Adaptive Tuning for Statistical Machine Translation (AdapT)

Mohamed A. Zahran[1] and Ahmed Y. Tawfik[2]

[1] Computer Engineering Department, Cairo University, Egypt
[2] Microsoft Advanced Technology Lab, Cairo, Egypt
moh.a.zahran@gmail.com, atawfik@microsoft.com

Abstract. In statistical machine translation systems, it is a common practice to use one set of weighting parameters in scoring the candidate translations from a source language to a target language. In this paper, we challenge the assumption that only one set of weights is sufficient to pick the best candidate translation for all source language sentences. We propose a new technique that generates a different set of weights for each input sentence. Our technique outperforms the popular tuning algorithm MERT on different datasets using different language pairs.

Keywords: Statistical Machine Translation, Adaptive Tuning, Sentence Representation, Per-sentence Translation.

1 Introduction

Tuning statistical machine translation systems (SMT) is a crucial step that has a significant impact on the overall performance of the system. Tuning is the process of finding optimal weights used to pick the best translation among the generated candidate translations. These weights reflect the relative importance of the SMT building models such as language model, translation model, word penalty, distortion, and any other additional features affecting the quality of translation.

Minimum error rate training (MERT) [4] is the popular tuning algorithm for many statistical machine translation systems. Given a parallel corpus $\{F, E\}$ of source language sentences $F = \{f_1, f_2, f_3 \dots\}$ and target language sentences $E = \{e_1, e_2, e_3 \dots\}$, a typical phrasal SMT system undergoes three main steps: training, tuning and testing. The training phase uses the source language and its parallel target language sentences to learn phrase translations and compute translation probabilities to them. These translations from source to target are stored with their probabilities and some additional information in a phrase table. The translation task requires building a language model for the target language to favor the translations obeying the language structure of the target language. The language model can be built with the target language side of the parallel training data, or it can be built using any additional target language text. The tuning phase is concerned with generating candidate translations $(e_{i1}, e_{i2}, e_{i3} \ldots)$ for source language sentence f_i and picking the best candidate (e_{i^*}) as the final translation.

$$e_{i^*} = argmax_e\, S(e,f) \tag{1}$$

The scoring function $S(e,f)$ combines the conditional log likelihood probabilities in the translation model $TM(e|f)$ the language model $LM(e)$ score, a distortion model $D(f,e)$ score, and a word penalty $W(e)$ term. The distortion model controls the amount of reordering of the translated phrases to suite the target language requirements. Word penalty ensures that the translations do not get too long or too short.

$$\begin{aligned} S(e,f) &= \lambda_{LM} LM(e) + \lambda_{TM} TM(e|f) + \lambda_D D(f,e) + \lambda_W W(e) \\ &= \lambda \,.\, \Psi(e,f) \end{aligned} \tag{2}$$

The goal of tuning algorithms like MERT is to find a set of optimal weighting parameters $(\lambda_{LM}, \lambda_{TM}, \lambda_D, \lambda_W)$ to weight the four model listed in (2) to achieve the best translation accuracy measured against a reference translation (ê) using a measure such as the popular BLEU score [5] such that:

$$Maximize\text{: } S(e^*, f)\ if\ e^* is\ most\ similar\ to\ \hat{e} \tag{3}$$

Practically, $S(e,f)$ can be viewed as the inner product of the weighting parameter vector λ with the model vector Ψ. Let E_λ be the set of translations selected by the model parameterized by the weight vector λ. MERT's goal is finding an optimal weight vector λ^* that minimizes the loss function $L(E_\lambda)$:

$$L(E_\lambda) = 1 - BLEU(E_\lambda) \tag{4}$$

$$\lambda^* = argmin_\lambda\, L(E_\lambda) \tag{5}$$

MERT explores the parameter space using either Powell's method [12] or Koehn coordinate descent as adapted by Moses the statistical machine translation package [8]. MERT finds a series of sub-optimal points (weight vectors) during its search until no new BLEU gain is achieved or the changes in the weights are less than a certain threshold. MERT's objective function is a non-convex piece-wise constant [3]. Which means that at certain critical points, small changes in the weights will change the relative ranking between candidate translations. To visualize these critical points, we will consider two weights only as shown in Figures 1&2, these critical points are on the boundaries of the shaded regions.

