International Journal of Scientific & Engineering Research, Volume ÑÛ(ÚÚÜÌ+Û)ÜÕÌıl¥hШ
ISSN 2229-5518

Cognitive German-to-Arabic Text translation with Translation Post-Editing and translation revision Systems

Prof. Dr. Reda Houssine Abo Aleze
Prof. Dr. Mohamed fared Zaphlol
Prof. Dr. Ass. Salwa al said hamada
Prof. Dr. Ass. Ahmad Almahdy
Eng. Dahey Gaber anem

Abstract

After we make a translation errors analysis by pen and paper for the first eleven chapters of the first book of the A Song of Ice and Fire saga – called A Game of Thrones – that has been translated from German to Arabic by Google translation system. We developed an automatic translation Post-Editing System (TPES) and translation revision System (TRS) to detect and fix these errors automatically.

In the TPES we attention on neural end-to-end models that associate both source German text (src), machine translation (MT) output Arabic text and the Arabic text translation post-editing (pe) in a single neural construction, modeling {MT, src} → pe directly, We consider the effect of hard-attention models which appear to be well-suited for monolingual and bilingual tasks. Dual-attention models that are combined with hard attention remain good in spite of applying fewer changes to the input. TPES produced by RNN show statistically significant improvements of 3.96 BLEU points absolute over the original MT. Also, human evaluation shows that, TPES generated PE translations are much better than the original MT output.

In the TRS, the system decomposed into two simple actions, Pick (P) and Revise (R). The picked phrase could be at any position of the sentence, which improves the efficiency of human computer interaction. We also propose automatic proposal models for the two actions to reduce the cost of human interaction. Experiment results demonstrate that, the translation quality by TRS could be significantly improved (around +2 BLEU points in one PR cycle). Greater improvements could be achieved the translation quality +17 BLEU by iteratively performing both revision actions.

1. Introduction

Translations provided by state-of-the-art MT systems suffer from a number of errors including incorrect lexical choice, word ordering, word insertion, word deletion, etc. The translation errors can be classified as the Post-editing parameters (errors) and the translation revision parameters (errors). Table 1 show, the translation Post-editing parameters and the translation revision parameters. The TPES is a system developed to improve the MT output by rectifying some of these errors. For this purpose we use a deep neural network (DNN) based approach. Neural MT (NMT) [1, 2] is a newly emerging approach to MT. On the one hand DNNs represent language in a continuous vector space which eases the modelling of semantic similarities (or distance) between phrases or sentences, and on the other hand it can also consider contextual information, e.g., utilizing all available history information in deciding the next target word, which is not an easy task to model with standard APE systems.

Our translation Post-editing system (TPES) detect and fix the errors:

1. Fluency parameters
2. Adequacy parameters

Our Translation Revision System (TRS) detect and fix the errors:

1. Meaning Transfer (Accuracy): Does the translation reflect The message of the source text?
2. Language (Mechanics): Have the rules of grammar, spelling, punctuation, house style been observed?

TPES developed by assume the availability of German source language input text (SL_{IP}), Arabic target language MT output (TL_{MT}) and Arabic target language PE data (TL_{PE}).

The TPES can be modelled as an MT system between SL_{IP}, TL_{MT} and TL_{PE}.

- *Author name is Eng.dahey gaber anem PHD degree program in electric power engineering in University ALAzhar, Country eygpt . E-mail: adahey@yahoo.com*

International Journal of Scientific & Engineering Research, Volume ÑÛ(ÚÚÜÌ ÌÛ)ÜÕÌ ÌÝÛÌÌ
ISSN 2229-5518

Parameters categories	Parameters types	Sub-parameters types
The translation Post-editing parameters	Group A – Fluency parameters	1. Wrong term 2. Syntactic errors 3. Punctuation errors
	Group B – Adequacy parameters	4. Omission 5. Word-structure 6. Misspelling 7. A miscellaneous errors.
The translation revision parameters	Group A (Transfer)	1. Accuracy 2. Completeness
	Group B (Content)	3. Logic 4. Facts
	Group C (Language)	5. Smoothness 6. Tailoring 7. Sub-languages 8. Idiom 9. Mechanics
	Group D (Presentation)	10. Layout 11. Typography 12. Organization

Table 1: the translation Post-editing parameters and the Translation revision parameters

However, if we do not have access to SL_{IP} , but have sufficiently large amounts of parallel TL_{MT} , TL_{PE} data, we can still build the TPES between TL_{MT} and TL_{PE}. Unlike phrase-based APE systems [3], our TPES builds and trains a single, large neural network that accepts a 'draft' translation (TL_{MT}) and outputs an improved translation (TL_{PE}). To make the translation errors correction automatically we use Google Translation System (GTS) to translate the source German text to the target Arabic text. The output of GTS is frequently extremely grammatically incorrect due to the absence of linguistic rules for the language pair being applied. Grammatical error not only fails the fluency and adequacy of the translation. In translating German to Arabic by Google translation system, the BLEU scores [4] range only between 0.21 and 0.29, depending on the test sets and numbers of reference translations.

TPES needs a lot of modifications to obtain high quality translations, human revisers usually wanted to modify the results generated by TPES. The revision process performed by human is time-consuming and need high energy. To speed up the revision process, we developed the TRS to update the good translation result after every human action. TRS make the translation modification by two actions:

1- a wrongly-translated phrase is selected from the whole text (Pick);

2- The correct translation is selected from the translation table (or manually added) to replace the original one (Revise).

Our TRS can retranslate the sentence and searches for the best translation using previous modifications as constraints. Also, we developed two automatic suggestion models that could predict the wrongly-translated phrases and select the revised translation, respectively. With the suggestion models, users only perform one of the actions (picking or revising) and let the suggestion models complete the other one. In this case, the interactions could be further simplified to be only one of the actions, which is as simple as one mouse click.

