \rightarrow 0$ changes in the coefficient values at each iteration.

Our experiments with the QP solution to *lasso* shows that QP achieves the same or close F_1 score to FSR as we expected with slightly lower precision and higher recall values. FSR achieves better performance than other learning models using 1&2-grams where large number of correlated variables exist.

We observe that Moses performs better when using 1&2-grams but performs worse with 2-grams. The decrease in the relative performance with respect to Moses may be due to the presence of correlated variables where consecutive bigrams are correlated and unigrams are correlated to the their parent bigrams. We also note that Moses is performing a word alignment step using GIZA++ (Och and Ney, 2003) in both the directions from source to target and target to source and combining these later on to obtain the phrases in the phrase table. The RegMT results we present rely on a single modeling direction from source features to target features and RegMT models the phrases found in the training set features directly.

8.6.2 Target Feature Estimation as Classification

We can also interpret the feature mapping problem as a classification problem and estimate whether a feature exists in the target (class 1) or not (class 0). We use logistic regression (*logreg*) and support vector classification (SVC) to determine the target feature for each feature of the source test sentence. We build a separate classifier for each target feature $\mathfrak{f}_y \in F_Y$ and determine if it is a 1.

When the number of 1's is much smaller than the number of 0's in the training set as we observe in our sparse learning setting, a classifier may tend to choose class 0 over class 1. In such cases, we need to introduce some bias towards choosing 1's. Thresholding approach is helpful in increasing the recall while maintaining a high precision value to optimize the F1 measure. For the *logreg* and SVC classification models, we use the probability of class 1 estimates for thresholding. The results on *dev2* are given in Table 8.7 separated by a line. Our interpretation of feature mapping as a classification problem does not give better results than regression but we achieve close results using 1-grams and *logreg*.

8.6.3 Regression Results on *test*

The regression results on *test* when using 100 training instances per test sentence is given in Table 8.8. Source and target coverage values, *scov* and *tcov*, are found to be $(1.0, 0.97)$, $(0.93, 0.76)$, and $(0.96, 0.86)$ for 1-grams, 2-grams, and 1&2-grams respectively. We observe similar performance of FSR compared to other learning models; however, Moses achieves better performance when using both 1-grams or 1&2-grams. The better performance in weighted F_1 is reflected to better BLEU results as we see in section 8.5.

	BER	*Prec*	*Rec*	F_1
		1-grams		
L_2	0.42	0.53	0.42	0.47
lasso	**0.37**	**0.71**	**0.50**	**0.59**
Moses	0.23	0.68	0.64	0.66
		2-grams		
L_2	0.49	0.34	0.12	0.17
lasso	**0.48**	**0.61**	**0.20**	**0.31**
Moses	0.45	0.29	0.31	0.30
		1/2-grams		
L_2	0.44	0.51	0.28	0.36
lasso	**0.41**	**0.68**	**0.35**	**0.47**
Moses	0.29	0.53	0.49	0.51

Table 8.8: Weighted F_1 results on *test* with 100 training instances used for each test sentence using *dice* selection. Dimensions are $N_X \times N_Y \approx 867.39 \times 836.69$ for 1-grams, 1833.67×1949.94 for 2-grams, and 2657.03×2744.96 for 1&2-grams.

8.7 RegMT for Word Alignment

This section presents results on the word alignment performance of the RegMT model using various learning models. We expand upon the example word alignment scenario of section 4.4 where we demonstrate the superiority of sparse regression models in word alignment and present experimental results on the word alignment task over the *test* set.

We experiment with the *de-en* language pair on the *test* set using only 1-gram features allowing multiple alignments. The dataset used is described in section 8.1. For each source sentence token, we create a source feature vector, $\Phi_X(\mathfrak{f}_x)$, which has a 1 for the given feature and 0 for other features. Then we estimate the target feature vector as follows:

$$\hat{\Phi_Y}(\mathbf{y}) = \mathbf{W}\Phi_X(\mathfrak{f}_x). \tag{8.14}$$

We set the target features in $\hat{\Phi_Y}(\mathbf{y})$ whose value is greater than the optimized threshold value [3] as sure (S) alignments (following the notation used in (Och and Ney, 2003)) and add these alignments to

[3] See subsection 8.6.1 about how we obtain feature thresholds optimizing the weighted F_1 score.

the word alignment file. Sure alignments correspond to unambiguous alignments and we have retained only the sure alignments in the *de-en test* alignment file, which we have manually created.

We use the threshold values optimized for the weighted F_1 measure, which can create a large number of word alignments at the expense of precision for some models. Alignment performance is measured by the following metrics:

$$\mathcal{A}_{\text{precision}} = \frac{|A \cap P|}{|A|}, \qquad \mathcal{A}_{\text{recall}} = \frac{|A \cap S|}{|S|}, \tag{8.15}$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}, \tag{8.16}$$

where A is the set of gold alignments, P corresponds to the possible alignments including the sure alignments, and AER is the alignment error rate, the lower the better. The results for *de-en test* set are given in Table 8.9. We compare various learning models described in subsection 8.6.1. We observe that FSR is able to achieve better performance than other learning models we compared. However, Moses achieves lower AER score than FSR, which is also reflected in the BLEU scores obtained in Table 8.4. Moses creates a phrase table after pruning the possible alignments and merging them using some heuristics. Pruning removes the connection between co-occurring but unlikely to be aligned phrases. However, in the RegMT approach, we retain all weights coming from the possible feature mappings in a given $\mathbf{W}$ when estimating $\hat{\mathbf{y}}$.

In line with our previous findings in Table 8.7, QP achieves lower $\mathcal{A}_{\text{precision}}$ and slightly higher $\mathcal{A}_{\text{recall}}$ than FSR but QP achieves higher AER and lower $\mathcal{A}_{\text{F}_1}$ than FSR. *logreg* include more alignment links than FSR and achieve higher $\mathcal{A}_{\text{recall}}$ but perform worse in terms of $\mathcal{A}_{\text{precision}}$, $\mathcal{A}_{\text{F}_1}$, and AER. SVR and SVC also use more alignment links but fail to achieve a performance better than FSR. Moses achieves the best $\mathcal{A}_{\text{precision}}$, $\mathcal{A}_{\text{F}_1}$, and AER scores.

Model	$\mathcal{A}_{\text{precision}}$	$\mathcal{A}_{\text{recall}}$	$\mathcal{A}_{\text{F}_1}$	AER	# of links
L_2	0.5998	0.4120	0.4885	0.5115	847
FSR	**0.6215**	**0.5807**	**0.6004**	**0.3996**	1152
QP	0.5769	0.5839	0.5804	0.4196	1248
iter-ϵ	0.1398	0.2092	0.1676	0.8967	1846
SVR	0.0547	0.3682	0.0952	0.9146	8302
logreg	0.1059	0.7429	0.1853	0.8147	8652
SVC	0.0947	0.2587	0.1386	0.8614	3373
Moses	0.7346	0.6464	0.6877	0.3123	1085

Table 8.9: Word alignment performance on *de-en test* set. # of links lists the number of alignment links found by different algorithms.

In Table 8.8, target feature prediction quality is measured using all of the source features whereas in Table 8.9 individual alignment performance is measured for each source token. If we have a decoder that has a performance close to optimal, then recall can be a more important factor.

8.8 Decoding

Regression is a well studied technique and suits well for learning alignment models. However, the learning framework that we use may be simpler than phrase-based machine translation models, where a large number of features in addition to the phrase translation probabilities are used. Also, Moses is obtaining the phrase table after combining the word alignments obtained in both the directions from source to target and target to source. RegMT relies on a single modeling direction from source features to target features and models the phrases found in the training set features directly. We describe the phrase table features used in Moses in section 8.3. In this section, we present decoding results using the Moses decoder and using the graph-based decoding algorithm for generating the translation.

8.8.1 Decoding Results with Moses

In our experiments, we compare the performance of *lasso* versus L_2 after decoding and we measure the effect of the number of instances used for training on the test set performance. We use *dev* set for tuning the weights, which is constructed similarly as the *test* set. We perform individual Moses translation experiments for each test sentence to compare the performance with replacing the phrase table with **W**.

For the *de-en* system, we built a Moses model with the default settings including a maximum sentence length limit of 80 tokens and a 5-gram target language model where about 1.6 million sentences were used for training and 1000 random development sentences including *dev* used for tuning. We obtained 0.3422 BLEU score on *test*.