If we have two source sentences f_1 and f_2 to translate. We will examine the effect of changing two weights only while holding the rest of weights constant on changing the relative order of the candidate translations of both $f_1$andf_2.

By examining Figure 1, if e_{11} is in fact the best translation for f_1,then the two weights (λ_1, λ_2) should be assigned values in the dotted region to make e_{11} the dominant candidate. Since the same weight values will be used in the translation of all source sentences in F, there is no guarantee that the best candidate translation of any other sentence will be in the dotted region of f_1 . In other words, if the candidate regions of f_2 is as shown in Figure 2, then one set of values for $\lambda_1$and λ_2 will not translate both f_1 and f_2 optimally, because the dotted regions of f_1 and f_2 do not overlap.

For one set of weights to translate all source sentences optimally, all the dotted regions for all source sentences must overlap, which is not a practical assumption as we explained. In the next sections, we propose new techniques to generate a set of weighting parameters to be used per input source sentence.

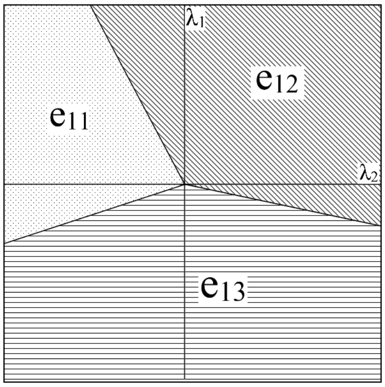

Fig. 1. Changing the relative order between three candidate translations (e_{11}, e_{12}, e_{13}) for the source sentence f_1 with the change of two weights only (λ_1, λ_2). Each region is labeled with its dominant candidate.

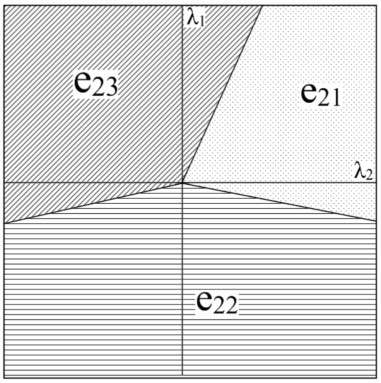

Fig. 2. Changing the relative order between three candidate translations (e_{21}, e_{22}, e_{23}) for the source sentence f_2 with the change of two weights only (λ_1, λ_2). Each region is labeled with its dominant candidate.

2 Related Work

While MERT is used broadly in many SMT systems, no research has been made –to the best of our knowledge- that discusses weight adaptation as presented here. Liu et al. [7] proposed a local training scheme, where the system retunes using a tailored tuning set for the input test sentence. A number of training sentences most similar to the input test sentence are appended to the default tuning set then retuning is performed, and finally the resulting weights are used to decode the test sentence.

Li et al. [6] presented an adaptive data selection, where given a test set, an iterative algorithm will select sentences from the tuning set most similar to the test set and using in tuning weights for this test set. Although our technique AdapT shares the same spirit as these two techniques, unlike them, our sentence specific weights are obtained without re-tuning. This is a major and important difference as re-tuning is a time consuming operation that cannot be done on the fly in real-time. Our methods ensure that decoding happens in real-time and the tuning phase happens only once.

There have been several attempts to enhance upon MERT directly, or enhancing SMT models. Hildebrand et al. [1] developed a method to adapt the translation model for the test set. For each test sentence, the corresponding top n similar sentences are selected from the training data. This selection results in a new subset of the training data used to build a translation model adapted to this particular test set. They represented the sentences as vectors using TF- IDF and used cosine the angle between vectors as the similarity measure between sentences. The amount of data to be selected from the train data per test sentence (n) is determined by minimizing the perplexity (PPL) of the language model built by the selected data.