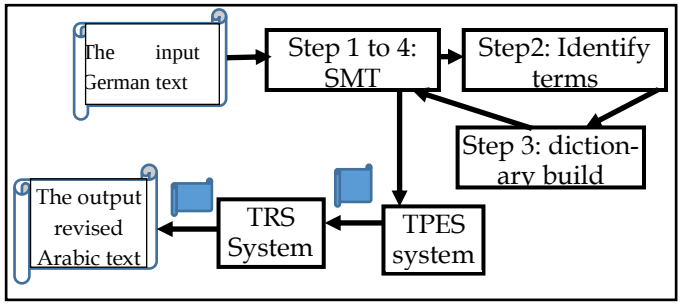

Figure 1: The translation system with translation post-editing and translation revision systems.

The remainder of the paper is organized as follows. Section 2 gives an overview of relevant related work. The proposed TPES is described in detail in Section 3. The proposed TRS is described in detail in Section 4. Section 5 presents the results of automatic and human evaluation together with some analysis. Section 6 concludes the paper and provides avenues for future work.

2. Related works

Before the application of neural sequence-to sequence models to TPES, most TPES would rely on phrase-based SMT following a Monolingual approach first introduced by [3]. [5] Proposed a "source-context aware" variant of this approach where automatically created word alignments were used to create a new source language which consisted of joined MT output and source token pairs. The addition of source-language information in that form was shown to improve the automatic post-editing results [5, 6]. The quality of the used word alignments plays an important role for these methods, as demonstrated for instance by [7]. During the WMT-2016 APE shared task two systems relied on neural models, the CUNI system [8] and the shared task winner, the system submitted by the AMU team [9]. This submission explored the application of neural translation models to the APE problem and achieved good results by treating different models as components in a log-linear model, allowing for multiple inputs (the source src and the translated sentence MT) that were decoded to the same target language (post-edited translation pe). Two systems were considered, one using src as the input (src → pe) and another using MT as the input (MT → pe). A simple string-matching penalty integrated within the log-linear model was used to control for higher faithfulness with regard to the raw MT output. The penalty fired if the APE system proposed a word in its output that had not been seen in MT. The influence of the components on the final result was tuned with Minimum Er-

International Journal of Scientific & Engineering Research, Volume ÑÙ(ÚÚÜÌ ÌÙ)ÜÕÌ ÌÝÙ
ISSN 2229-5518

ror Rate Training [10] with regard to the task metric TER. Following the WMT-2016 APE shared task, [11] published work on another neural APE system that integrated recomputed word alignment features into the neural structure and enforced symmetric attention during the neural training process. The result was the best reported single neural model for the WMT-2016 APE test set prior to this work. With n-best list re-ranking and combination with phrase-based post-editing systems, the authors improved their results even further. None of their systems, however, integrated information from src, all modelled MT → pe.

3. Translation Post-Editing System (TPES)

3.1 A Bidirectional RNN APE Encoder-Decoder

Our TPES encodes a variable-length sequence of TL_{MT} (e.g. $x = x_1\ x_2\ x_3 \ldots\ x_m$) into a fixed-length vector representation and then decodes a given fixed-length vector representation back into a variable-length sequence of TL_{PE} (e.g. $y = y_1\ y_2\ y_3 \ldots\ y_n$). Input and output sequence lengths, *m* and *n*, may differ.

A Bidirectional RNN encoder consists of forward and backward RNNs. The forward RNN encoder reads in each x sequentially from x_1 to x_m and at each time step *t*, the hidden state h_t of the RNN is updated by using a non-linear activation function *f* (Equation 1), an element wise logistic sigmoid with an LSTM unit.

$$h_t = f(h_{t-1}, x_t) \quad (1)$$

Similarly, the backward RNN encoder reads the input sequence and calculates hidden states in reverse direction (i.e. x_m to x_1 and h_m to h_1 respectively). After reading the entire input sequence, the hidden state of the RNN is provided a summary *c* context vector ('C' in Figure 2) of the whole input sequence.

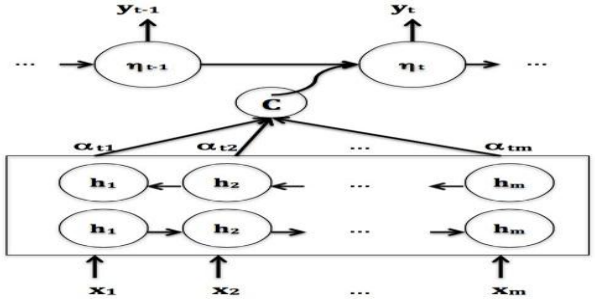

Figure 2: Generating the t^{th} TL_{PE} word y_t for a given TL_{MT} (x) by our TPES.

The decoder is another RNN trained to generate the output sequence by predicting the next word y_t given the hidden state η_t and the context vector c_t. The hidden state of the decoder at time *t* is computed as given below.

$$P(y_t \mid y_1 \ldots, y_{t-1}, x) = f(\eta_t, y_{t-1}, c_t) \quad (2)$$

$$\eta_t = f(\eta_{t-1}, y_{t-1}, c_t) \quad (3)$$

The context vector c_t can be computed as

$$c_t = \sum_{i=1}^{m} \propto_{ti} h_i \quad (4)$$

Here, $\propto_{ti}$ is the weight of each h_i and can be computed as

$$\propto_{ti} = \exp(e_{ti}) \,/\, \sum_{i=1}^{m} \exp(e_{ti}) \quad (5)$$

Where $e_{ti} = a(\eta_{t-1}, h_i)$ is an alignment model which provides a matching score between the inputs around position *i* and the output at position *t*. The alignment score is based on the i^{th} annotation h_i of the input sentence and the RNN hidden state η_{t-1}. The alignment model itself is a feed forward neural network which directly computes a soft alignment that allows the gradient of the cost function to be back propagated through. The gradient is used to train the alignment model as well as the $TL_{MT} - TL_{PE}$ translation model jointly.

The alignment model is computed $m \times n$ times as follows:

$$a(\eta_{t-1}, h_i) = v_a^T \tanh(W_a \eta_{t-1} + U_a h_i) \quad (6)$$

Where $W_a \in \boldsymbol{R}^{n\hbar \times n\hbar}$, $U_a \in \boldsymbol{R}^{n\hbar \times 2n\hbar}$, and $v_a \in \boldsymbol{R}^{n_h}$ are the weight matrices of n_h hidden units.