Individual translations: In the individual translation setting, the training sets are composed of only the set of instances selected for each test sentence and separate SMT models are built with Moses for each sentence (subsection 5.4.3). Individual translation results are given in Table 8.10. Individual SMT training and translation can be preferable due to smaller computational costs and high parallelizability. As we translate a single sentence with each SMT system, tuning weights becomes important and the variance of the weights learned can become high in the individual setting. As we increase the training set size, we observe that the performance gets closer to the Moses system using all of the training corpus.

BLEU	100	250	500
$\leq$ 2-grams	0.3021	0.3397	0.3357

Table 8.10: Individual Moses results with training instances selected individually for each source test sentence.

Transductive RegMT W as the phrase table: The results obtained when the coefficients matrix obtained by RegMT is used as the phrase table for Moses is given in Figure 8.1. Due to computational limitations, we use the weights obtained for L_2 to decode with FSR-*lasso* phrase table and skip tuning for FSR-*lasso*. The learning curve

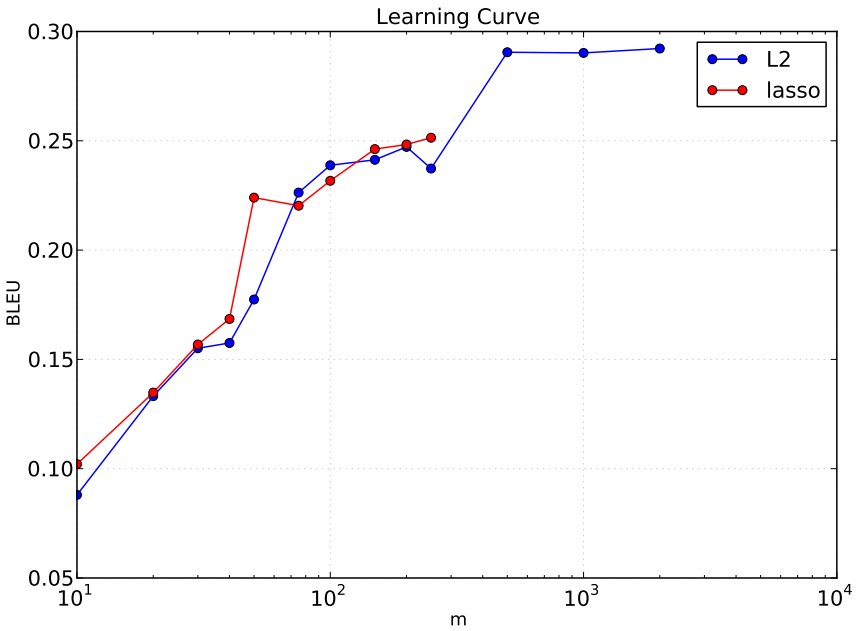

Figure 8.1: *de-en* translation results using Moses decoder with RegMT $\mathbf{W}$ used as the phrase table.

for increasing size of the training set is given in Figure 8.1.

We obtain lower performance with the L_2 model when compared with the individual translations obtained using Moses. *lasso* selects few possible translations for a given source feature. The decrease in the vocabulary and testing smaller target phrase possibilities may result in lower performance although predictions with higher precision scores are obtained as we observe in Table 8.8. The increased precision pays when creating the translation from the bigrams found in the estimation with graph decoding (subsection 8.8.2). We observe similar learning curves both with graph decoding and decoding using Moses. RegMT model may need a larger training set size for achieving better performance when the learned mappings are used as the phrase table. We are able to achieve the Moses performance using 100 training instances with 500 training instances when decoding with Moses using $\mathbf{W}$ as the phrase table. RegMT model is good in estimating the target features but has difficulty in correctly finding the target sentence when using the Moses

decoder. Moses achieves higher F_1 score than *lasso* and this is likely to be due to achieving higher recall on the test set (Table 8.8).

8.8.2 Graph Decoding Experiments

We demonstrate machine translation results using graph decoding on the German-English *test* set as well as in a constrained translation domain from Spanish to English (*es-en*) using the categorized EuTrans corpus (Serrano et al., 2009) containing hotel front desk requests with small vocabulary size. The corpus provides a more restricted translation environment for decoding and contains 9000 training, 1000 development, and 3000 test sentences.

We perform graph-based decoding by first generating a De Bruijn graph (Cortes et al., 2007) from the predicted features of $\mathbf{y}$, $\mathbf{W}\Phi_X(\mathbf{x})$, and then finding Eulerian paths with maximum path weight. We use four features when scoring paths: (1) estimation weight from regression, (2) language model score, (3) brevity penalty as found by $e^{\alpha(l_R - |s|/|path|)}$ for l_R representing the length ratio from the parallel training sentences and $|path|$ representing the length of the current path, (4) future cost as in Moses (Koehn et al., 2007) and weights are tuned using MERT (Och, 2003) on the *de-en dev* set.

Regression results for *de-en* with increasing training data size, m, can be seen in Figure 8.2 where 2-grams are used for decoding. We see a large BLEU gain of *lasso* over L_2 in our transductive learning setting although the performance is lower than Moses.

In the *es-en* translation task, Moses achieves 0.9340 BLEU on the test set using all of the training data. Regression results for increasing training set size can be seen in Figure 8.3 where 1&2-grams are used for decoding. The red line corresponds to the Moses baseline. We see that *lasso* performs better than L_2 in the beginning when we use smaller number of training instances but it performs worse as the training set size increase. These results are comparable to previous work (Serrano et al., 2009).

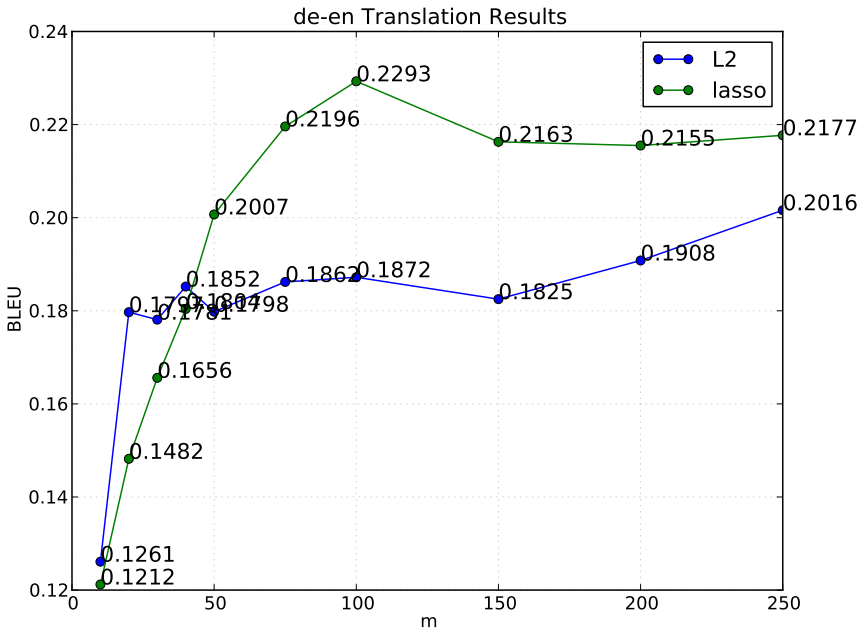

Figure 8.2: *de-en* translation results with graph decoding for increasing training data size, m, using 2-grams.

We demonstrate that sparse L_1 regularized regression performs better than L_2 regularized regression in the *de-en* translation task and in the *es-en* translation task when using small sized training sets, $m \leq 100$. Graph based decoding can provide an alternative to state of the art phrase-based decoding system Moses in translation domains having low vocabulary size.