Hildebrand and Vogel [2] introduced a scheme to leverage the individual strength of different machine translation systems. For a test sentence, they pool the N-best list of all machine translation systems together forming a joint N-best list. The best hypothesis is selected depending on the features scores. These features are based solely on the hypothesis without any prior knowledge of the corresponding machine translation systems. Linear combination weighting between features scores is optimized using MERT. To optimize MERT, Cer et al. [3] presented two alterations to MERT's search techniques. The first is introducing a new simple stochastic search strategy that outperformed Powell's method and coordinate descent. The second is presenting a regularization scheme for both Powell's method and coordinate descent that lead to performance gain.

3 Adaptive Tuning

Using hypothetical examples, we introduced that using one weight vector cannot guarantee choosing the best candidate for all source language sentences, so using different weight vectors in translating different source sentences can –in the best case scenario- translate all source sentences optimally. Optimal translation in this context is choosing the best candidate translation. In this paper, we introduce new methods to generate tailored weight vectors influenced by the input sentence so that the weights change adaptively according to this particular input sentence. Changing the weight vector from one sentence to another reveals the relative importance of the SMT models (LM, TM, D, W) in translating different sentences. The basic idea is to have a pool of weight vectors preferably diverse enough to explore different relative importance for the SMT models. Then using features from the input sentence we choose the appropriate weight vector from the pool specifically for this sentence. Our technique asks three main questions. First, how to formulate the weights pool. Second, how to represent the input sentence. Third, how to map a sentence to its suitable weight vector. We propose answers to these questions in the next sections.

3.1 Input Representation

The idea is to give similar sentences similar representation. Similarity of sentences can be measured in terms of use of words, topic, length … etc. In the literature term frequency-inverse document frequency (tf-idf) with cosine similarity were used in the context of machine translation to compare sentences. However, tf-idf vectors can be too sparse; instead, we used two techniques to represent input source language sentences. The first is using Latent Semantic Analysis (LSA) [10] which performs SVD over tf-idf bag of words representation for the sentences projecting them into lower dimensionality (200~300 dimensions generally works fine for LSA).

The second representation technique combines semantic projections of individual word representations to form a sentence representation. Several attempts have been made in the literature to generate a continuous vectorized representation (embeddings) for individual words for a given language. Mikolov et al. [11] proposed new technique for learning vectorized representation for individual words called continuous bag of words (CBOW). It is a neural network that predicts a word by using a context window from its history and future. The hidden layer is replaced with a projection layer into which the context words are projected by averaging their vector representations, thus decreasing the computational complexity.

We used the source language side of the parallel training data to learn the CBOW model. Then using the learnt word vectors we represented the source side language sentences of the development set by combining the individual words representation per sentence. A simple combining technique is just adding the words vectors together, but this will not highlight the relative importance of words in the final sentence representation, therefore we used tf-idf weighting as a measure to stress the impact of words over others.

The same data used to learn the CBOW is also used to build the tf-idf model so that the final representation of an N-words sentence is:

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{tfidf(w_\mathrm{i}) \; .* \; vec(w_\mathrm{i})}{\sum_{\mathrm{j}=1}^{\mathrm{N}} tfidf(w_\mathrm{j})} \tag{6}$$

Where $vec(w_\mathrm{i})$ is the CBOW vector representation for the word w_i of size M. The $tfidf(w_\mathrm{i})$ is a scalar weight for word w_i. The operator $.*$ is an element-wise multiplication operator for each element in the vector. The denominator is a normalizing factor. The result of equation (6) is the vectorized sentence representation of size M.

3.2 Weight Vector Pool

The core idea of our technique is that the impact of the SMT models varies from one sentence to another, these variations are reflected in the choice of weight vector for each sentence. Since eventually the vectors in the pool will be used as reference weights to translate sentences, they should satisfy two conditions: diversity and performance. The first method to fill the pool is via MERT. While MERT's goal is to find a single weight vector that maximizes the BLEU score for the whole development set, it also finds sets of sub-optimal weight vectors along its search for a single global optimum. These sub-optimal vectors can outperform the final optimal vector for individual sentences, which means that we can use these sub-optimal vectors together with the optimal vector in the weights pool. This technique treats the development set as a whole and finds a series of weight vectors performing globally well over the whole development set. We will refer to this technique as "**MERTprogress**".