We evaluate the model on a German–Arabic ATPE system, Figure 4 show generating the t^{th} TL_{PE} word y_t for a given TL_{MT} (x) by our TPES.

3.2 Translation Post-Editing System (TPES) architecture

The TPES is based on a bidirectional (forward-backward) RNN based encoder-decoder. This is where most of the Tensor Flow code is located. Encoders are parts of the system which take some input and compute a representation of it. The TPES workflow illustrated in Figure 3, the system uses two encoders which are implemented using a bi-directional GRU [19] network. The first encoder is fed with sentences in the source language and the second one is fed with the machine translation output. Decoders are system parts that produce some outputs. A decoder combines outputs of both the encoders and attends to their hidden states during decoding. A trainer that follows the decoders computes the error of the decoder outputs given the desired target text from the train dataset. Our definition of encoders and decoders is more general than in the classical sequence-to-sequence (S2S) learning. The RNN decoder is for us only a special type of decoder; it can be also a sequence labeler or a simple multilayer perceptron classifier or regression. Decoders are executed using so-called runners. Different runners represent different ways of running the model. We might want to get a single best estimation, get an n-best list or a sample from the model. We might want to use an RNN decoder to get the decoded sequences or we might be interested in the word alignment obtained by its attention model. This is all done by employing different runners over the decoders. The outputs of the runners can be subject to more post processing. In addition to runners, each experiment has to have its trainer. A trainer is a special case of a runner that actually modifies the parameters of the model. It collects the objective functions and uses them in an optimizer. RNN manages Tensor Flow sessions using an object called Tensor Flow manager. Its basic capability is to execute runners on provided datasets. It encapsulates all logic concerning the Tensor Flow sessions. The experiments are described using configuration files. They contain a complete specification of the network architecture, preprocessing and post-processing of the data, the training and validation data, all me-

International Journal of Scientific & Engineering Research, Volume ÑÛ(ÚÚÜÌ ÌÛ)ÜÕÌı l ¥hШ
ISSN 2229-5518

ta-parameters of the training, and metrics used for model evaluation. The configuration files are the main tool of RNN experiment management. The same configuration can be used after training to run the trained model. Figure 3 show the TPES workflow.

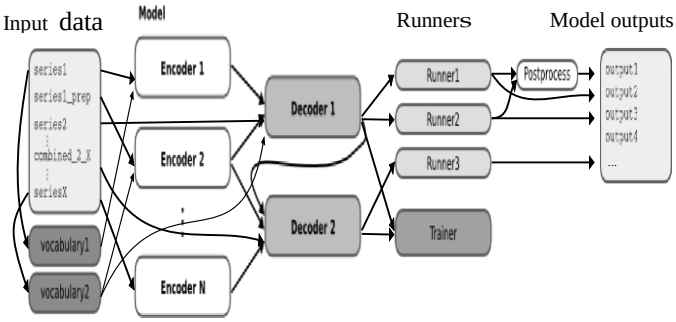

Figure 3: TPES workflow

Our TPES contains 1000 hidden units for the forward backward RNN encoder and 1000 hidden units for the decoder. The network is basically a many-sided neural network with a single maxout unit as hidden layer [12] to compute the conditional probability of each target word. The word embedding vector dimension is 620 and the size of the maxout hidden layer in the deep output is 500. The number of hidden units in the alignment model is 1000. The model has been trained on a mini-batched stochastic gradient descent (SGD) with 'Adadelta' [13]. The main reason behind the use of 'Adadelta' is to automatically adapt the learning rate of each parameter ($\epsilon = 10^{-6}$ and ρ = 0.95). Each SGD update direction is computed using a mini-batch of 80 sentences. Figure 4 show the architecture of the Translation Post-Editing System (TPES).

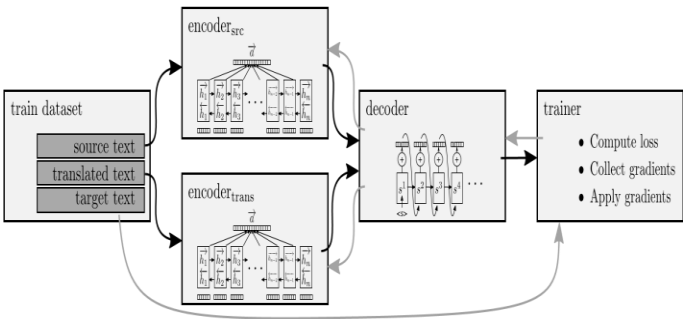

Figure 4: The architecture for the Translation Post-Editing System (TPES)

For building the TPES, we set maximum phrase length to 7. A 5-gram language model built using Ken LM [14] was used for decoding. System tuning was carried out using both k-best MIRA [15] and Minimum Error Rate Training (MERT) [16] on the held out development set (dev. set). After parameters were tuned, decoding was carried out on the held out test set. Dataset series can be used to create a vocabulary. A vocabulary represents an indexed set of tokens and provides functionality for converting lists of tokenized sentences into matrices of token indices and vice versa. Vocabularies are used by encoders and decoders for feeding the provided series into the neural network. We used 12k sentences TPES training dataset for training and we computed BLEU [17] on the 73 chapters of A Game of Thrones (German-Arabic) and consist of 312K $TL_{MT} - TL_{PE}$ parallel sentences to compare the baselines. The evaluation was performed during training. We thus did not use beam search and simply greedily chose the most probable output at each decoding step to get the validation output. We can see that the character-level post-editing models outperform the sub word-level models. However, the training was done using only a small dataset which may possibly indicate that the character level architecture is able to better exploit the training data. Nevertheless, we chose the character-level system for our remaining experiments.

We also trained a separate model that generates a sequence of post-editing operations Inspired by [18], ("edit ops") instead of directly generating the target sequence of characters. Aside from generating characters present in the training data, the system learns to use special tokens "<keep>" and "<delete>", or "<insert>" to indicate the modifications needed for the MT output.