8.9 Contributions

We use transductive regression techniques to learn mappings between source and target features of given parallel training sentences and use these mappings to generate machine translation outputs. The results show the effectiveness of using L_1 regularization versus L_2 used in ridge regression. We show that L_1 regularized regression performs better than L_2 regularized regression in the target features prediction measurements, in word alignment, and in the translation experiments

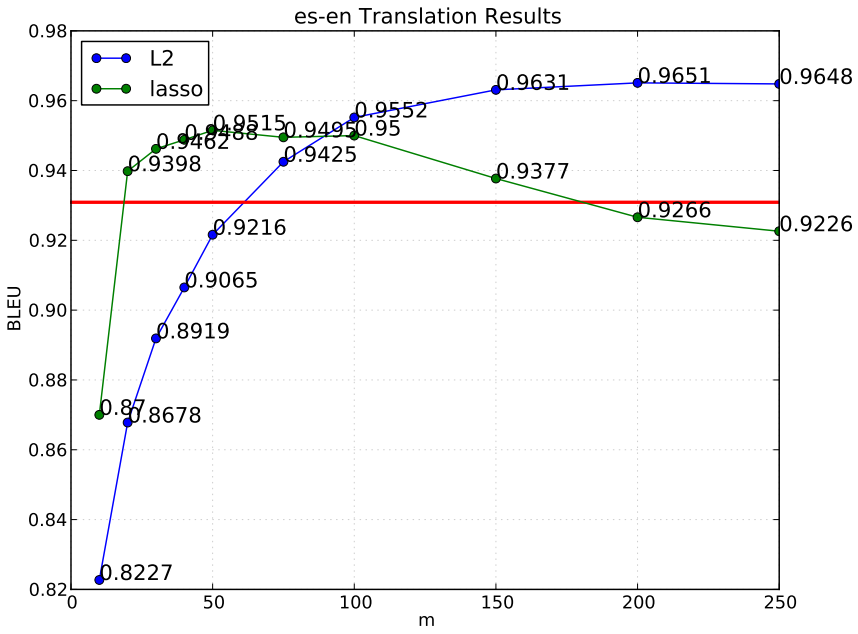

Figure 8.3: *es-en* translation results with graph decoding for increasing training data size, m, using 1&2-grams.

using graph decoding. We present encouraging results when translating from German to English and Spanish to English.

We also demonstrate results when the phrase table of a phrase-based decoder is replaced with the mappings we find with the regression model. When we use Moses to decode, we observe that RegMT model achieves lower performance than Moses system built individually for each test sentence. RegMT model may need a larger training set size for achieving better performance when the mappings are used as the phrase table. RegMT model is good in estimating the target features but has difficulty in correctly finding the target sentence when Moses is used as the decoder. Moses achieves higher F_1 score than *lasso* due to achieving higher recall on the test set (Table 8.8) and higher F_1 score results with higher BLEU performance.

We use F_1 measure for evaluating the performance before performing decoding. F_1 performs better than BLEU when translating into English according to an evaluation by human judgments at WMT'11 (Callison-

Burch et al., 2011). F_1 allows us to evaluate the performance of the RegMT model using the target feature prediction vectors or the coefficients matrices learned or an SMT model using its phrase table without performing the decoding step.

Using the Moses decoder, we observe that we can achieve similar BLEU performance comparing a model having higher weighted F_1 score with a model having more entries in its phrase table increasing the recall but with less accurate entries decreasing the precision. If we have a decoder that has a performance close to optimal, then recall can be a more important factor even though the phrase translation probabilities of the entries are very low.

With an optimal decoder, recall becomes important, which determines the vocabulary. Therefore, having more entries in the phrase table can compensate for worse prediction performance than having more accurate phrase translations as we observe when we compare the L_2 regularized solution with the *lasso* solution.

Acknowledgments

I am thankful to Nicolás Serrano Martínez Santos, one of the authors of (Serrano et al., 2009), for giving me access to the EuTrans corpus.

Chapter 9

Conclusion

This thesis is about the regression based machine translation (RegMT), which provides a learning framework for machine translation, separating learning models for training, training instance selection, feature representation, and decoding. We investigate techniques for making the RegMT model computationally more scalable and more practical by using transductive learning for training, by developing better training instance selection techniques, by building better regression models that fits the sparse nature of the translation problem, by creating translation performance evaluation metrics for our learning approach, and by developing decoding alternatives including graph decoding using the target feature prediction vectors obtained with the regression model.

We use L_2 regularized regression and sparse regression techniques including L_1 regularized regression to predict the target features for given input source features. We develop better training instance selection techniques than previous work from given parallel training sentences for achieving more accurate RegMT models using less training instances. We use graph decoding on the prediction vectors represented in n-gram or word sequence counts space found in the training set. Graph based decoding can provide an alternative to phrase-based decoding in translation domains having low vocabulary size. We also decode using Moses after transforming the learned weight matrix representing the mappings between source and target features to a phrase table.

Our results demonstrate that RegMT is useful for N-best list reranking and in online SMT. We develop training instance selection algorithms that not only make RegMT computationally scalable but also improve the performance of standard SMT systems. We introduce L_1 regularized regression as a better model than L_2 regularized regression for statistical machine translation. We show that sparse regression models are better than L_2 regularized regression for statistical machine translation in predicting target features, estimating word alignments, creating phrase tables, and generating translation outputs.

We use F_1 measure, which performs good when evaluating translations into English according to human judgments. F_1 allows us to evaluate the performance of the RegMT model using the target feature prediction vectors or the coefficients matrices learned or an SMT model using its phrase table without performing the decoding step.

9.1 Research Contributions

We review and list our research contributions and findings including:

- **RegMT is useful for reranking:** We show that regression mapping score is effective in reranking translation outputs and in selecting the best system combinations with encouraging results on different language pairs. We obtain statistically significant improvements over the baseline SMT system by reranking the N-best lists generated by it. N-best list reranking in SMT can be used for demonstrating the usefulness of models.
- **RegMT is useful in online SMT:** Online SMT receives test source sentences one by one and generates a translation for each instance, which can be modeled with the reranking approach given an N-best list. We describe the competitive statistical machine translation (CSMT) problem where we try to select the best translator among competing statistical machine translators and provide a solution using RegMT. The competitive predictor assigns a prob-

ability per model weighted by the sequential performance. We define additive, multiplicative, and loss-based weight updates with exponential loss functions using the RegMT scores as the basis for evaluation at each instance. Without any pre-knowledge of the performance of the translation models, we have been able to achieve the performance of the best model in all translation experiments and we are able to surpass its performance as well as RegMT's performance with some of the weight update models we considered.

- **Training Instance Selection Techniques:** Proper selection of training instances plays an important role to learn feature mappings with limited computational resources accurately. RegMT uses training instance selection techniques to reduce the computational demand and to increase the relevancy of the training data. The instance selection algorithms we developed not only make RegMT computationally more scalable but also improve the performance of standard SMT systems.
  - *High coverage → High BLEU:* We show that target feature coverage (percentage of test features found in the training set) and BLEU (Papineni et al., 2001) performance are correlated.
  - *Training on a small subset of the parallel corpora is enough:* We show that selecting the best 3000 training sentences for each test sentence is sufficient to reach the baseline performance or using 5% of the training data relevant to the test set is sufficient to exceed the baseline SMT model performance.
  - *We focus on the relevancy of the training data used rather than the number of training instances they contain:* Increased number of parallel training sentences size may increase the chance of finding a good match for a test source sentence but it need not lead to better translations.
- **Sparse RegMT:** We use sparse regression models for statistical machine translation and compare the performance with other

learning models. We show that sparse regression models (i.e. L_1 regularized regression or *lasso*) are better than L_2 regularized regression for statistical machine translation:

  - *lasso is better than other learning models in predicting target features:* When predicting target features, *lasso* achieves better performance than the other learning models we compared in terms of weighted BER, precision, recall, and F_1 values.
  - *RegMT is useful in word alignment:* We demonstrate on the test set that L_1 performs better than L_2 in the word alignment task in terms of alignment links precision, recall, and AER. We also demonstrate that L_1 is better than L_2 in the word alignment task on an example word alignment scenario.
  - F_1 *is a good measure for performance evaluation:* F_1 performs good when evaluating translations into English according to an evaluation by human judgments in WMT'11 (Callison-Burch et al., 2011).
  - F_1 *can evaluate the performance before decoding:* We use F_1 measure for evaluating the performance before performing decoding. F_1 allows us to evaluate the performance of the RegMT model using the target feature prediction vectors or the coefficients matrices learned or an SMT model using its phrase table without performing the decoding step.

- **Decoding with RegMT:** We performed RegMT decoding experiments both with graph decoding and with a phrase-based decoder, Moses (Koehn et al., 2007). We demonstrate that sparse L_1 regularized regression performs better than L_2 regularized regression in the German-English translation task and in the Spanish-English translation task when using small sized training sets, ($m \leq 100$). Given the same training data, Moses achieves better BLEU performance than decoding with Moses using the phrase tables generated with the RegMT coefficients matrices. Graph based decoding can

provide an alternative to phrase-based decoding in translation domains having low vocabulary size.

