Bridging SMT and TM with Translation Recommendation

Yifan He **Yanjun Ma** **Josef van Genabith** **Andy Way**
Centre for Next Generation Localisation
School of Computing
Dublin City University
{yhe,yma,josef,away}@computing.dcu.ie

Abstract

We propose a translation recommendation framework to integrate Statistical Machine Translation (SMT) output with Translation Memory (TM) systems. The framework recommends SMT outputs to a TM user when it predicts that SMT outputs are more suitable for post-editing than the hits provided by the TM. We describe an implementation of this framework using an SVM binary classifier. We exploit methods to fine-tune the classifier and investigate a variety of features of different types. We rely on automatic MT evaluation metrics to approximate human judgements in our experiments. Experimental results show that our system can achieve 0.85 precision at 0.89 recall, excluding exact matches. Furthermore, it is possible for the end-user to achieve a desired balance between precision and recall by adjusting confidence levels.

1 Introduction

Recent years have witnessed rapid developments in statistical machine translation (SMT), with considerable improvements in translation quality. For certain language pairs and applications, automated translations are now beginning to be considered acceptable, especially in domains where abundant parallel corpora exist.

However, these advances are being adopted only slowly and somewhat reluctantly in professional localization and post-editing environments. Post-editors have long relied on translation memories (TMs) as the main technology assisting translation, and are understandably reluctant to give them up. There are several simple reasons for this: 1) TMs are useful; 2) TMs represent considerable effort and investment by a company or (even more so) an individual translator; 3) the fuzzy match score used in TMs offers a good approximation of post-editing effort, which is useful both for translators and translation cost estimation and, 4) current SMT translation confidence estimation measures are not as robust as TM fuzzy match scores and professional translators are thus not ready to replace fuzzy match scores with SMT internal quality measures.

There has been some research to address this issue, see e.g. (Specia et al., 2009a) and (Specia et al., 2009b). However, to date most of the research has focused on better confidence measures for MT, e.g. based on training regression models to perform confidence estimation on scores assigned by post-editors (cf. Section 2).

In this paper, we try to address the problem from a different perspective. Given that most post-editing work is (still) based on TM output, we propose to recommend MT outputs which are better than TM hits to post-editors. In this framework, post-editors still work with the TM while benefiting from (better) SMT outputs; the assets in TMs are not wasted and TM fuzzy match scores can still be used to estimate (the upper bound of) post-editing labor.

There are three specific goals we need to achieve within this framework. Firstly, the recommendation should have high precision, otherwise it would be confusing for post-editors and may negatively affect the lower bound of the post-editing effort. Secondly, although we have full access to the SMT system used in this paper, our method should be able to generalize to cases where SMT is treated as a black-box, which is of-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 622–630,
Uppsala, Sweden, 11-16 July 2010.

ten the case in the translation industry. Finally, post-editors should be able to easily adjust the recommendation threshold to particular requirements without having to retrain the model.

In our framework, we recast translation recommendation as a binary classification (rather than regression) problem using SVMs, perform RBF kernel parameter optimization, employ posterior probability-based confidence estimation to support user-based tuning for precision and recall, experiment with feature sets involving MT-, TM- and system-independent features, and use automatic MT evaluation metrics to simulate post-editing effort.

The rest of the paper is organized as follows: we first briefly introduce related research in Section 2, and review the classification SVMs in Section 3. We formulate the classification model in Section 4 and present experiments in Section 5. In Section 6, we analyze the post-editing effort approximated by the TER metric (Snover et al., 2006). Section 7 concludes the paper and points out avenues for future research.

2 Related Work

Previous research relating to this work mainly focuses on predicting the MT quality.

The first strand is confidence estimation for MT, initiated by (Ueffing et al., 2003), in which posterior probabilities on the word graph or N-best list are used to estimate the quality of MT outputs. The idea is explored more comprehensively in (Blatz et al., 2004). These estimations are often used to rerank the MT output and to optimize it directly. Extensions of this strand are presented in (Quirk, 2004) and (Ueffing and Ney, 2005). The former experimented with confidence estimation with several different learning algorithms; the latter uses word-level confidence measures to determine whether a particular translation choice should be accepted or rejected in an interactive translation system.