3.3 Loading and Processing Artificial Datasets

Before the system is applied to the data, there is a short pipeline of steps preparing the data. In the simple case of machine translation, there are two data series: a list of sentences in the source language and a list of sentences in the target language. Figure 5 show creating a dataset in TPES, The dataset is created in the following steps:

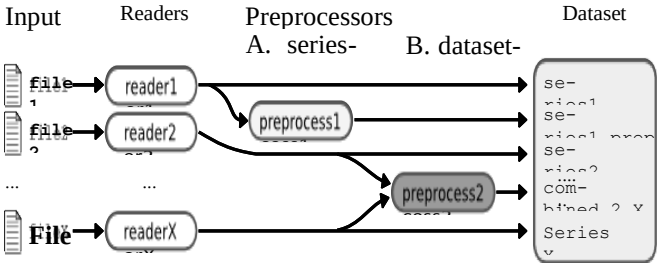

Figure 5: Creating a dataset in TPES

1. An input file is read using a reader. Reader can e.g., load a file containing paths to JPEG images and load them as NumPy arrays, or read tokenized text as a list of lists (sentences) of string tokens.
2. Series created by the readers can be preprocessed by some series-level preprocessors. An example of such preprocessing is byte-pair encoding [20] which loads a list of merges and segments the text accordingly.
3. The final step before creating a dataset is applying dataset-level preprocessors which can take more series and output a new series.

At present, there are two implementations of a dataset:

International Journal of Scientific & Engineering Research, Volume
ISSN 2229-5518

1. An in-memory dataset which stores all data in the memory.
2. A lazy dataset which gradually reads the input files step by step and only store the batches necessary for the computation in the memory.

The data will be divided into two set of texts: (I) the original text in German, which is the source text (ST); and (II) the Google translation for Arabic text, which is the target text (TT), The training data used for the experiments was developed in the 73 chapters of A Game of Thrones (German-Arabic), 11 chapters were analyzed, there are 129 occurrences of translation errors were found and consists of 312K $TL_{MT} - TL_{PE}$ parallel sentences (German -Arabic). Google Translate (GT) is the MT engine which provided the original Arabic TL_{MT} output. Since the data contains some non-Arabic sentences, we applied automatic language identification [21] in order to select only Arabic sentences. Automatic cleaning and pre-processing of the data was carried out by sorting the entire parallel training corpus based on sentence length, filtering the parallel data on maximum allowable sentence length of 80 and sentence length ratio of 1:2 (either direction), removing duplicates and applying tokenization and punctuation normalization using Moses [22] scripts. After cleaning the corpus we obtained a sentence aligned $TL_{MT} - TL_{PE}$ parallel corpus containing 213,795 sentence pairs. We randomly extracted 1000 sentence pairs each for the development set and test set from the pre-processed parallel corpus and used the remaining (211,795) as the training corpus. The training data features 57,568 and 61,582 unique word types in TL_{MT} and TL_{PE}, respectively. We chose the 40,000 most frequent words from both TL_{MT} and TL_{PE} to train our NNAPE model. The remaining words which are not among the most frequent words are replaced by a special token ([UNK]). The model was trained for approximately 35 days, which is equivalent to 2,000,000 updates with GPU settings. Table 2 shows the examples occurrences of Translation Errors in Artificial Data, Table 3 show the occurrences of Translation Errors according to Mossop's Model.

The input German text	**The Google Arabic text translation**	**Human Arabic text translation**	**Errors category**	**Errors parameter**
Der winter kommt	الشتاء قادم	الشتاء قادم	Post-editing	Consistency
Gared gehörte seit vierzig Jahren der Nachtwache an, als Mann und schon als Junge, und er war es nicht gewohnt, dass man sich über ihn lustig machte.	كان جارد حارساً ليلياً لمدة أربعين عاماً ، كرجل وكصبي ، ولم يكن معتاداً على السخرية منه.	لقد قضي الشيخ أربعين عاما كاملة مع حرس الليل،منذ إلتحق بهم وهو صبي.ولم يكن يروق له أن يستخف به الأخرين	language and style	idiom
Vier Jahre war er auf der Mauer. Als man ihn zum ersten Mal auf die andere Seite geschickt hatte, waren ihm all die alten Geschichten wieder eingefallen, und fast war ihm das Herz in die Hose gerutscht.	كان على الحائط لمدة أربع سنوات. في المرة الأولى التي تم إرساله فيها إلى الجانب الآخر ، كانت جميع القصص القديمة قد عادت إليه ، وكان قلبه ينزلق في سرواله.	ويوم أرسلوه وراء الجدارللمرة الأولي وجد الحكايات القديمة تتدفق من ذاكرته وشعر بأمعائه تتقلص	meaning transfer	completeness
Was gibt es da?	ما هو هناك؟	من هناك ؟	language and style	smoothness
Will trat an den Baum, einen gewölbten, graugrünen Wachbaum, und begann zu klettern.	صعد إلى الشجرة ، شجرة حراسة خضراء رمادية مقببة ، وبدأ في الصعود.	إتجه إلي شجرة الحارس الضخمة ذات الأفرع المقنطرة واللون الأخضروالرمادي وبدأ يتسلق	meaning transfer	accuracy
Der Andere zögerte. Will sah seine Augen, dunkler und blauer, als Menschenaugen jemals sein konnten, ein Blau, das brannte wie Eis. Sie richteten sich auf das Langschwert, das dort oben bebte, betrachteten das Mondlicht, das kalt über das Metall lief. Einen Herzschlag lang wagte er zu hoffen.	الآخر تردد. سوف يرى عينيه ، أكثر قتامة وأكثر زرقة من العين البشرية يمكن أن تكون ، زرقاء تحترق مثل الثلج. كانوا يستهدفون الفستان الطويل الذي يرتعد هناك ، يراقبون ضوء القمر البارد فوق المعدن. لدقات قلب تجرأ على الأمل.	توقف الأخر ،ورأي ويل عينيه .كانتا ذات لون أزرق شديد العمق،يحرق كالجليد،أعمق وأكثر زرقة من أي عين بشرية ،وقد ثبتت نظراتهما علي السيف الطويل الذي يرتفع مرتجفا في يد صاحبه،وراقبتا نور القمر البارد يجري علي المعدن،وللحظة جرؤ ويل علي الأمل	content	logic
Er nannte sie »kleine Prinzessin«	دعاها "الأميرة الصغيرة" وأحيانا "سيدتي"	وقال أمرا :"قفي مكانك "، ثم	language and	mechanics

und manchmal »Mylady«, und seine Hände waren weich wie altes Leder.	، وكانت يديه ناعمة مثل الجلود القديمة.	"إستديري. نعم عظيم. تبدين بهية..." "كملكة" وكان يمرر يده علي صفحة الماء	style	

Table 2: The examples of occurrences of Translation Errors

3.4 TPES Evaluation

The performance of the TPES was evaluated using both automatic and human evaluation methods.