9.2 Future Work

Our new instance selection techniques can be useful for reducing the time and effort for deploying time critical SMT systems. We have observed that a couple of thousand training instances for each test sentence is enough to achieve a translation performance close to using the full training set. The experimental result we obtained can be helpful in developing machine translation systems in disaster and crisis situations (Lewis et al., 2011).

Appendix A

Linear Regression Model and Least Squares Estimation

The linear prediction problem for multinomial distributions is given in the following setting. Given a sequence of observations, $\mathbf{x}^i = (x_1, \ldots, x_i)$ and their previous outcomes, $\mathbf{y}^{i-1} = (y_1, \ldots, y_{i-1})$, the observer predicts y_i for $i = 1, \ldots, n$. Let $\mathcal{T} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be the training set for the prediction problem, where $\mathbf{x} = (x_1, \ldots, x_m)^T$ and $\mathbf{y} = (y_1, \ldots, y_m)^T$. In a linear multinomial regression setting, the linear interpolation function $g(\mathbf{x}^i)$ of model order i gives the estimate of the corresponding y_i:

$$g(\mathbf{x}^i) = \langle \mathbf{w}, \mathbf{x}^i \rangle = \mathbf{w}^T \mathbf{x}^i = \hat{y}_i. \tag{A.1}$$

When $x_i \in \mathbb{R}^n$, $g(\mathbf{x}^i)$ corresponds to a hyperplane. Therefore, the learning problem is choosing $\mathbf{w}$ that minimizes the difference between the estimated $\hat{y}_i$ for y_i for all $i = 1, \ldots, m$.

The cumulative sum of squared error or loss of a regressor g with model order p can be defined as:

$$\mathcal{L}(g, \mathcal{T}) = \mathcal{L}(\mathbf{w}_p, \mathcal{T}) = \sum_{i=1}^{m} (y_i - g(\mathbf{x}_p^i))^2 = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2, \tag{A.2}$$

where $\hat{y}_i$ is the estimated value for y_i and $\mathbf{x}_p^i = (x_{i-p+1}, \ldots, x_i)$ for p^{th} order regressor $\mathbf{w}_p$. The learning problem now boils down to choosing $\mathbf{w}_p$ that minimizes this cumulative squared error.

When y_i is not a linear combination of $\mathbf{w}_p$, i.e. y_i is not an element of $\mathcal{R}(\mathbf{w}_p)$, where $\mathcal{R}(\mathbf{w}_p)$ is the range space of $\mathbf{w}_p$, then:

$$y_i = \mathbf{w}_p^T \mathbf{x}_p^i + \epsilon_i, \tag{A.3}$$

and the error becomes:

$$\epsilon_i = y_i - \mathbf{w}_p^T \mathbf{x}_p^i. \tag{A.4}$$

If we assume that the error is normally distributed:

$$p(y|\mathbf{x}_p^i, \mathbf{w}_p, \sigma^2) = \mathcal{N}(y|\mathbf{w}_p^T \mathbf{x}_p^i, \sigma^2). \tag{A.5}$$

Note that assuming the normality of this error with variance σ^2 and zero mean implies that the distribution of y given $\mathbf{x}$ is unimodal. For the whole training set, the likelihood becomes:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}_p, \sigma^2) = \prod_{i=1}^{m} \mathcal{N}(y_i|\mathbf{w}_p^T \mathbf{x}_p^i, \sigma^2) \tag{A.6}$$

$$= \prod_{i=1}^{m} \frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2\sigma^2}(y_i - \mathbf{w}_p^T \mathbf{x}_p^i)^2). \tag{A.7}$$

The maximum likelihood $\hat{\mathbf{w}}_p$ becomes:

$$\hat{\mathbf{w}}_p = \arg\max_{\mathbf{w}_p} p(\mathbf{y}|\mathbf{x}, \mathbf{w}_p, \sigma^2) = \arg\max_{\mathbf{w}_p} \log p(\mathbf{y}|\mathbf{x}, \mathbf{w}_p, \sigma^2) \tag{A.8}$$

$$\log p(\mathbf{y}|\mathbf{x}, \mathbf{w}_p, \sigma^2) = \sum_{i=1}^{m} \log \mathcal{N}(y_i|\mathbf{w}_p^T \mathbf{x}_p^i, \sigma^2) \tag{A.9}$$

$$= -\frac{m}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - \mathbf{w}_p^T \mathbf{x}_p^i)^2 \tag{A.10}$$

We can find take the derivative of the log likelihood function and set

it to zero to find $\hat{\mathbf{w}}_p$:

$$\begin{aligned}
\nabla \log p(\mathbf{y}|\mathbf{x}, \mathbf{w}_p, \sigma^2) = 0 &= \frac{1}{\sigma^2} \sum_{i=1}^{m} (y_i - \hat{\mathbf{w}}_p^T \mathbf{x}_p^i) \mathbf{x}_p^{i\,T}, \\
\sum_{i=1}^{m} y_i \mathbf{x}_p^{i\,T} &= \sum_{i=1}^{m} \hat{\mathbf{w}}_p^T \mathbf{x}_p^i \mathbf{x}_p^{i\,T}, \\
\mathbf{y}^T \mathbf{X}_p &= \hat{\mathbf{w}}_p^T \mathbf{X}_p \mathbf{X}_p{}^T, \\
\mathbf{X}_p^T \mathbf{y} &= \mathbf{X}_p \mathbf{X}_p{}^T \hat{\mathbf{w}}_p, \\
\hat{\mathbf{w}}_p &= (\mathbf{X}_p \mathbf{X}_p{}^T)^{-1} \mathbf{X}_p^T \mathbf{y}, \qquad \text{(A.11)}
\end{aligned}$$

where $\mathbf{X}_p = [\mathbf{x}_p^{1\,T}, \ldots, \mathbf{x}_p^{m\,T}]$. Equation A.11 is known as the *normal equations* for the least squares. Therefore, the maximum likelihood estimate of $\mathbf{w}$ is the same as the least squares solution to the problem.

In the *multivariate regression model, each x_i is a vector*, forming $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)^T$ and our system of linear equations becomes:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}, \qquad \text{(A.12)}$$

and the error vector can be defined as:

$$\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\mathbf{w}. \qquad \text{(A.13)}$$

The loss function, $\mathcal{L}(\mathbf{w}, \mathcal{T})$ can be written as follows:

$$\mathcal{L}(\mathbf{w}, \mathcal{T}) = \|\boldsymbol{\epsilon}\|^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}). \qquad \text{(A.14)}$$

By taking the derivative of the loss function with respect to $\mathbf{w}$ and setting it equal to zero, we obtain the *normal equations*:

$$\mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y} \qquad \text{(A.15)}$$

which leads to the following solution assuming that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \qquad \text{(A.16)}$$

The inverse operation is approximately cubic in the number of equations, thus finding $\hat{\mathbf{w}}$ is in $O(n^3)$, when $\mathbf{X}$ is m by n.

When $\mathbf{y} \notin \mathcal{R}(\mathbf{X})$ then the least squares solution, $\hat{\mathbf{w}}$ is the one that minimizes the error. Therefore,

$$\| \mathbf{y} - \mathbf{X}\hat{\mathbf{w}} \|^2 \leq \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2 \tag{A.17}$$

for all $\mathbf{w} \in \mathbb{R}^n$ (Kailath et al., 2000). If $\mathbf{y} \in \mathcal{R}(\mathbf{X})$ then $\hat{\mathbf{w}}$ will be an exact solution but there can be other solutions if $\mathbf{X}$ is not full rank.

We can also use different regressors to estimate a multivariate target, $\mathbf{y}$. *When each w_i is a vector of length n*, forming $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)^T$, we can represent the *least squares estimation problem* similarly, where we are trying to find an $\mathbf{x}$ that is closest to $\mathbf{y}$ that satisfies the following equation:

$$\mathbf{W}\mathbf{x} \cong \mathbf{y}, \tag{A.18}$$

where $\cong$ corresponds to the inconsistency, which is a result of $\mathbf{y} \notin \mathcal{R}(\mathbf{W})$. There will be many solutions when $\mathbf{W}$ is not full rank.