Another idea is to divide the development set into clusters, and then we run MERT on each cluster so that the final optimal MERT weight vector for each cluster is used in the weights pool. We used kmeans clustering over the development set sentence representations. We will refer to this technique as "**MERTclusters**".

3.3 Mapping Sentences to Weight Vectors

When decoding a new input test sentence, there should be a method to map this sentence to an appropriate weight vector from the pool. Here we present two mapping techniques. The first technique is only applicable to MERTclusters. Using the clusters formed by MERTclusters on the development set sentences, we can uses the test sentence representation to assign it to one of these clusters and uses the assigned cluster's weight vector in its translation.

The second technique is applicable to both MERTprogress and MERTclusters. It is building a regression neural network with objective of mapping the representation of the development set sentences with their corresponding weight vectors. Typical regression neural networks minimize the mean square error (MSE) between outputs and reference values. For the problem at hand, we propose the objective function to maximize the cosine similarity between the predicted weights with the reference weights. The intuition behind this choice for the objective function is as shown in Figures 1 & 2 that the optimal candidate is top ranked when the weights allow the best candidate to dominate (the dotted region). This region extends to infinity, which means that the relative order of the candidate translations is scale invariant with respect to the weight values. Consider equation (2) if we multiply the whole equation by a constant α then the relative ranking of the candidates will remain unchanged.

$$S(e,f) = \alpha\,\lambda\,.\,\Psi(e,f) = \emptyset\,.\,\Psi(e,f) \qquad (7)$$
$$where\ \emptyset = \alpha\,\lambda$$

This property illustrates that any scaled version of the weight vector will not tamper with the relative order of the candidate translations, which means that it is not important how close in values the predicted weights are from the reference weights as long as they form a scaled version of the reference weights. This directly follows the intuition behind maximizing the cosine similarity between the predicted weights and the reference weights. We used back propagation as a learning algorithm for the neural network. (Check appendix). It is worth noting that minimizing the Cosine error between two normalized vectors is equivalent to minimizing half the square error between them in terms of the objective function.

For MERTclusters, choosing the appropriate weight vector to each sentence in the development set in order to train the neural network is straight forward; the neural network will be trained to map sentences representations with their corresponding cluster weight vector.

On the other hand, for MERTprogress, choosing the best weight vector per sentence requires some calculations. We used all the MERTprogress weight vectors to translate the development set, then for each sentence in the development set we choose which weight vector with the best translation for this particular sentence. The best translation is one with the highest BLEU score. Calculating the BLEU score per sentence usually equals zero. This happens when one of the n-gram scores is zero. To avoid this problem we use equation 8 as a per sentence gauge of translation quality (s) inspired by BLEU.

$$s = BP \sum_{\mathrm{i}=1}^{\mathrm{n}} b_{\mathrm{i}} \tag{8}$$

Where s is the score, BP is the brevity penalty, b_{i} are the i-gram score. When MERT converges after k iterations it finds k points[1] in the weight space one after each iteration. Earlier vectors tend to be associated with lower BP value. Let $P = \{p_1, p_2 \ldots p_k\}$ be the set of points ordered by MERT iterations, and $S = \{s_1, s_2 \ldots s_k\}$ are the scores computed for the translation of a given sentence using the weights in set P respectively. The set of points that correspond to best scoring translations is:

$$P^* = argmax_{\mathrm{i}}\, s_{\mathrm{i}} \tag{9}$$

If P^* is a singleton, i.e. there is only one point that gives, the best score, then the corresponding point is optimal for this sentence. If not singleton, i.e. if there is a tie, selecting one of the candidates affects the results, and in the experiments we considered two tie resolution mechanisms: the first favors the first or earliest point in P^* (lower index) that produces the best score. The other tie resolution strategy favors the latest point (higher index) in the set P^* that produces the best score. Experiments show, tie resolution, either early or late, affects the performance of the system.

[1] A point in the space is a vector of weights.

4 Phrasal SMT Systems Using AdapT

Using AdapT as tuning algorithm for SMT systems, we will follow the same steps as normal SMT training with few exceptions. The first is the need to build representations for the source language (LSA/CBOW), which can be built using the source language side of the parallel training data or any other source language data. The second difference is the need to have a relatively larger development set, because small development set will not be enough to fit the mapping neural network, and it will not be enough to give neither adequate number of clusters nor representative cluster members.