3.4.1 Automatic Evaluation

The output of the TPES on the 1000 sentences test set was evaluated using two MT evaluation metrics: BLEU [23] and TER [24]. Table 4 provides a comparison of our neural system performance against the original GT output. Table 1 means that the improvement provided by S_2 in BLEU is statistically significant over Google Translator and phrase-based APE. A logical trend (*RNN* > *GT*) can be observed in Table 3 and the improvements are consistent across the two metrics. The relative performance gain achieved by RNN over GT is highest in TER.

System	BLEU	TER
GT	61.26	30.94
RNN	65.22	27.56

Table 3: TPES automatic evaluation

3.4.2 Human Evaluation

Human evaluation was carried out by four professional translators, native speakers of Arabic, with professional translation experience between one and two years. Since human evaluation is very costly and time consuming, it was carried out on a small portion of the test set consisting of 145 randomly sampled sentences and only compared TPES with the original GT output. We used a polling scheme with three different options. Translators choose which of the two (GT or TPES) outputs the better translation is or whether there is a tie ('uncertain'). To avoid any bias towards any particular system, the order in which two system outputs are presented is randomized so that the translators do not know which system they are contributing their votes to. We analyzed the outcome of the voting process (4 translators each giving 145 votes) and found that the winning TPES received 285 (49.13%) votes compared to 99 (17.07%) votes received by the GT system, while the rest of the votes (196, 33.79%) go to the 'uncertain' option. On manual examination we found that the TPES drastically reduced the preposition insertion and deletion error in Arabic GT output and was also able to handle the improper use of prepositions and determiners.

4 Translation Revision System (TRS) architecture

In TRS fewer interactions are needed to add improve the translation quality. The TRS picks the phrase which was considered the most critical translation error and revises the translation according to phrase table. After a PR cycle, our Controlled decoder retranslates the sentence. It not only generates the correct translation for the pick-revise pair (PRP), but also improves the translation around the PRP. Our TRS can modify the most critical error at first, which brings larger improvements on translation quality and improves the translation efficiency. Figure 6 shows an overview of our framework for a source sentence $s_1\ s_2\ s_3 \ldots\ s_n$. Our TRS iteratively generates the translation using a Controlled decoder. The Controlled come from previous picking and revising processes. The picking and revising results can also be collected for model adaptation. The whole process continues until the translation is considered acceptable by the users. We explain the key components of our system below.

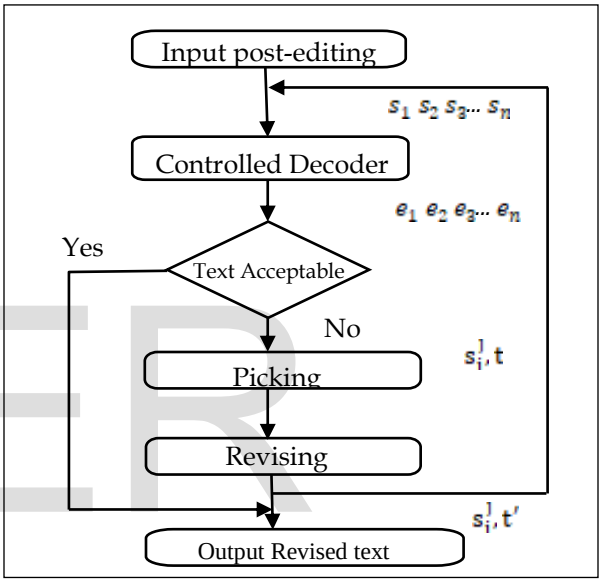

Figure 6: The TRS architecture.

4.1 The automatic Picking process

In the picking step, the TRS pick the wrongly translated phrase (s_i^j, t) to be revised (where s_i^j is the phrase that covers the source words from index i to j), and translated into t. The picking process aims at finding critical errors in the translation, caused by errors in the translation table or essential translation ambiguities. The correcting these critical errors make the larger improvement of translation quality [25]. Critical errors might have a large effect to the translation of their context.

To make the picking step easier to be integrated into TPES, we limit the selection of translation errors to be those phrases in the previous PR cycle output. If it's the first PR-cycle, then those errors come from phrases used to generate the baseline post-editing translation. For more suitable user interactions, in our TRS system, critical errors can be picked from both the source and target side by simply a mouse click on it. The correspondence/alignment between source and target phrases is visualized for easier human observation.

To further reduce the human actions, we propose to use automatic suggestion models for the picking step. Such models can offer suggestions to users in picking. Because picking actions is

International Journal of Scientific & Engineering Research, Volume [illegible]
ISSN 2229-5518

performing selections from multiple candidates, we use classifier-based approaches to model this step. Note that these automatic suggestion models could be interpreted as simplified self-assurance measurements.

Modeling the picking process needs two aspects of information. One of them is to determine whether the phrase is difficult-to-translate; the other is to determine whether the current translation option is correct. We use features from translation models (TMs), language models (LMs), lexical reordering models (LRMs), as well as counting and lexical features in Table 4. These features cover information of the source side, target side, translation ambiguity, and context, etc.

Category	Description
TM	TM scores of baseline translation
	Normalized TM scores of baseline translation
	TM entropy of all translation options
LM	LM score of baseline translation
	LM score of previous/next phrase translation
	LM score of each target word
	LM score of the bigram at the border of current and previous/next phrase
LRM	LRM scores of baseline translation
	LRM scores of previous/next phrase translation
Count	Source/target word count
	Number of translation options for current source phrase
POS	POS-tags of source words
	POS-tags of previous/next word of source phrase
Lexical	Source words
	Target words

Table 4: the Features of the automatic Picking process

The goal of automatic Picking process is to recognize those phrases that might be wrongly translated, and the users can either accept or refuse the suggestion. Within all the phrases of a source sentence, we need to separate the wrongly-translated phrases and correctly-translated phrases. Because translation errors often cause low translation quality, we use the translation quality gain after the revising action as a measurement. We treat those phrases that achieve translation quality improvement after revising as wrongly-translated phrases; those lead to translation quality worsening as correctly-translated phrases. We select phrases that lead to a BLEU improvement/deterioration greater than a threshold is set as 10% of the BLEU score of the baseline sentence.