When we are given a set of multivariate targets, $\mathbf{y}_i$, for $i = 1, \ldots, m$, the least squares regression problem becomes solving the following system of linear equations:

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \mathbf{V}. \tag{A.19}$$

$\mathbf{V}$ represents the error matrix. After taking the derivative of the cost function, $\mathcal{L}(\mathbf{W}) = \| \mathbf{Y} - \mathbf{W}\mathbf{X} \|^2$, with respect to $\mathbf{W}$ and zetting it to zero, we obtain the least squares solution:

$$\begin{aligned} (\mathbf{Y} - \mathbf{W}\mathbf{X})\mathbf{X}^T &= 0 && \text{(A.20)} \\ \mathbf{W}\mathbf{X}\mathbf{X}^T &= \mathbf{Y}\mathbf{X}^T && \text{(A.21)} \\ \mathbf{W} &= \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} && \text{(A.22)} \end{aligned}$$

The training error of a model as found by the loss obtained with the learned model will be an optimistic estimate of the error obtained on the test set (Hastie et al., 2009). Mallows C_p statistic estimates the prediction risk for a model by the sum of the lack of fit and the complexity penalty and this statistic yields the same results with cross-validation for the linear regression task (Wasserman, 2004).

A.1 Regularized Least Squares and Dual Representation

Regularized least squares is also known as ridge regression. The loss function is defined as follows:

$$\mathcal{L}_\lambda(\mathbf{w}, \mathcal{T}) = \sum_{i=1}^{m} (y_i - g(\mathbf{x}_i))^2 + \lambda \parallel \mathbf{w} \parallel^2 \tag{A.23}$$

where $\lambda \geq 0$.

Proposition 2. *The solution to the cost function given in Equation A.23 can be found by the following identities:*

$$\begin{aligned} \boldsymbol{w} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{X}^T\boldsymbol{y} \quad \textit{(primal)} \\ \boldsymbol{w} &= \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\lambda} \boldsymbol{I}_m)^{-1}\boldsymbol{y} \quad \textit{(dual)} \end{aligned} \tag{A.24}$$

Proof. We follow a derivation close to the one found in (Taylor and Cristianini (2004), section 2.2.2). By taking the derivative with respect to $\mathbf{w}$ and setting the resulting equation to zero, we obtain:

$$\mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_n)\mathbf{w} = \mathbf{X}^T\mathbf{y}, \tag{A.25}$$

where $\mathbf{I}_n$ is the identity matrix of size n. If $\lambda > 0$, then $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_n)$ is always invertible:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_n)^{-1}\mathbf{X}^T\mathbf{y} \tag{A.26}$$

Equation A.26 is known as the primal solution. The complexity of finding $\mathbf{w}$ is still in $O(n^3)$. By dividing Equation A.25 by λ, we can obtain the following:

$$\mathbf{w} = \lambda^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^T\boldsymbol{\beta}, \tag{A.27}$$

where $\boldsymbol{\beta} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$. Thus,

$$\lambda\boldsymbol{\beta} = (\mathbf{y} - \mathbf{X}\mathbf{X}^T\boldsymbol{\beta}) \tag{A.28}$$

$$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}_m)\boldsymbol{\beta} = \mathbf{y} \tag{A.29}$$

$$\boldsymbol{\beta} = (\mathbf{G} + \boldsymbol{\lambda}\mathbf{I}_m)^{-1}\mathbf{y} \tag{A.30}$$

where $\mathbf{G} = \mathbf{X}\mathbf{X}^T$ is the Gram matrix, which is an $m \times m$ symmetric matrix with:

$$\mathbf{G}_{i,j} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) \tag{A.31}$$

where $k(x, y)$ is the kernel function defined as:

$$k(x, y) = \phi(x)^T \phi(y). \tag{A.32}$$

The complexity of solving for $\boldsymbol{\beta}$ is in $O(m^3)$. Equation A.30 is known as the dual solution. □

The prediction function becomes:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{y}^T (\mathbf{G} + \boldsymbol{\lambda}\mathbf{I}_m)^{-1} \mathbf{X}\mathbf{x} = \mathbf{y}^T (\mathbf{G} + \boldsymbol{\lambda}\mathbf{I}_m)^{-1} \mathbf{k}, \tag{A.33}$$

where $k_i = \langle \mathbf{x}_i, \mathbf{x} \rangle$. If the feature space size n is larger than the number of training examples m, then it is computationally less costly to use the dual solution. In the dual representation, the least-squares solution is found entirely by the kernel function $k(\mathbf{x}, \mathbf{y})$.

We can also find the dual representation by using the matrix inversion lemma, which is a very useful matrix identity (Kailath et al., 2000):

Lemma 3 (Matrix Inversion)**.** *For any given matrices, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$:*

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{D})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{C}^{-1} + \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{D}\boldsymbol{A}^{-1} \tag{A.34}$$

If $\boldsymbol{C}$ is not invertible, the following can be used:

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{D})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{D}\boldsymbol{A}^{-1} \tag{A.35}$$

We can invert the primal solution given in Equation A.26 by using the matrix inversion lemma to obtain the dual solution:

$$\begin{aligned}
\mathbf{w} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_n)^{-1}\mathbf{X}^T\mathbf{y} \\
&= \left[\lambda^{-1}\mathbf{I}_n - \lambda^{-1}\mathbf{X}^T(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1}\lambda^{-1}\mathbf{X}\right]\mathbf{X}^T\mathbf{y} \\
&= \left[\lambda^{-1}\mathbf{X}^T - \lambda^{-1}\mathbf{X}^T(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1}\lambda^{-1}\mathbf{X}\mathbf{X}^T\right]\mathbf{y} \\
&= \left[\lambda^{-1}\mathbf{X}^T - \lambda^{-1}\mathbf{X}^T(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1}(-\mathbf{I}_m + \mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^T)\right]\mathbf{y} \\
&= \left[\lambda^{-1}\mathbf{X}^T + \lambda^{-1}\mathbf{X}^T(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^T)^{-1} - \lambda^{-1}\mathbf{X}^T)\right]\mathbf{y} \\
&= \mathbf{X}^T(\lambda\mathbf{I}_m + \mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}
\end{aligned} \tag{A.36}$$

A.2 Stochastic Least Squares Estimation

Stochastic estimation as discussed in (Kailath et al., 2000) seeks to find an estimate, $\hat{\mathbf{y}}$ of a random variable $\mathbf{y}$ that is dependent on some other random variable $\mathbf{x}$ by using an estimator $h(\mathbf{x})$, which is optimal based on some criteria. For example, the optimal least mean squares estimator of $\mathbf{y}$ given $\mathbf{x}$ is $E[\mathbf{y}|\mathbf{x}]$, which becomes $E[\mathbf{y}]$ when $\mathbf{y}$ and $\mathbf{x}$ are independent. The expectation of the conditional requires the knowledge of the joint distribution of $\mathbf{y}$ and $\mathbf{x}$. When $h(.)$ is a linear estimator we only need the first ($E[\mathbf{y}]$, $E[\mathbf{x}]$) and the second order ($\mathtt{cov}(\mathbf{y},\mathbf{x}) = E[\mathbf{y}\mathbf{x}^T] = \mathbf{R}_{\mathbf{yx}}$, $\mathtt{var}(\mathbf{y},\mathbf{y}) = E[\mathbf{y}\mathbf{y}^T] = \mathbf{R}_{\mathbf{y}}$, $\mathtt{var}(\mathbf{x},\mathbf{x}) = E[\mathbf{x}\mathbf{x}^T] = \mathbf{R}_{\mathbf{x}}$) statistics to estimate $\mathbf{y}$. The optimal linear least mean squares estimator (LLMSE) theorem follows (Kailath et al., 2000).