The second strand of research focuses on combining TM information with an SMT system, so that the SMT system can produce better target language output when there is an exact or close match in the TM (Simard and Isabelle, 2009). This line of research is shown to help the performance of MT, but is less relevant to our task in this paper.

A third strand of research tries to incorporate confidence measures into a post-editing environment. To the best of our knowledge, the first paper in this area is (Specia et al., 2009a). Instead of modeling on translation quality (often measured by automatic evaluation scores), this research uses regression on both the automatic scores and scores assigned by post-editors. The method is improved in (Specia et al., 2009b), which applies Inductive Confidence Machines and a larger set of features to model post-editors' judgement of the translation quality between 'good' and 'bad', or among three levels of post-editing effort.

Our research is more similar in spirit to the third strand. However, we use outputs and features from the TM explicitly; therefore instead of having to solve a regression problem, we only have to solve a much easier binary prediction problem which can be integrated into TMs in a straightforward manner. Because of this, the precision and recall scores reported in this paper are not directly comparable to those in (Specia et al., 2009b) as the latter are computed on a pure SMT system without a TM in the background.

3 Support Vector Machines for Translation Quality Estimation

SVMs (Cortes and Vapnik, 1995) are binary classifiers that classify an input instance based on decision rules which minimize the regularized error function in (1):

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\limits_{i=1}^{l}\xi_i \\ \text{s. t.} & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{array} \quad (1)$$

where $(\mathbf{x}_i, y_i) \in R^n \times \{+1, -1\}$ are l training instances that are mapped by the function ϕ to a higher dimensional space. $\mathbf{w}$ is the weight vector, ξ is the relaxation variable and $C > 0$ is the penalty parameter.

Solving SVMs is viable using the 'kernel trick': finding a kernel function K in (1) with $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. We perform our experiments with the Radial Basis Function (RBF) kernel, as in (2):

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \quad (2)$$

When using SVMs with the RBF kernel, we have two free parameters to tune on: the cost parameter C in (1) and the radius parameter γ in (2).

In each of our experimental settings, the parameters C and γ are optimized by a brute-force grid

search. The classification result of each set of parameters is evaluated by cross validation on the training set.

4 Translation Recommendation as Binary Classification

We use an SVM binary classifier to predict the relative quality of the SMT output to make a recommendation. The SVM classifier uses features from the SMT system, the TM and additional linguistic features to estimate whether the SMT output is better than the hit from the TM.

4.1 Problem Formulation

As we treat translation recommendation as a binary classification problem, we have a pair of outputs from TM and MT for each sentence. Ideally the classifier will recommend the output that needs less post-editing effort. As large-scale annotated data is not yet available for this task, we use automatic TER scores (Snover et al., 2006) as the measure for the required post-editing effort. In the future, we hope to train our system on HTER (TER with human targeted references) scores (Snover et al., 2006) once the necessary human annotations are in place. In the meantime we use TER, as TER is shown to have high correlation with HTER.

We label the training examples as in (3):

$$y = \begin{cases} +1 & \text{if } TER(\text{MT}) < TER(\text{TM}) \\ -1 & \text{if } TER(\text{MT}) \geq TER(\text{TM}) \end{cases} \quad (3)$$

Each instance is associated with a set of features from both the MT and TM outputs, which are discussed in more detail in Section 4.3.

4.2 Recommendation Confidence Estimation

In classical settings involving SVMs, confidence levels are represented as margins of binary predictions. However, these margins provide little insight for our application because the numbers are only meaningful when compared to each other. What is more preferable is a probabilistic confidence score (e.g. 90% confidence) which is better understood by post-editors and translators.

We use the techniques proposed by (Platt, 1999) and improved by (Lin et al., 2007) to obtain the posterior probability of a classification, which is used as the confidence score in our system.

Platt's method estimates the posterior probability with a sigmod function, as in (4):

$$Pr(y = 1|\mathbf{x}) \approx P_{A,B}(f) \equiv \frac{1}{1 + exp(Af + B)} \quad (4)$$

where $f = f(\mathbf{x})$ is the decision function of the estimated SVM. A and B are parameters that minimize the cross-entropy error function F on the training data, as in Eq. (5):

$$\min_{z=(A,B)} F(z) = -\sum_{i=1}^{l}(t_i log(p_i) + (1 - t_i) log(1 - p_i)),$$
$$\text{where } p_i = P_{A,B}(f_i), \text{and } t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} \text{ if } y_i = +1 \\ \frac{1}{N_- + 2} \text{ if } y_i = -1 \end{cases} \quad (5)$$

where $z = (A, B)$ is a parameter setting, and N_+ and N_- are the numbers of observed positive and negative examples, respectively, for the label y_i. These numbers are obtained using an internal cross-validation on the training set.