4.2 The automatic Revising process

The TRS revise the translation of s_i^j by selecting the correct translation t' from the translation table, or manually add one if there is no correct translation in the translation table. Whether to perform selection or adding depends on the quality of the translation table. When the translation system is trained with large enough parallel data, the quality of the translation table is usually high enough to offer the correct translation. For a picked phrase, the translation options in the phrase table could be presented to the users as a list. The users just need to click on the correct translation to complete the revising step. The users could also type a new translation through a separated input area. To further reduce the human actions, we propose to use automatic suggestion models for the picking and revising step, respectively. Such models can offer suggestions to users in both picking and revising steps. Because both picking and revising actions are performing selections from multiple candidates, we use classifier-based approaches to model these two steps. Note that these automatic suggestion models could be interpreted as simplified self-assurance measurements.

The features of automatic Revising process are showed in Table 5. For translations of a given source phrase, there is no need to compare their source-side information because these translation options share the same source phrase and context. So these features mainly focus on estimating the translation quality of a given translation option. As a result, features for automatic Revising process only including the scores for TM, LM and LRM, etc., which are simpler compared to PSM. Table 7 show the Features for the RSM

Category	Description
TM	TM scores of current translation option
LM	LM score of current translation option
	LM score of each target word
LRM	LRM scores of current translation option
count	Target word count
Lexical	Target words

Table 5: The features of automatic Revising process

The goal of automatic Revising process is to predict the correct translation and suggest users to replace the wrong translation with the predicted one. The users can either accept it or use another translation.

Translation table has multiple translation options for one phrase. Within the translation option set of a source phrase, we need to separate the correct and wrong translation options.

International Journal of Scientific & Engineering Research, Volume
ISSN 2229-5518

Instead of asking human translators to label these translations, we use two criteria to distinguish correct translation options from wrong translation options.

Firstly, the correct translation option should be a substring of the references, which ensures the correctness of the options itself. Secondly, the correct translation option should be consistent with pertained word alignment on the translated sentence pair[1]. This is to ensure that the translation option does not get credit for words that are not translations of the source side phrase. The remaining options are considered wrong translations. With the above criteria, we select all correct translation options as positive instances for the revising step, and randomly sample the same number of wrong translation options to be negative instances. Specifically, translation options that are used by the baseline system are included as negative instances.

4.3 The Controlled Decoder

A pick-revise pair (PRP), (s_i^j, t^r), is obtained after a PR cycle for a source sentence. We use a Controlled decoder to search for the best translation with the previous PRPs as constraints. The constrained search algorithm is similar to the algorithm in a typical phrase-based machine translation [26]. The only exception is that it makes an extra comparison between each translation option and previous PR pairs, which ignores all the phrases that overlap with the source side of a PRP. As a result, a lot of translation options are ignored, which makes the search space much smaller than standard decoding. In this way, we could guarantee that all the PRPs are correctly translated and the whole process can be carried out in real-time. The system could collect all PRPs and adapt the models using methods described in [27] or [28].

5 Experiment Sceneries

5.1 Loading and Processing Artificial Datasets

Throughout the experiments, we use Google machine translation system and join our TRS into the TPES. The parallel data for training the TRS includes in the 73 chapters of A Game of Thrones (German-Arabic) and consists of 312K $\mathrm{TL_{MT}} - \mathrm{TL_{PE}}$ parallel sentences. The parallel sentences are (German to Arabic) MT output and their corresponding (human) post-edited Arabic translations. Google Translate (GT) is the MT engine which provided the original Arabic TLMT output. A 5gram language model is trained with MKN smoothing [29] on Xinhua portion of Arabic Giga word Data There are 319 files, totaling approximately 1.1GB in compressed form (4348 MB uncompressed, and 391619 K words).

We use a combination of the parallel data for training to tune the MT system parameters and train the suggestion models. We test the system on the parallel data. The translation results are evaluated with case insensitive 4-gram BLEU [30]. Our baseline phrase-based MT system has comparable performance with the Google machine translation system.

5.2 Classification Sceneries

We use the classification model to implement the PRP, the neural network model. We use a feed forward neural network with the CNTK implementation [33]. The neural network has one hidden layer of 80 nodes, with sigmoid function as the activation function. We use pre-trained word embedding [34] for the neural model.

5.3 TRS Procedure

Because real-world human interactions are expensive and time-consuming to obtain, we use simulated human interactions for picking and revising in the TRS, directly identifying critical errors in the translation is not an easy task without human annotation. Instead, we find critical errors by judging the effect of a given error to the translation of their context. We try picking every phrase in a baseline translation result and revising it using the simulated revising strategy. The influence of the phrase is measured by the translation quality improvement after re-translation with the current phrase be revised. The phrase with the highest translation quality improvement is picked to be the simulated human picking result. Given the phrase to be revised, the simulated revising action is direct. Among all the translation options that are considered correct, we choose the longest one to be the simulated human revising result. With the above simulated actions, one PR cycle takes exactly two mouse clicks and none keystroke. We compare the post editing method which selects the most critical error and edits it to be the simulated revising translation. The key-stroke count for each editing is the number of characters of the correct phrase translation.

5.4 Examples of applying PR actions

We additional analyze the performance of our TRS system by examples. Table 6 shows the TRS procedure of improving translation quality for three different sentences.