If the development data is small it will be essential to remove parallel sentences from the training set to be added to the development set, an adequate development set in the order of 10,000 to 20,000 sentences.

We performed a number of experiments to utilize both sentence representation schemes (LSA & CBOW), both weight vectors pooling schemes (MERTprogress & MERTclusters) and finally both mapping schemes (Neural Network & Clustering).

The first experiment using LSA representation with MERTprogress pooling technique and the regression neural network as the mapping scheme (LSA_MERT progress_NN). The second experiment is using the CBOW representations with MERTclusters and clustering to map representation to suitable weight vectors (CBOW_MERTclusters_clustering). The third experiment is using the CBOW representations with MERTclusters and the regression neural network as the mapping scheme (CBOW_MERTclusters_NN). We choose those three experiments to report their results exhaustively. Other possible experiments were carried out using different combinations, but their initial results were redundant. For example, one possible experiment is (CBOW_MERT progress_NN) has almost the same results as (LSA_MERT progress_NN). Testing the SMT system after tuning using one of the pervious settings is shown in Figure 3.

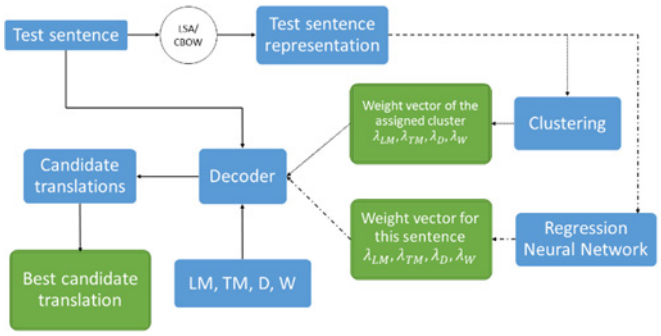
Test sentence
LSA/ CBOW
Test sentence representation
Weight vector of the assigned cluster $\lambda_{LM}, \lambda_{TM}, \lambda_D, \lambda_W$
Clustering
Candidate translations
Decoder
Weight vector for this sentence $\lambda_{LM}, \lambda_{TM}, \lambda_D, \lambda_W$
Regression Neural Network
Best candidate translation
LM, TM, D, W

Fig. 3. The decoding of a test sentence using either the clustering or the neural network mapping techniques

5 Results and Evaluation

We applied AdapT in comparison with MERT on different language pairs French-English, Spanish-English, and German-English using different datasets. We used the European Parliament Proceeding Parallel Corpus (europarl-v7) as the training set[2] and used the target side of the parallel data to build the language model and the source side to build the sentence representations for either LSA or CBOW models.

We used the news test 2008 as the development set. Standard development sets are usually small around 3000 sentences, so we removed 15000 sentences from the training set and appended them to the default development set. We build the LSA model using 300 topics to represent the sentence using gensim [9]. On the other hand, CBOW model requires large data to be built accurately, since we intended not to use external data to the WMT datasets, that left us to use the source language side of the training set (around 1.5~2 million lines). This relatively small dataset suggested to use smaller vector size to be learnt by CBOW. Thus, we used vectors of size 200. For MERTclusters pooling technique we used kmeans with k=50 clusters over the development set. Too few clusters will result in big non-homogeneous clusters, which can set back exploiting enough weight vectors variations. On the other hand, too many clusters will result in sparse clusters.

We manually examined the clusters of the development set for the Parliament Spanish-English dataset. Sentences in each cluster share some characteristics as semantic scope, topic and domain (Economics, Energy, Money, Middle-east and conflicts), sentence structure (use of articles, commas, dots and quotations), use of words (named entities, numbers and quantifications), length (short questions and long questions) and type (questions, objections, opening statements, closing statements, question numbers, lists, thank-you sentences, applause and conclusion). Next, in Table 1 we show sample clusters with representative sentences:

Table 1. Sentences and their clusters for the English-Spanish development set

Short Questions:
Why not?
Whose turn is next?
Any comments?
Can you?
Question Numbers:
question No 28 by (H-0781 / 99).
question No 29 by (H-0786 / 99).
Opening statements:
I would like , first of all , to thank the rapporteur for his exceptionally accurate …
Mr President , Commissioners , first of all , I cannot help but reflect upon …
I call upon you , ladies and gentlemen , to vote in favour of this report …
Closing statements:
the debate is closed.
that concludes Question Time.
the vote will take place tomorrow at 12 p.m.