4.3 The Feature Set

We use three types of features in classification: the MT system features, the TM feature and system-independent features.

4.3.1 The MT System Features

These features include those typically used in SMT, namely the phrase-translation model scores, the language model probability, the distance-based reordering score, the lexicalized reordering model scores, and the word penalty.

4.3.2 The TM Feature

The TM feature is the fuzzy match (Sikes, 2007) cost of the TM hit. The calculation of fuzzy match score itself is one of the core technologies in TM systems and varies among different vendors. We compute fuzzy match cost as the minimum Edit Distance (Levenshtein, 1966) between the source and TM entry, normalized by the length of the source as in (6), as most of the current implementations are based on edit distance while allowing some additional flexible matching.

$$h_{fm}(t) = \min_{e} \frac{EditDistance(s, e)}{Len(s)} \quad (6)$$

where s is the source side of t, the sentence to translate, and e is the source side of an entry in the TM. For fuzzy match scores F, this fuzzy match cost h_{fm} roughly corresponds to $1-F$. The difference in calculation does not influence classification, and allows direct comparison between a pure TM system and a translation recommendation system in Section 5.4.2.

4.3.3 System-Independent Features

We use several features that are independent of the translation system, which are useful when a third-party translation service is used or the MT system is simply treated as a black-box. These features are source and target side LM scores, pseudo source fuzzy match scores and IBM model 1 scores.

Source-Side Language Model Score and Perplexity. We compute the language model (LM) score and perplexity of the input source sentence on a LM trained on the source-side training data of the SMT system. The inputs that have lower perplexity or higher LM score are more similar to the dataset on which the SMT system is built.

Target-Side Language Model Perplexity. We compute the LM probability and perplexity of the target side as a measure of fluency. Language model perplexity of the MT outputs are calculated, and LM probability is already part of the MT systems scores. LM scores on TM outputs are also computed, though they are not as informative as scores on the MT side, since TM outputs should be grammatically perfect.

The Pseudo-Source Fuzzy Match Score. We translate the output back to obtain a pseudo source sentence. We compute the fuzzy match score between the original source sentence and this pseudo-source. If the MT/TM system performs well enough, these two sentences should be the same or very similar. Therefore, the fuzzy match score here gives an estimation of the confidence level of the output. We compute this score for both the MT output and the TM hit.

The IBM Model 1 Score. The fuzzy match score does not measure whether the hit could be a correct translation, i.e. it does not take into account the correspondence between the source and target, but rather only the source-side information. For the TM hit, the IBM Model 1 score (Brown et al., 1993) serves as a rough estimation of how good a translation it is on the word level; for the MT output, on the other hand, it is a black-box feature to estimate translation quality when the information from the translation model is not available. We compute bidirectional (source-to-target and target-to-source) model 1 scores on both TM and MT outputs.

5 Experiments

5.1 Experimental Settings

Our raw data set is an English–French translation memory with technical translation from Symantec, consisting of 51K sentence pairs. We randomly selected 43K to train an SMT system and translated the English side of the remaining 8K sentence pairs. The average sentence length of the training set is 13.5 words and the size of the training set is comparable to the (larger) TMs used in the industry. Note that we remove the exact matches in the TM from our dataset, because exact matches will be reused and not presented to the post-editor in a typical TM setting.

As for the SMT system, we use a standard log-linear PB-SMT model (Och and Ney, 2002): GIZA++ implementation of IBM word alignment model 4,[1] the refinement and phrase-extraction heuristics described in (Koehn et al., 2003), minimum-error-rate training (Och, 2003), a 5-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995) trained with SRILM (Stolcke, 2002) on the English side of the training data, and Moses (Koehn et al., 2007) to decode. We train a system in the opposite direction using the same data to produce the pseudo-source sentences.

We train the SVM classifier using the libSVM (Chang and Lin, 2001) toolkit. The SVM-training and testing is performed on the remaining 8K sentences with 4-fold cross validation. We also report 95% confidence intervals.