[1] We trained word alignments with Giza++(Och and Ney, 2003)

International Journal of Scientific & Engineering Research, Volume [illegible]
ISSN 2229-5518

	Die WHO bestätigte Vietnams sechste tote Vogelgrippefall
	تؤكد منظمة الصحة العالمية وفاة السادس في إنفلونزا الطيور في فيتنام
	(الصحة العالمية) (المنظمة) (تأكيد) (فيتنام) (٦) () (أنفلونزا الطيور) (الموت) (حالة)
Ref	تؤكد منظمة الصحة العالمية حالة الوفاة السادسة من إنفلونزا الطيور في فيتنام
Baseline	أكدت منظمة الصحة العالمية حالات موت أنفلونزا الطيور في فيتنام
PR*1	تؤكد منظمة الصحة العالمية حدوث حالات وفاة لإنفلونزا الطيور1 في فيتنام
PR*2	تؤكد منظمة الصحة العالمية حالة 2 الموت السادسة من إنفلونزا الطيور في فيتنام
	Nationale Versöhnung erfordert in der Regel eine gewisse Menge an Prozess, und es ist schwierig
	تتطلب المصالحة الوطنية عادة كمية معينة من العملية ، ومن الصعب ذلك
	(الوطنية) (المصالحة) (عادة) (تحتاج) (معينة) () (بالطبع) (،) (لا) (إنجاز في إجراء واحد(.)
Ref	تحتاج المصالحة الوطنية عادة إلى مسار معين ، ولا يمكن تحقيقه في إجراء واحد.
Baseline	المصالحة الوطنية هي عملية صعبة للغاية.
PR*1	تحتاج عملية المصالحة الوطنية عادة إلى معينة صعبة للغاية.
PR*2	المصالحة الوطنية تحتاج عادة إلى مسار معين ، وأنجزت.
PR*3	المصالحة الوطنية عادة ما تحتاج إلى مسار معين ، وأنه لا يمكن أن يتحقق.
	Die zwei Antworten israels können jedoch von den Vereinigten Staaten nicht vollständig gelöst werden.
	(ومع ذلك) (إسرائيل) (الرد) (تفشل) (واضح تماما) (لنا) () (شك) (.)
Ref	ومع ذلك ، فشل رد إسرائيل في إزالة الشكوك بشكل كامل.
Baseline	ومع ذلك ، فإن الرد الإسرائيلي على الإزالة الكاملة للولايات المتحدة.
PR*1	ومع ذلك ، فشلت الاستجابة الإسرائيلية في الشك بوضوح تام.
PR*2	ومع ذلك ، فشل رد إسرائيل في توضيح الشكوك بشكل كامل

Table 6: Examples of applying PR actions multiple times in the German -Arabic translation.

In the first sentence, two PR cycles lead to a perfect translation. In the first PR cycle (PR*1), revising the translation of "Die sechste" from "٦" to "السادس" improves the neighboring translation. The translation of “Bestätigung” changes from “ التأكيد" to "تؤكد ", which a positive effect is. In PR*2, revising the translation of "Fälle” from "حالات" to "حالة" also changes the neighborhood translation, after two PR cycles, the reference translation is obtained. In our current settings, the reference translation could not always be obtained. The maximum achievable BLEU is around 60-70 in general environment.

In the second sentence, "Muss sicher sein" " يجب أن تكون على يقين" is picked in the first PR cycle. Revising the translation from "a" to "need a certain" makes the translation of "normalerweise" "عادي" changing from "يكون" to "دائما". In the next PR cycle, revising the translation of "Prozess" "عملية" from "process" to "course" makes the neighboring translation changing from "," to ", and". Meanwhile, the position of "course" moves to the right place (in front of ","). In the last PR cycle, the translation of "Es ist schwer zu" "من الصعب" is revised from "it" to "it cannot be". After three PR cycles, the translation quality improves significantly. However, the translation is still different from the Reference.

This is because "Auf einen Schlag” should be translated into "دفعة واحدة" instead of "يتحقق". But there is no suitable translation option for it in the current phrase table. So the system cannot generate a perfect translation. The problems will be less significant when real-world human translators are involved. Human translator inputs the correct translation "accomplished in one action", the system will generate the reference translation after constrained decoding (Human).

In the last sentence in Table 9, "Kann nicht" "لا يمكن" is picked as the critical error. Revising the translation from "إلي ال" to "تفشل", leads to an improvement on neighboring phrase (the translation of "Volles Kehren" to "بشكل كامل "), In the second PR cycle, "Israels" "إسرائيل" is picked. Revising the translation from "إسرائيلي" to "اسرائيل"، makes the translation of "Antwort" change from "الإجابة" to “ الاستجابة", this is also a positive effect. However, after two PR cycles, all phrase translations are correct, but the translation is still different from the reference. This is because the language model and lexical reordering model prefer the wrong phrase ordering, which put "الولايات المتحدة" "the US" at the end of the whole sentence. This problem arises from the MT system itself, which may not be solved directly in our TRS.

If more interactions are allowed, for example, performing reordering operations, the above problems could be solved. But the interactions become more complex, and may not be acceptable to human translators. Other solutions include using better statistical models such as neural language models [35].

6. The TPES and TRS evaluation

We also validate the improvements of translation quality in a general environment. We perform similar experiments on all data. In some of the sentences, the translation table might not contain the correct translation for source phrase, due to the limitation of the training of our current MT system. The results are listed in Table 7 Experiments on all data, (PR*n denotes system that repeat picking and revising for n cycles; the PE system post edits the most critical error).

Although the BLEU score in general environment is lower than those in ideal environment, the results show basically the same trends as in the previous experiment. The third row (PR*1) in Table 8 shows that picking and revising the most critical error

International Journal of Scientific & Engineering Research, Volume [illegible]
ISSN 2229-5518

can bring around +11 BLEU improvements in both data sets. Three PR-cycles can achieve +17 BLEU improvements (PR*3). Compared to PE methods, our framework still has a significant advantage in the general environment.

Data	BLEU
Baseline	30.64
PE*1	34.18 (+2.54)
PR*1	**41.47 (+10.83))**
PR*2	**45.76 (+15.12)**
PR*3	**48.33 (+17.69)**

Table 7: The TPES and TRS evaluation Analysis example

7. Conclusions and Future Works

The TPES provides statistically significant improvements over existing state-of-the art APE models and produces significantly better translations than GT which is very difficult to beat. This enhancement in translation quality through TPES should reduce human PE effort. Human evaluation exposed that the TPES generated PE translations contain less lexical errors, TPES corrects erroneous word insertions and deletions, and improves word ordering.