[2] Data available at: `http://www.statmt.org/wmt14/`

Table 1. (*Continued*)

Energy:
we need real cost-effectiveness for our entire energy supply system.
we must reduce CO2 emissions , employ renewable energies , and generally make …
promoting the use of renewables is especially important for the environment.
Conclusion:
Parliament approved the Commission proposal.
the President declared the common position approved (as amended).
Parliament rejected the proposal.
Money:
a special provision of up to EUR 50 million for Greece.
Kyrgyzstan received EUR 17 million.
between EUR 1500 and 2000 billion are traded every day on the financial markets.
Thank-you:
thank you, Mr Poettering.
thank you very much.
thank you, Commissioner, for your statement.
I thank him for that.

Table 2. BLEU scores for MERT and AdapT for different language pairs on different test sets

Test set	MERT	LSA_MERT progress_NN _Early	LSA_MERT progress_NN _Late	CBOW_ MERT clusters_NN	CBOW_MERT clusters_ Clustering
fr-en 2010	22.07	22.31 (+0.24)	22.08 (+0.01)	**22.37 (+0.3)**	22.22 (+0.15)
fr-en 2011	23.07	23.30 (+0.23)	23.10 (+0.03)	23.52 (+0.45)	**23.55 (+0.48)**
es-en 2010	23.73	23.95 (+0.22)	23.97 (+0.24)	**24.25 (+0.52)**	24.05 (+0.32)
es-en 2011	23.25	23.43 (+0.18)	23.54 (+0.29)	23.69 (+0.44)	**23.78 (+0.53)**
de-en 2010	17.00	16.83 (-0.17)	17.25 (+0.25)	17.34 (+0.34)	**17.53 (+0.53)**
de-en 2011	15.82	15.80 (-0.02)	16.17 (+0.35)	16.09 (+0.27)	**16.50 (+0.68)**

Table 2 shows the BLEU score for different data sets. We can notice the loss in BLEU score for the “LSA_MERTprogress_NN_Early” in the last two test sets due to the low BP associated with the “early” configuration. On the other hand, “CBOW_MERTclusters_Clustering” shows the best performance in most of the test sets, which suggests that the SMT systems can leverage information from multiple domains by considering each cluster as a separate domain and classifying the new test sentence to one of the domains. Table 3 shows the statistical significance for our results against MERT using bootstrap resampling techniques [13]. The test aims to estimate the degree to which the true translations quality lies within a certain confidence interval (q) around the measurement on test sets. The commonly used confidence interval is 95%. For a test set of m BLEU score, this test finds an interval $[m - d, m + d]$ in which the true BLEU score lies with probability $q = 0.95$. This test shows that AdapT performs better than MERT with a 95% confidence, and indicates that they are two independent systems as evidenced by the P-value.

Table 3. Statistical significance results for AdapT in comparison to MERT against different datasets at $q = 0.95$, subsampling size equals to the whole test set, and repeating the subsampling for 1000 times. The P-value is the probability that both MERT and AdapT translations are generated from the same system.