Theorem 4 (Optimal Linear Least Mean Squares Estimators)**.** *The LLMS estimator of the zero mean random variable* $\boldsymbol{y}$ *given the zero mean random variable* $\boldsymbol{x}$ *is found by the normal equations:*

$$\hat{\boldsymbol{W}}\boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{R}_{\boldsymbol{yx}} \tag{A.37}$$

where $\hat{\boldsymbol{W}}$ *is the optimal coefficients matrix. The corresponding minimum mean square error (MMSE) is given by:*

$$\arg\min_{\hat{\boldsymbol{W}}} \quad \boldsymbol{R}_{\boldsymbol{y}} - \hat{\boldsymbol{W}}\boldsymbol{R}_{\boldsymbol{xy}} = \boldsymbol{R}_{\boldsymbol{y}} - \boldsymbol{R}_{\boldsymbol{yx}}\hat{\boldsymbol{W}}^T \tag{A.38}$$

where we minimize $E[\boldsymbol{e}\boldsymbol{e}^T]$ *and the error* $\boldsymbol{e}$ *is* $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$*.*

Proof. $\hat{\mathbf{W}}$ obtains the minimum error among all $\mathbf{W}$. If we follow a geometric approach, then $\mathbf{e} \perp \mathbf{x}$ and therefore $E[\mathbf{e}\mathbf{x}^T] = \langle \mathbf{e}, \mathbf{x} \rangle = 0$ for a generalized inner product defined as $\langle \mathbf{a}, \mathbf{b} \rangle = E[\mathbf{a}, \mathbf{b}^T]$ for two given vectors $\mathbf{a}$ and $\mathbf{b}$.

$$\begin{aligned} E[\mathbf{e}\mathbf{x}^T] = E[(\mathbf{y} - \hat{\mathbf{W}}\mathbf{x})\mathbf{x}^T] &= 0 \\ E[\mathbf{y}\mathbf{x}^T] &= \hat{\mathbf{W}}E[\mathbf{x}\mathbf{x}^T] \\ \mathbf{R}_{\mathbf{yx}} &= \hat{\mathbf{W}}\mathbf{R}_{\mathbf{x}} \\ \hat{\mathbf{W}} &= \mathbf{R}_{\mathbf{yx}}\mathbf{R}_{\mathbf{x}}^{-1} \end{aligned} \tag{A.39}$$

The corresponding MMSE:

$$\begin{aligned} E[(\mathbf{y}-\hat{\mathbf{y}})(\mathbf{y}-\hat{\mathbf{y}})^T] &= E[(\mathbf{y}-\hat{\mathbf{y}})\mathbf{y}^T - (\mathbf{y}-\hat{\mathbf{y}})\hat{\mathbf{y}}^T] \\ &= E[(\mathbf{y}-\hat{\mathbf{W}}\mathbf{x})\mathbf{y}^T \text{ since } \langle \mathbf{y}-\hat{\mathbf{y}}, \hat{\mathbf{y}} \rangle = 0 \\ &= \mathbf{R_y} - \hat{\mathbf{W}}\mathbf{R_{xy}} \end{aligned} \tag{A.40}$$

□

Kailath et al. (2000) show that stochastic least squares estimation and the deterministic least squares estimation learning settings that we discussed earlier are equivalent or dual.

Appendix B

Statistical Significance Testing of Results

For small sized test sets, comparing the performance of two different SMT systems can pose a challenge especially when their performance difference is small. Even when the difference may seem large, we are still interested whether the difference is significant enough to say that one is better than the other.

Confidence interval specifies an interval of values that contain the true value with a high probability, where the width of the confidence interval determines the accuracy for the estimate (Leon-Garcia, 1994). Koehn (2004) evaluates statistical significance tests for machine translation performance. For N sentences, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, sample mean and variance of sentence scores, $\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\}$, where $f(\cdot)$ is a scoring function are found as follows:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{n} f(\mathbf{x}_i), \tag{B.1}$$

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(f(\mathbf{x}_i) - \bar{x})^2. \tag{B.2}$$

We are interested in finding a confidence interval, $[\bar{x} - t\frac{s}{\sqrt{N}}, \bar{x} + t\frac{s}{\sqrt{N}}]$, where the value for t is found by the Student's t-distribution for $N-1$ degrees of freedom with confidence level α. Thus, the true score lies in the interval with probability $1-\alpha$ or for μ representing the true mean

of the scores:

$$P(\bar{x} - t\frac{s}{\sqrt{N}} \leq \mu \leq \bar{x} + t\frac{s}{\sqrt{N}}) = 1 - \alpha. \tag{B.3}$$

Bootstrap resampling method randomly samples from a given test set with replacement multiple times. Given N data points, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, a new dataset $\mathbf{X}_b$ is created by randomly selecting N instances from $\mathbf{X}$ with replacement, where $\mathbf{X}_b$ may have repeated or missing instances (Bishop, 2006). This is repeated B times to produce $\mathbf{X}_1, \ldots, \mathbf{X}_B$, which are then used to examine the behavior of any quantity computed from $\mathbf{X}_b$ (i.e. the mean $\bar{X}_b$) (Hastie et al., 2009). In machine translation evaluation, BLEU scores of each bootstrap sample are calculated to estimate the true BLEU score of the system within the confidence interval for given confidence level. Usually, 95% confidence interval (0.95 confidence level) is used where $\alpha = 0.05$. Let the function $BLEU_S(\cdot)$ return the BLEU score of the translation of a dataset generated by system S for which the reference is known, then we can calculate the interval by using the following statistics:

$$\bar{X} = \frac{1}{B}\sum_{i=1}^{B} BLEU_S(\mathbf{X}_i), \tag{B.4}$$

$$s^2 = \frac{1}{B-1}\sum_{i=1}^{B}(BLEU_S(\mathbf{X}_i) - \bar{X})^2, \tag{B.5}$$

$$1 - \alpha = P(\bar{X} - t\frac{s}{\sqrt{B}} \leq \mu \leq \bar{X} + t\frac{s}{\sqrt{B}}). \tag{B.6}$$

Paired bootstrap resampling (Koehn, 2004) uses bootstrap resampling to create virtual test sets to estimate which of the two translation systems, S_1 or S_2, perform better by counting the number of times one is better than the other with the given confidence level. We can estimate the confidence interval by measuring the BLEU score differences over the bootstrap datasets and if the left boundary of the confidence interval is greater than zero, accept that the improvement is statistically significant (Macherey and Och, 2007). This is equivalent

to saying that the mean value, 0, for the null hypothesis, which claims that the two systems are not different, is in the confidence interval. Therefore, at this confidence level, the two systems are not different.

Paired t-test can be used to test the null hypothesis and learn the confidence level of the test where the null hypothesis is given as follows:

$\mathcal{H}_0$: "S_1 and S_2 have exactly the same performance".

We calculate the following t-statistic and compare with the value from the t-distribution table corresponding to the desired level of significance (Wasserman, 2004):

$$\bar{\Delta} = \frac{1}{N}\sum_{i=1}^{N}\Delta_i = \frac{1}{N}\sum_{i=1}^{N}(BLEU_{S_1}(\mathbf{X}_i) - BLEU_{S_2}(\mathbf{X}_i)), \quad \text{(B.7)}$$

$$s^2 = \frac{1}{B-1}\sum_{i=1}^{B}(\Delta_i - \bar{\Delta})^2, \quad \text{(B.8)}$$

$$\hat{t} = \frac{\bar{\Delta} - \Delta_0}{s/\sqrt{B}}, \quad \text{(B.9)}$$

where Δ_0 stands for the Δ for $\mathcal{H}_0$, which is zero and we have $B-1$ degrees of freedom. If $\hat{t}$ is greater than the tabulated value for the level of significance chosen (i.e. $\alpha = 0.05$) for t-distribution, then we accept that the difference between the two systems is significant. The smallest confidence level α that rejects $\mathcal{H}_0$ is called the p-value, since for all $\alpha' > \alpha$, the hypothesis will be rejected (Wasserman, 2004). By calculating $\hat{t}$, we can find the p-value for the null hypothesis from the distribution of t, which is also reported for statistical significance tests. A p-value less than 0.05 is considered to be strong evidence against $\mathcal{H}_0$.

We can use paired t-test to compare the performance at the sentence level. However, BLEU score may not be informative at the sentence level as most of the higher order n-grams will not find a corresponding match. Other evaluation metrics can be more useful for sentence by sentence comparison. Bootstrap resampling methods infer statistics from the samples with a cost of reduced accuracy and other test

Algorithm 4: Approximate randomization test for comparing two systems.