The SVM hyper-parameters are tuned using the training data of the first fold in the 4-fold cross validation via a brute force grid search. More specifically, for parameter C in (1) we search in the range $[2^{-5}, 2^{15}]$, and for parameter γ (2) we search in the range $[2^{-15}, 2^{3}]$. The step size is 2 on the exponent.

5.2 The Evaluation Metrics

We measure the quality of the classification by precision and recall. Let A be the set of recommended MT outputs, and B be the set of MT outputs that have lower TER than TM hits. We standardly define precision P, recall R and F-value as in (7):

[1] More specifically, we performed 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

$$P = \frac{|A \bigcap B|}{|A|}, R = \frac{|A \bigcap B|}{|B|} \text{ and } F = \frac{2PR}{P+R} \quad (7)$$

5.3 Recommendation Results

In Table 1, we report recommendation performance using MT and TM system features (SYS), system features plus system-independent features (ALL:SYS+SI), and system-independent features only (SI).

Table 1: Recommendation Results

	Precision	Recall	F-Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
SI	82.56±1.46	95.83±0.52	88.70±.65
ALL	83.45±1.33	95.56±1.33	89.09±.24

From Table 1, we observe that MT and TM system-internal features are very useful for producing a stable (as indicated by the smaller confidence interval) recommendation system (SYS). Interestingly, only using some simple system-external features as described in Section 4.3.3 can also yield a system with reasonably good performance (SI). We expect that the performance can be further boosted by adding more syntactic and semantic features. Combining all the system-internal and -external features leads to limited gains in Precision and F-score compared to using only system-internal features (SYS) only. This indicates that at the default confidence level, current system-external (resp. system-internal) features can only play a limited role in informing the system when current system-internal (resp. system-external) features are available. We show in Section 5.4.2 that combing both system-internal and -external features can yield higher, more stable precision when adjusting the confidence levels of the classifier. Additionally, the performance of system SI is promising given the fact that we are using only a limited number of simple features, which demonstrates a good prospect of applying our recommendation system to MT systems where we do not have access to their internal features.

5.4 Further Improving Recommendation Precision

Table 1 shows that classification recall is very high, which suggests that precision can still be improved, even though the F-score is not low. Considering that TM is the dominant technology used by post-editors, a recommendation to replace the hit from the TM would require more confidence, i.e. higher precision. Ideally our aim is to obtain a level of 0.9 precision at the cost of some recall, if necessary. We propose two methods to achieve this goal.

5.4.1 Classifier Margins

We experiment with different margins on the training data to tune precision and recall in order to obtain a desired balance. In the basic case, the training example would be marked as in (3). If we label both the training and test sets with this rule, the accuracy of the prediction will be maximized.

We try to achieve higher precision by enforcing a larger bias towards negative examples in the training set so that some borderline positive instances would actually be labeled as negative, and the classifier would have higher precision in the prediction stage as in (8).

$$y = \begin{cases} +1 & \text{if } TER(\text{SMT}) + b < TER(\text{TM}) \\ -1 & \text{if } TER(\text{SMT}) + b \geqslant TER(\text{TM}) \end{cases} \quad (8)$$

We experiment with b in [0, 0.25] using MT system features and TM features. Results are reported in Table 2.

Table 2: Classifier margins

	Precision	Recall
TER+0	83.45±1.33	95.56±1.33
TER+0.05	82.41±1.23	94.41±1.01
TER+0.10	84.53±0.98	88.81±0.89
TER+0.15	85.24±0.91	87.08±2.38
TER+0.20	87.59±0.57	75.86±2.70
TER+0.25	89.29±0.93	66.67±2.53

The highest accuracy and F-value is achieved by $TER + 0$, as all other settings are trained on biased margins. Except for a small drop in $TER{+}0.05$, other configurations all obtain higher precision than $TER + 0$. We note that we can obtain 0.85 precision without a big sacrifice in recall with b=0.15, but for larger improvements on precision, recall will drop more rapidly.

When we use b beyond 0.25, the margin becomes less reliable, as the number of positive examples becomes too small. In particular, this causes the SVM parameters we tune on in the first fold to become less applicable to the other folds. This is one limitation of using biased margins to

obtain high precision. The method presented in Section 5.4.2 is less influenced by this limitation.