Test set		**LSA_MERTprog ress_NN Late**	**CBOW_MERT clusters_NN**	**CBOW_MERT clusters_Clustering**
fr-en 2010	MERT	21.3645 +/- 0.5976	21.3885+/-0.5756	21.3556+/-0.5976
	AdapT	21.3671 +/- 0.6041	21.6333+/-0.5802	21.4953+/-0.5755
	P-value	0.391	0.013	0.11
fr-en 2011	MERT	22.1985 +/- 0.5763	22.2038+/-0.5730	22.2299+/0.55699
	AdapT	22.25489+/-0.5754	22.6409+/-0.5881	22.7129+/-0.5706
	P-value	0.168	≈ 0	≈ 0
es-en 2010	MERT	23.1678 +/- 0.6198	23.1409+/-0.5969	23.1546+/-0.6275
	AdapT	23.3848 +/- 0.6238	23.4080+/-0.6140	23.3988+/-0.6010
	P-value	≈ 0	0.01	0.008
es-en 2011	MERT	22.5934 +/-0.5863	22.5888+/-0.5798	22.61975+/0.5818
	AdapT	22.8795 +/- 0.5879	23.0219 +/- 0.577	23.135 +/- 0.5988
	P-value	≈ 0	≈ 0	≈ 0
de-en 2010	MERT	16.3774 +/- 0.5064	16.3504 +/- 0.479	16.3605+/-0.4864
	AdapT	16.5735 +/- 0.494	16.5996+/-0.5202	16.8447+/-0.4822
	P-value	0.001	0.045	≈ 0
de-en 2011	MERT	15.2275 +/- 0.446	15.2069+/-0.4389	15.2263+/-0.4457
	AdapT	15.5275 +/- 0.4473	15.4073+/-0.4965	15.7768+/-0.4525
	P-value	≈ 0	0.068	≈ 0

6 Conclusion

In this paper, we showed the limitations of using one set of weighting parameters in the SMT systems. We presented a number of experiments to choose weight vectors adaptively according to the input sentence. Our preliminary results show that AdapT is a promising approach that has outperformed standard MERT on different language pairs using different datasets. The results of our analysis suggest that there still more room for improvements. We believe that AdapT can influence future research to target the area of adaptive tuning. One possible extension is using AdapT with other tuning algorithm like MIRA or PRO.

References

1. Hildebrand, A., Eck, M., Vogel, S., Waibel, A.: Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval. In: EAMT: Proceedings of the Tenth, European Association for Machine Translation in Budapest, Hungary, May 30-31, pp. 133–142 (2005)
2. Hildebrand, A., Vogel, S.: Combination of Machine Translation Systems via Hypothesis Selection from Combined N-Best Lists. In: AMTA: Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas, Hawaii, pp. 254–261 (October 2008)

3. Cer, D., Jurafsky, D., Manning, C.: Regularization and Search for Minimum Error Rate Training. In: WMT: Proceedings of the Third Workshop on Statistical Machine Translation, Columbus, Ohio, USA, pp. 26–34 (June 2008)
4. Och, F.: Minimum Error Rate Training in Statistical Machine Translation. In: ACL: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 160–167 (2003)
5. Papineni, K., Roukos, S., Ward, T., Zhu, W.: BLEU a Method for Automatic Evaluation of Machine Translation. In: ACL: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, 311–318 (July 2002)
6. Li, M., Zhao, Y., Zhang, D., Zhou, M.: Adaptive Development Data Selection for Log-linear Model in Statistical Machine Translation. In: COLING: Proceedings of the 23rd International Conference on Computational Linguistics, Beijing, pp. 662–670 (August 2010)
7. Liu, L., Cao, H., Watanabe, T., Zhao, T., Yu, M., Zhu, C.: Locally Training the Log-Linear Model for SMT. In: EMNLP: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, pp. 402–411 (July 2012)
8. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open Source Toolkit for Statistical Machine Translation. In: ACL: Proceedings of the Association for Computational Linguistics Demo and Poster Sessions, pp. 177–180 (2007)
9. Rehurek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: LREC: Proceedings of the Language Resources and Evaluation Conference workshop on new challenges for NLP Frameworks, Valletta, Malta, pp. 45–50 (May 2010)
10. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science, 391–407 (1990)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Distributed Representation of Words and Phrases and their Compositionality. In (NIPS): Proceedings of Neural Information Processing Systems, Nevada, United States (2013)
12. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical recipes 3rd edition: The art of scientific computing. Cambridge University Press (2007)
13. Koehn, P.: Statistical Significance Tests for Machine Translation Evaluation. In: EMNLP: Proceedings of Empirical Methods in Natural Language Processing, pp. 388–395 (2004)