$c = 0$;
Let S_1's and S_2's output be $\mathbf{X}^1 = \{\mathbf{x}_1^1, \ldots, \mathbf{x}_N^1\}$ and $\mathbf{X}^2 = \{\mathbf{x}_1^2, \ldots, \mathbf{x}_N^2\}$ respectively ;
Let $d = |f(\mathbf{X}^1) - f(\mathbf{X}^2)|$ be the initial score difference using the scoring function $f(\cdot)$;
for $r \leftarrow 1$ **to** R **do**
    $\mathbf{X}_r^1 = \{\}$; $\mathbf{X}_r^2 = \{\}$; // *Shuffled datasets*
    for $i \leftarrow 1$ **to** N **do**
        $rand = random(0, 1)$; // *Random number in range* $[0, 1]$
        if $rand > 0.5$ **then**
            $\mathbf{X}_r^1[i] = \mathbf{X}^1[i]$;
            $\mathbf{X}_r^2[i] = \mathbf{X}^2[i]$;
        else
            $\mathbf{X}_r^1[i] = \mathbf{X}^2[i]$;
            $\mathbf{X}_r^2[i] = \mathbf{X}^1[i]$;
    if $|f(\boldsymbol{X}_r^1) - f(\boldsymbol{X}_r^2)| > d$ **then**
        $c = c + 1$;
$p = \frac{c+1}{R+1}$;

statistics such as the approximate randomization test may measure the statistical significance of the difference better (Riezler and Maxwell, 2005).

Approximate randomization test shuffles the labels of the instances (i.e. which system they are produced by) to test the confidence level of $\mathcal{H}_0$. Approximate randomization tests only some of the permutations since full randomization requires 2^N permutations for N scores. The p-value is found by $\frac{c+1}{R+1}$, where c is the number tests that pass and R is the number of randomized tests (Riezler and Maxwell, 2005). The pseudocode is given in Algorithm 4.

The more the number of samples selected and their relative size, the better the estimates would be. Therefore, we try to optimize both with

the amount of computation time we have.

Appendix C

Mehmet Ergun Biçici's Biography

Mehmet Ergun Biçici had his primary schooling in Izmir, Turkey. He went to Bornova Anatolian High School (BAL) for secondary school education and to Izmir Science High School (IFL) for high school. He received his Bachelor of Science degree in Computer Science from Bilkent University, Ankara, Turkey in year 2000. He obtained Master of Science degree in Computer Science from North Carolina State University, Raleigh, USA, in 2002. From 2002 to 2005, he was a PhD student at North Carolina State University. In 2005, he started the PhD program in Computer Engineering at Koç University, Istanbul, Turkey where he worked on this thesis. In 2011, he graduated from the PhD program in Computer Engineering at Koç University.

Appendix D

Mehmet Ergun Biçici's Publications

Ergun Bicici and Deniz Yuret. Instance selection for machine translation using feature decay algorithms. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 272–283, Edinburgh, Scotland, July 2011a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2131.

Ergun Bicici and Deniz Yuret. RegMT system for machine translation, system combination, and evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 323–329, Edinburgh, Scotland, July 2011b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2137.

Ergun Bicici and Deniz Yuret. L1 regularized regression for reranking and system combination in machine translation. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, Uppsala, Sweden, July 2010a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W10/W10-1741.pdf.

Ergun Bicici and S. Serdar Kozat. Adaptive model weighting and transductive regression for predicting best system combinations. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, Uppsala, Sweden, July

2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W10/W10-1740.pdf.

Ergun Bicici and Deniz Yuret. L1 regularization for learning word alignments in sparse feature matrices. In *Proceedings of the Computer Science Student Workshop, CSW'10*, Koc University, Istinye Campus, Istanbul, Turkey, February 2010b. URL URL:http://myweb.sabanciuniv.edu/csw/.

Deniz Yuret and Ergun Bicici. Modeling morphologically rich languages using split words and unstructured dependencies. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 345–348, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-2087.

Ergun Biçici. Context-based sentence alignment in parallel corpora. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2008), LNCS*, Haifa, Israel, February 2008a. Springer-Verlag.

Ergun Biçici and Marc Dymetman. Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2008), LNCS*, Haifa, Israel, February 2008. Springer-Verlag.

Ergun Biçici. Consensus ontologies in socially interacting multiagent systems. *Journal of Multiagent and Grid Systems*, 2008b.

Ergun Biçici. Local context selection for aligning sentences in parallel corpora. In *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007), LNAI*, volume 4635, pages 82–93, Roskilde, Denmark, August 2007. Springer-Verlag.

Ergun Biçici and Deniz Yuret. Locally scaled density based clustering. In *Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2007), LNCS 4431*, volume 4431, pages 739–748, Warsaw, Poland, April 2007. ISBN 978-3-540-71589-4. Springer-Verlag.

Ergun Biçici and Deniz Yuret. Clustering word pairs to answer analogy questions. In *Proceedings of the Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN '06)*, pages 277–284, Akyaka, Mugla, June 2006.

Ergun Biçici. Generating consensus ontologies among socially interacting agents. In *Proceedings of the International Workshop on Agents and Multiagent Systems, from Theory to Application (AMTA 2006)*, Québec City, Québec, Canada, June 2006a.

Ergun Biçici. Consensus ontology generation in a socially interacting multiagent system. In *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, pages 279–284, Washington, DC, USA, September 2006b. IEEE Computer Society.

Ergun Biçici. Formal consensus. Research note, 2005. Technical Report.

Ergun Biçici. Relational learning from drug adverse events reports. In *North Carolina Symposium on Biotechnology & Bioinformatics, IEEE Tech4Life*, 2004.

Ergun Biçici and Robert St. Amant. Reasoning about the functionality of tools and physical artifacts. Technical Report TR-2003-22, North Carolina State University, 2003. MSc Thesis Work.

Sameer Rajyaguru, Ergun Biçici, and Robert St. Amant. Intelligent user interface tools. MSc Work, 2003.

Ergun Biçici. Prolegomenon to commonsense reasoning in user interfaces. *ACM Crossroads*, 9(1), 2002. URL

http://home.ku.edu.tr/~ebicici/publications/2002/ACMCrossroads2002/CommonsenseUI6.htm.

Ergun Biçici and Baris Arslan. Corpus annotation tool for turkish language. Technical report, Bilkent University, 2000. Senior Project under the supervision of Kemal Oflazer.

References

Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Active learning and crowd-sourcing for machine translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Sankaranarayanan Ananthakrishnan, Rohit Prasad, David Stallard, and Prem Natarajan. Discriminative sample selection for statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 626–635, Cambridge, MA, October 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D10-1061.

Auer, Cesa-Bianchi, and Gentile. Adaptive and self-confident on-line learning algorithms. *JCSS: Journal of Computer and System Sciences*, 64, 2002.

Francis Bach, R. Jenatton, J. Mairal, and G. Obozinski. *Convex optimization with sparsity-inducing norms*. MIT Press, Cambridge, MA, USA, 2011.

Francis R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 33–40. ACM, 2005. ISBN 1-59593-180-5. URL http://doi.acm.org/10.1145/1102351.1102356.

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In Carla E. Brodley, editor, *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. URL http://doi.acm.org/10.1145/1015330.1015424.

Gökhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007. ISBN 0262026171.

Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073017. URL http://www.aclweb.org/anthology/P01-1005.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Ergun Biçici and S. Serdar Kozat. Adaptive model weighting and transductive regression for predicting best system combinations. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W10/W10-1740.pdf.

Ergun Biçici and Deniz Yuret. L_1 regularization for learning word alignments in sparse feature matrices. In *Proceedings of the Computer*

Science Student Workshop, CSW'10, Koc University, Istinye Campus, Istanbul, Turkey, February 2010a. URL URL:http://myweb.sabanciuniv.edu/csw/.

Ergun Biçici and Deniz Yuret. L_1 regularized regression for reranking and system combination in machine translation. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, Uppsala, Sweden, July 2010b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W10/W10-1741.pdf.

Ergun Biçici and Deniz Yuret. Instance selection for machine translation using feature decay algorithms. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 272–283, Edinburgh, Scotland, July 2011a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2131.

Ergun Biçici and Deniz Yuret. RegMT system for machine translation, system combination, and evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 323–329, Edinburgh, Scotland, July 2011b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2137.

Ergun Biçici. Local context selection for aligning sentences in parallel corpora. In *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007), LNAI*, volume 4635, pages 82–93, Roskilde, Denmark, August 2007. © Springer-Verlag.

Ergun Biçici. Context-based sentence alignment in parallel corpora. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2008), LNCS*, Haifa, Israel, February 2008. © Springer-Verlag.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Michael Bloodgood and Chris Callison-Burch. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 854–864, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1088.