We introduced a pick-revise TRS system, TRS, where the users could pick critical translation errors anywhere in the sentence and revise the translation. By correcting the critical error instead of the left most one, our framework could improve the translation quality in a quicker and more efficient way. By using automatic suggestion models, we could reduce human interaction to a single type, either picking or revising. It is also possible to let different human translators to perform different actions. In this case every translator will focus on a single action, which might be easier to train and may have higher efficiency.

In future, we would like to test our system in a real-life translation scenario to analyze productivity gains in a commercial environment. We also want to extend the TPES by including source language knowledge into the network and compare LSTM against GRU hidden units.

In future, the performance of current TRS system is still related to the underlying MT system. Additional improvement could be achieved by supporting other type of interactions, such as reordering operations, or building the system with stronger statistical models.

References

[1] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

[2] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. on the Properties of Neural Machine Translation: EncoderDecoder Approaches. *CoRR*, abs/1409.1259.

[3] Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007b. Rule-Based Translation with Statistical Phrase-Based Post-Editing, In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206.

[4] K. Oazer and I.D. El-Kahlout. Exploring different representational units in English-to-Turkish statistical machine translation. In Proceedings of the Second Workshop on Statistical Machine Translation, pages 25-32. Association for Computational Linguistics, 2007.

[5] Hanna Béchara, Raphaël Rubino, Yifan He, Yanjun Ma, and Josef van Genabith. 2012. An evaluation of statistical post-editing systems applied to RBMT and SMT systems. In Proceedings of COLING 2012. Mumbai, India, pages 215–230. http://www.aclweb.org/anthology/C12-1014.

[6] Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In Proceedings of the First Conference on Machine Translation. Association for Computational Linguistics, Berlin, Germany, pages 131–198. http://www.aclweb.org/anthology/W/W16/W162301.

[7] Santanu Pal, Mihaela Vela, Sudip Kumar Naskar, and Josef van Genabith. 2015. USAAR-SAPE: An

[8] English–Spanish statistical automatic post-editing system. In Proceedings of the Tenth Workshop on Statistical Machine Translation. Association for Computational Linguistics, Lisbon, Portugal, pages 216–221. http://aclweb.org/anthology/W15-3026.

[9] Jindrich Libovický, Jindrich Helcl, Marek Tlustý, Ondrej Bojar, and Pavel Pecina. 2016. CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In Proceedings of the First Conference on Machine Translation. Association for Computational Linguistics, Berlin, Germany, pages 646–654. http://www.aclweb.org/anthology/W/W16/W162361.

[10] Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In Proceedings of the First Conference on Machine Translation. Pages 751– 758. http://www.aclweb.org/anthology/W16-2378.

[11] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Sapporo, Japan, pages 160– 167. http://www.aclweb.org/anthology/P03-1021.

[12] Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, Qun Liu, and Josef van Genabith. 2017. Neural automatic post-editing using prior alignment and reranking. In Proceedings of the European Chapter of the Association for Computational Linguistics. Pages 349–355.

[13] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. ArXiv preprint arXiv: 1302.4389.

[14] Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. ArXiv: 1212.5701 [cs.LG].

International Journal of Scientific & Engineering Research, Volume ÑÛ(ÚÚÜÌ+Û)ÜÕÌıl¥hШ
ISSN 2229-5518

[15] Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation, pages 187–197.

[16] Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 427–436.

[17] Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, pages 160–167.

[18] Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318.

[19] Jindrich Libovický, Jindrich Helcl, Marek Tlustý, Ondrej Bojar, and Pavel Pecina. 2016. CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In Proceedings of the First Conference on Machine Translation. Association for Computational Linguistics, Berlin, Germany, pages 646–654. http://www.aclweb.org/anthology/W/W16/W162361.

[20] Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4012.

[21] Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a Toolkit for Neural Machine Translation. In Proceedings of the Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 2017.

[22] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondˇrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, pages 177–180.

[23] Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318.

[24] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of Translation Edit Rate with Targeted Human Annotation. In Proceedings of association for machine translation in the Americas, pages 223–231.

[25] Behrang Mohit and Rebecca Hwa. 2007. Localization of difficult-to-translate phrases. In Proceedings of the Second Workshop on Statistical Machine Translation, pages 248--255. Association for Computational Linguistics.

[26] Philipp Koehn. 2009. A web-based interactive computer aided translation tool. In Proceedings of the ACLIJCNLP 2009 Software Demonstrations, pages 17--20. Association for Computational Linguistics.

[27] Germán Sanchis-Trilles, Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin L Hill, Philipp Koehn, et al. 2014. Interactive translation prediction versus conventional post-editing in practice: a study with the casmacat workbench. Machine Translation, 28(3-4):217--235.

[28] Benjamin Marie, Lingua ET Machina, France Le Chesnay, and Aurélien Max. 2015. Touch-based pre-postediting of machine translation output. In Conference on Empirical Methods in Natural Language Processing.

[29] Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. Computer Speech & Language, 13(4):359--393.

[30] Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics, pages 311--318. Association for Computational Linguistics.

[31] Le Zhang. 2004. Maximum entropy modeling toolkit for python and c++.

[32] Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3):27.

[33] Amit Agarwal, Eldar Akchurin, Chris Basoglu, Guoguo Chen, Scott Cyphers, Jasha Droppo, Adam Eversole, Brian Guenter, Mark Hillebrand, Xuedong Huang, Zhiheng Huang, Vladimir Ivanov, Alexey Kamenev, Philipp Kranen, Oleksii Kuchaiev, Wolfgang Manousek, Avner May, Bhaskar Mitra, Olivier Nano, Gaizka Navarro, Alexey Orlov, Marko Padmilac, Hari Parthasarathi, Baolin Peng, Alexey Reznichenko, Frank Seide, Michael L. Seltzer, Malcolm Slaney, Andreas Stolcke, Huaming Wang, Kaisheng Yao, Dong Yu, Yu Zhang, and Geoffrey Zweig. 2014. An introduction to computational networks and the computational network toolkit. Technical Report MSR-TR-2014-112, August.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111--3119. Neural Information Processing Systems.

[35] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. The Journal of Machine Learning Research, 3:1137--1155.