Appendix

The objective function is to maximize the cosine similarity between the predicted vector (y) and the reference vector (d). This is equivalent to:

$$Minimize\ E = 1 - \cos(y, d) = 1 - \frac{y.d}{|y||d|}$$

Let the activation function be $y_{\mathrm{i}}^{\mathrm{z}} = F(x_{\mathrm{i}}^{\mathrm{z}})$. The superscript denotes the layer number, and the subscript denotes the input number. The notation $l(z)$ refers to the number of neurons in the layer z. The derivative of the error function E with respect to weights at layer z for the training sample m:

$$\frac{\partial E}{\partial w_{\mathrm{IJ}}^{\mathrm{z}}} = \frac{\partial E}{\partial y_{\mathrm{I}}^{\mathrm{z+1}}} \times \frac{\partial y_{I}^{\mathrm{z+1}}}{\partial x_{\mathrm{I}}^{\mathrm{z+1}}} \times \frac{\partial x_{\mathrm{I}}^{\mathrm{z+1}}}{\partial w_{\mathrm{IJ}}^{\mathrm{z}}} = \delta_{\mathrm{I}}^{\mathrm{z+1}} \times \frac{\partial x_{\mathrm{I}}^{\mathrm{z+1}}}{\partial w_{\mathrm{IJ}}^{\mathrm{z}}} \tag{10}$$

$$where, \quad \delta_{\mathrm{I}}^{\mathrm{z+1}} = \frac{\partial E}{\partial y_{\mathrm{I}}^{\mathrm{z+1}}} \times \frac{\partial y_{\mathrm{I}}^{\mathrm{z+1}}}{\partial x_{\mathrm{I}}^{\mathrm{z+1}}} \tag{11}$$

$$\frac{\partial E}{\partial y_{\mathrm{I}}^{\mathrm{z+1}}} = \frac{(y.d)y_{\mathrm{I}}^{\mathrm{z+1}} - d_{\mathrm{I}}^{\mathrm{z+1}}|y|^2}{|d||y|^3} \tag{12}$$

$$y_{\mathrm{I}}^{\mathrm{z+1}} = f(x_{\mathrm{I}}^{\mathrm{z+1}})$$

$$f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right)$$

$$\frac{\partial y_{\mathrm{I}}^{\mathrm{z+1}}}{\partial x_{\mathrm{I}}^{\mathrm{z+1}}} = f'(x_{\mathrm{I}}^{\mathrm{z+1}}) = \left(1.7159 \times \frac{2}{3}\right)(1 - \left(\frac{y_{\mathrm{I}}^{\mathrm{z+1}}}{1.7159}\right)^2) \tag{13}$$

$$x_{\mathrm{I}}^{\mathrm{z+1}} = \sum_{\mathrm{j=1}}^{l(\mathrm{z})} w_{\mathrm{Ij}}^{\mathrm{z}}\, y_{\mathrm{j}}^{\mathrm{z}} \quad \Rightarrow \quad \frac{\partial x_{\mathrm{I}}^{\mathrm{z+1}}}{\partial w_{\mathrm{IJ}}^{\mathrm{z}}} = y_{\mathrm{J}}^{\mathrm{z}} \tag{14}$$

Finally $\frac{\partial E}{\partial w_{\mathrm{IJ}}^{\mathrm{z}}}$ is calculated by substituting from (11), (12), (13) and (14) in (10).

Now we will prove that optimizing for a training sample for, Cosine error is equivalent for half the square error (SE) if the both the reference (d) and the predicted vector (y) are normalized.

$$|y| = 1 \; and \; |d| = 1$$

$$E_{\mathrm{COS}} = 1 - \cos(y, d) = 1 - y.d$$

$$E_{\mathrm{SE}} = \sum_{\mathrm{i=i}}^{\mathrm{K}} (d_{\mathrm{i}} - y_{\mathrm{i}})^2 = (d - y).(d - y)$$

$$= d.d - d.y - y.d + y.y$$

$$= |d|^2 - 2d.y + |y|^2$$

$$= 2 - 2\cos(y, d)$$

$$\frac{1}{2} E_{\mathrm{SE}} = 1 - \cos(y, d) = E_{\mathrm{COS}}$$