Avrim Blum. On-line algorithms in machine learning. In *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl*, pages 306–325. Springer, 1996.

Avrim Blum and Yishay Mansour. Learning, regret minimization and equilibria. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory (Cambridge University Press, 2007)*. 2007.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 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 MetricsMATR*, pages 17–53, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1703.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omer F. Zaidan. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, England, July 2011. Association for Computational Linguistics.

Olivier Chapelle, Vladimir Vapnik, and Jason Weston. Transductive inference for estimating values of functions. In *NIPS*, pages 421–427, 1999.

Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July 2002. doi: 10.3115/1118693.1118694. URL http://www.aclweb.org/anthology/W02-1001.

Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression framework for learning string-to-string mappings. In Gokhan H. Bakir, Thomas Hofmann, and Bernhard Sch editors, *Predicting Structured Data*, pages 143–168. The MIT Press, September 2007.

Hal Daumé III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, University of Southern California, Los Angeles, CA, August 2006.

Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. ISSN 00129658. URL http://www.jstor.org/stable/1932409.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Iddo Drori. Fast $\ell 1$ minimization by iterative thresholding for multidimensional nmr spectroscopy. *EURASIP Journal on Advances in Signal Processing*, 2007, 2007.

Matthias Eck, Stephan Vogel, and Alex Waibel. Low cost portability for statistical machine translation based on n-gram coverage. In *Proceedings of the 10th Machine Translation Summit, MT Summit X*, pages 227–234, Phuket, Thailand, September 2005.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

Michel Galley and Christopher D. Manning. Quadratic-time dependency parsing for machine translation. 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 773–781, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1087.

Gholamreza Haffari and Anoop Sarkar. Active learning for multilingual statistical machine translation. 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 181–189, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1021.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N09/N09-1047.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36:381–410, 2002. URL /ref/oflazer/oflazer02statistical.pdf.

Saša Hasan, Richard Zens, and Hermann Ney. Are very large N-best lists useful for SMT? In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 57–60, Rochester, New York, April 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-2015.

Trevor Hastie, Jonathan Taylor, Robert Tibshirani, and Guenther Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1, 2006.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, 2nd edition, 2009.

K.K. Herrity, A.C. Gilbert, and J.A. Tropp. Sparse approximation via iterative thresholding. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, page III, May 2006. doi: 10.1109/ICASSP.2006.1660731.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. *Distributed representations*, pages 77–109. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://portal.acm.org/citation.cfm?id=104279.104287.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. International Conference on Machine Learning (ICML)*, 1999.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice-Hall Inc., New Jersey, 2000.

KernelLearning. NIPS Workshop: Kernel learning: Automatic selection of optimal kernels, 2008. URL http://www.cs.nyu.edu/learning_kernels/. http://www.cs.nyu.edu/learning_kernels/.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999a. ISSN 0891-2017.

Kevin Knight. A statistical machine translation tutorial workbook, August 1999b. URL http://www.isi.edu/natural-language/mt/wkbk.rtf. URL: http://www.isi.edu/natural-language/mt/wkbk.rtf.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Philipp Koehn. Statistical machine translation: the basic, the novel, and the speculative, 2006. Tutorial at EACL 2006.

Philipp Koehn and Kevin Knight. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 2001.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *HLT-NAACL*, number June. Association for Computational Linguistics, 2003.

Philipp Koehn, Marcello Federico, Wade Shen, Nicola Bertoldi, Ondrej Bojar, Chris Callison-Burch, Brooke Cowan, Chris Dyer, Hieu Hoang, Richard Zens, Alexandra Constantin, Christine Corbett Moran, and Evan Herbst. Open source toolkit for statistical machine translation, 2006. Report of the 2006 Summer Workshop at Johns Hopkins University.

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. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Assoc. for*

Computational Linguistics, pages 177–180, Prague, Czech Republic, June 2007.

S.S. Kozat and A.C. Singer. Further results in multistage adaptive filtering. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, 2:1329–1332, 2002.

Alberto Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley Pub Co, 2nd edition, 1994.

William Lewis, Robert Munro, and Stephan Vogel. Crisis mt: Developing a cookbook for mt in crisis situations. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 501–511, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2164.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220271. URL http://www.aclweb.org/anthology/P06-1096.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

Nick Littlestone and Manfred K. Warmuth. The Weighted Majority Algorithm. Technical Report UCSC-CRL-91-28, University of California, Santa Cruz, Jack Baskin School of Engineering, October 26, 1992. URL ftp://ftp.cse.ucsc.edu/pub/tr/ucsc-crl-91-28.ps.Z.

Adam Lopez. A survey of statistical machine translation. Technical

report, 2007. University of Maryland technical report, UMIACS-TR-2006-47.

Yajuan Lü, Jin Huang, and Qun Liu. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1036.

Wolfgang Macherey and Franz Josef Och. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 986–995, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1105.

Ray Maleh. *Efficient Sparse Approximation Methods for Medical Imaging*. PhD thesis, The University of Michigan, 2009.

A. Mandal, D. Vergyri, W. Wang, J. Zheng, A. Stolcke, G. Tur, D. Hakkani-Tur, and N.F. Ayan. Efficient data selection for machine translation. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 261 –264, 2008. doi: 10.1109/SLT.2008.4777890.

Ruslan Mitkov, editor. *The Oxford Handbook of Computational Linguistics*. Oxford University Press, Oxford, 2003. ISBN 0-19-823882-7.

M. Mørup and L. H. Clemmensen. Multiplicative updates for the LASSO. In *MLSP 2007*, 2007. URL http://www2.imm.dtu.dk/pubdb/p.php?5254.

Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 681–688, New

York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273582.

Franz Josef Och. Minimum error rate training in statistical machine translation. *Association for Computational Linguistics*, 1:160–167, 2003. doi: http://dx.doi.org/10.3115/1075096.1075117.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302, 2002. URL http://www.aclweb.org/anthology/P02-1038.pdf.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Kemal Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996. ISSN 0891-2017.

Kemal Oflazer. Statistical machine translation into a morphologically complex language. In *CICLing*, pages 376–387, 2008.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2001. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073083.1073135.

Florian A. Potra and Stephen J. Wright. Interior-point methods. *J. Comput. Appl. Math.*, 124:281–302, December 2000. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/S0377-0427(00)00433-7. URL http://dx.doi.org/10.1016/S0377-0427(00)00433-7.

Calyampudi Radhakrishna Rao and Helge Toutenburg. *Linear Models: Least Squares and Alternatives (Springer Series in Statistics)*. Springer, 1999.

Sujith Ravi and Kevin Knight. Does giza++ make search errors? *Computational Linguistics*, 36(3):295–302, 2010.

Stefan Riezler and John T. Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W05/W05-0908.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

Nicolas Serrano, Jesus Andres-Ferrer, and Francisco Casacuberta. On a kernel regression approach to machine translation. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 394–401, 2009. URL http://dx.doi.org/10.1007/978-3-642-02172-5_51.

Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

J. Shawe Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. URL http://www.kernel-methods.net/.

Robert J. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-361-7.

Berwin A. Turlach, William N. Venables, and Stephen J. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, August 2005.

Raghavendra Udupa and Hemanta Kumar Maji. Computational complexity of statistical machine translation. In *EACL*, 2006.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. Transductive learning for statistical machine translation. In *ACL*. The Association for Computer Linguistics, 2007. URL http://aclweb.org/anthology-new/P/P07/P07-1004.pdf.

Jakob Uszkoreit, Jay Ponte, Ashok Popat, and Moshe Dubiner. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1101–1109, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL http://www.aclweb.org/anthology/C10-1124.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York., 2000.

S. V. N. Vishwanathan and Alex Smola. Fast kernels for string and tree matching. *Advances in Neural Information Processing Systems*, 15, 2003. URL http://citeseer.ist.psu.edu/652394.html.

Zhuoran Wang and John Shawe-Taylor. Kernel regression framework for machine translation: UCL system description for WMT 2008 shared translation task. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 155–158, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W08/W08-0322.

Zhuoran Wang, John Shawe-Taylor, and Sandor Szedmak. Kernel regression based machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 185–188, Rochester, New York, April 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-2047.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 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, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1080.

Deniz Yuret and Ergun Biçici. Modeling morphologically rich languages using split words and unstructured dependencies. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 345–348, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-2087.