5.4.2 Adjusting Confidence Levels

An alternative to using a biased margin is to output a confidence score during prediction and to threshold on the confidence score. It is also possible to add this method to the SVM model trained with a biased margin.

We use the SVM confidence estimation techniques in Section 4.2 to obtain the confidence level of the recommendation, and change the confidence threshold for recommendation when necessary. This also allows us to compare directly against a simple baseline inspired by TM users. In a TM environment, some users simply ignore TM hits below a certain fuzzy match score F (usually from 0.7 to 0.8). This fuzzy match score reflects the confidence of recommending the TM hits. To obtain the confidence of recommending an SMT output, our baseline (FM) uses fuzzy match costs $h_{FM} \approx 1-F$ (cf. Section 4.3.2) for the TM hits as the level of confidence. In other words, the higher the fuzzy match cost of the TM hit is (lower fuzzy match score), the higher the confidence of recommending the SMT output. We compare this baseline with the three settings in Section 5.

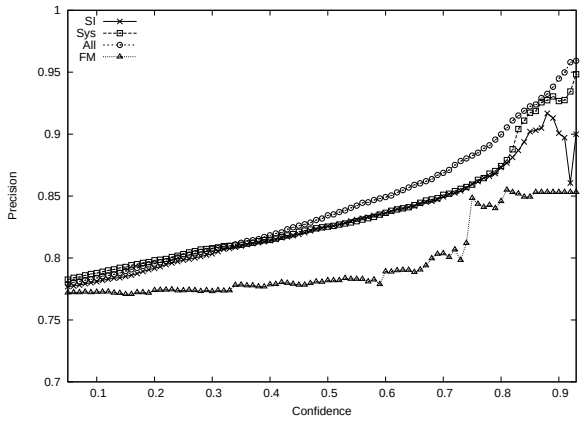

Figure 1: Precision Changes with Confidence Level

Figure 1 shows that the precision curve of FM is low and flat when the fuzzy match costs are low (from 0 to 0.6), indicating that it is unwise to recommend an SMT output when the TM hit has a low fuzzy match cost (corresponding to higher fuzzy match score, from 0.4 to 1). We also observe that the precision of the recommendation receives a boost when the fuzzy match costs for the TM hits are above 0.7 (fuzzy match score lower than 0.3), indicating that SMT output should be recommended when the TM hit has a high fuzzy match cost (low fuzzy match score). With this boost, the precision of the baseline system can reach 0.85, demonstrating that a proper thresholding of fuzzy match scores can be used effectively to discriminate the recommendation of the TM hit from the recommendation of the SMT output.

However, using the TM information only does not always find the easiest-to-edit translation. For example, an excellent SMT output should be recommended even if there exists a good TM hit (e.g. fuzzy match score is 0.7 or more). On the other hand, a misleading SMT output should not be recommended if there exists a poor but useful TM match (e.g. fuzzy match score is 0.2).

Our system is able to tackle these complications as it incorporates features from the MT and the TM systems simultaneously. Figure 1 shows that both the SYS and the ALL setting consistently outperform FM, indicating that our classification scheme can better integrate the MT output into the TM system than this naive baseline.

The SI feature set does not perform well when the confidence level is set above 0.85 (cf. the descending tail of the SI curve in Figure 1). This might indicate that this feature set is not reliable enough to extract the best translations. However, when the requirement on precision is not that high, and the MT-internal features are not available, it would still be desirable to obtain translation recommendations with these black-box features. The difference between SYS and ALL is generally small, but ALL performs steadily better in [0.5, 0,8].

Table 3: Recall at Fixed Precision

	Recall
SYS @85PREC	88.12±1.32
SYS @90PREC	52.73±2.31
SI @85PREC	87.33±1.53
ALL @85PREC	88.57±1.95
ALL @90PREC	51.92±4.28

5.5 Precision Constraints

In Table 3 we also present the recall scores at 0.85 and 0.9 precision for SYS, SI and ALL models to demonstrate our system's performance when there is a hard constraint on precision. Note that our system will return the TM entry when there is an exact match, so the overall precision of the system

is above the precision score we set here in a mature TM environment, as a significant portion of the material to be translated will have a complete match in the TM system.

In Table 3 for MODEL@K, the recall scores are achieved when the prediction precision is better than K with 0.95 confidence. For each model, precision at 0.85 can be obtained without a very big loss on recall. However, if we want to demand further recommendation precision (more conservative in recommending SMT output), the recall level will begin to drop more quickly. If we use only system-independent features (SI), we cannot achieve as high precision as with other models even if we sacrifice more recall.

Based on these results, the users of the TM system can choose between precision and recall according to their own needs. As the threshold does not involve training of the SMT system or the SVM classifier, the user is able to determine this trade-off at runtime.

Table 4: Contribution of Features

	Precision	Recall	F Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
+M1	82.87±1.26	96.23±0.53	89.05±.52
+LM	82.82±1.16	96.20±1.14	89.01±.23
+PS	83.21±1.33	96.61±0.44	89.41±.84

5.6 Contribution of Features

In Section 4.3.3 we suggested three sets of system-independent features: features based on the source- and target-side language model (LM), the IBM Model 1 (M1) and the fuzzy match scores on pseudo-source (PS). We compare the contribution of these features in Table 4.

In sum, all the three sets of system-independent features improve the precision and F-scores of the MT and TM system features. The improvement is not significant, but improvement on every set of system-independent features gives some credit to the capability of SI features, as does the fact that SI features perform close to SYS features in Table 1.

6 Analysis of Post-Editing Effort

A natural question on the integration models is whether the classification reduces the effort of the translators and post-editors: after reading these recommendations, will they translate/edit less than they would otherwise have to? Ideally this question would be answered by human post-editors in a large-scale experimental setting. As we have not yet conducted a manual post-editing experiment, we conduct two sets of analyses, trying to show which type of edits will be required for different recommendation confidence levels. We also present possible methods for human evaluation at the end of this section.

6.1 Edit Statistics

We provide the statistics of the number of edits for each sentence with 0.95 confidence intervals, sorted by TER edit types. Statistics of positive instances in classification (i.e. the instances in which MT output is recommended over the TM hit) are given in Table 5.

When an MT output is recommended, its TM counterpart will require a larger average number of total edits than the MT output, as we expect. If we drill down, however, we also observe that many of the saved edits come from the *Substitution* category, which is the most costly operation from the post-editing perspective. In this case, the recommended MT output actually saves more effort for the editors than what is shown by the TER score. It reflects the fact that TM outputs are not actual translations, and might need heavier editing.

Table 6 shows the statistics of negative instances in classification (i.e. the instances in which MT output is not recommended over the TM hit). In this case, the MT output requires considerably more edits than the TM hits in terms of all four TER edit types, i.e. insertion, substitution, deletion and shift. This reflects the fact that some high quality TM matches can be very useful as a translation.

6.2 Edit Statistics on Recommendations of Higher Confidence

We present the edit statistics of recommendations with higher confidence in Table 7. Comparing Tables 5 and 7, we see that if recommended with higher confidence, the MT output will need substantially less edits than the TM output: e.g. 3.28 fewer substitutions on average.

From the characteristics of the high confidence recommendations, we suspect that these mainly comprise harder to translate (i.e. different from the SMT training set/TM database) sentences, as indicated by the slightly increased edit operations

Table 5: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	0.9849 ± 0.0408	2.2881 ± 0.0672	0.8686 ± 0.0370	1.2500 ± 0.0598
TM	0.7762 ± 0.0408	4.5841 ± 0.1036	3.1567 ± 0.1120	1.2096 ± 0.0554

Table 6: Edit Statistics when NOT Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	1.0830 ± 0.1167	2.2885 ± 0.1376	1.0964 ± 0.1137	1.5381 ± 0.1962
TM	0.7554 ± 0.0376	1.5527 ± 0.1584	1.0090 ± 0.1850	0.4731 ± 0.1083

Table 7: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.85

	Insertion	Substitution	Deletion	Shift
MT	1.1665 ± 0.0615	2.7334 ± 0.0969	1.0277 ± 0.0544	1.5549 ± 0.0899
TM	0.8894 ± 0.0594	6.0085 ± 0.1501	4.1770 ± 0.1719	1.6727 ± 0.0846

on the MT side. TM produces much worse edit-candidates for such sentences, as indicated by the numbers in Table 7, since TM does not have the ability to automatically reconstruct an output through the combination of several segments.

6.3 Plan for Human Evaluation

Evaluation with human post-editors is crucial to validate and improve translation recommendation. There are two possible avenues to pursue:

- Test our system on professional post-editors. By providing them with the TM output, the MT output and the one recommended to edit, we can measure the true accuracy of our recommendation, as well as the post-editing time we save for the post-editors;
- Apply the presented method on open domain data and evaluate it using crowdsourcing. It has been shown that crowdsourcing tools, such as the Amazon Mechanical Turk (Callison-Burch, 2009), can help developers to obtain good human judgements on MT output quality both cheaply and quickly. Given that our problem is related to MT quality estimation in nature, it can potentially benefit from such tools as well.

7 Conclusions and Future Work

In this paper we present a classification model to integrate SMT into a TM system, in order to facilitate the work of post-editors. Insodoing we handle the problem of MT quality estimation as binary prediction instead of regression. From the post-editors' perspective, they can continue to work in their familiar TM environment, use the same cost-estimation methods, and at the same time benefit from the power of state-of-the-art MT. We use SVMs to make these predictions, and use grid search to find better RBF kernel parameters.

We explore features from inside the MT system, from the TM, as well as features that make no assumption on the translation model for the binary classification. With these features we make glass-box and black-box predictions. Experiments show that the models can achieve 0.85 precision at a level of 0.89 recall, and even higher precision if we sacrifice more recall. With this guarantee on precision, our method can be used in a TM environment without changing the upper-bound of the related cost estimation.

Finally, we analyze the characteristics of the integrated outputs. We present results to show that, if measured by number, type and content of edits in TER, the recommended sentences produced by the classification model would bring about less post-editing effort than the TM outputs.

This work can be extended in the following ways. Most importantly, it is useful to test the model in user studies, as proposed in Section 6.3. A user study can serve two purposes: 1) it can validate the effectiveness of the method by measuring the amount of edit effort it saves; and 2) the byproduct of the user study – post-edited sentences – can be used to generate HTER scores to train a better recommendation model. Furthermore, we want to experiment and improve on the adaptability of this method, as the current experiment is on a specific domain and language pair.

Acknowledgements

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University. We thank Symantec for providing the TM database and the anonymous reviewers for their insightful comments.

References

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *The 20th International Conference on Computational Linguistics (Coling-2004)*, pages 315 – 321, Geneva, Switzerland.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263 – 311.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *The 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 286 – 295, Singapore.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273 – 297.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *The 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181 – 184, Detroit, MI.

Philipp. Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *The 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL/HLT-2003)*, pages 48 – 54, Edmonton, Alberta, Canada.

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

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707 – 710.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. 2007. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267 – 276.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 295 – 302, Philadelphia, PA.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *The 41st Annual Meeting on Association for Computational Linguistics (ACL-2003)*, pages 160 – 167.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61 – 74.

Christopher B. Quirk. 2004. Training a sentence-level machine translation confidence measure. In *The Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, pages 825 – 828, Lisbon, Portugal.

Richard Sikes. 2007. Fuzzy matching in theory and practice. *Multilingual*, 18(6):39 – 43.

Michel Simard and Pierre Isabelle. 2009. Phrase-based machine translation in a computer-assisted translation environment. In *The Twelfth Machine Translation Summit (MT Summit XII)*, pages 120 – 127, Ottawa, Ontario, Canada.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *The 2006 conference of the Association for Machine Translation in the Americas (AMTA-2006)*, pages 223 – 231, Cambridge, MA.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009a. Estimating the sentence-level quality of machine translation systems. In *The 13th Annual Conference of the European Association for Machine Translation (EAMT-2009)*, pages 28 – 35, Barcelona, Spain.

Lucia Specia, Craig Saunders, Marco Turchi, Zhuoran Wang, and John Shawe-Taylor. 2009b. Improving the confidence of machine translation quality estimates. In *The Twelfth Machine Translation Summit (MT Summit XII)*, pages 136 – 143, Ottawa, Ontario, Canada.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *The Seventh International Conference on Spoken Language Processing*, volume 2, pages 901 – 904, Denver, CO.

Nicola Ueffing and Hermann Ney. 2005. Application of word-level confidence measures in interactive statistical machine translation. In *The Ninth Annual Conference of the European Association for Machine Translation (EAMT-2005)*, pages 262 – 270, Budapest, Hungary.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *The Ninth Machine Translation Summit (MT Summit IX)*, pages 394 – 401, New Orleans, LA.