CDER: Efficient MT Evaluation Using Block Movements

Gregor Leusch and **Nicola Ueffing** and **Hermann Ney**
Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{leusch,ueffing,ney}@i6.informatik.rwth-aachen.de

Abstract

Most state-of-the-art evaluation measures for machine translation assign high costs to movements of word blocks. In many cases though such movements still result in correct or almost correct sentences. In this paper, we will present a new evaluation measure which explicitly models block reordering as an edit operation. Our measure can be exactly calculated in quadratic time.
Furthermore, we will show how some evaluation measures can be improved by the introduction of word-dependent substitution costs. The correlation of the new measure with human judgment has been investigated systematically on two different language pairs. The experimental results will show that it significantly outperforms state-of-the-art approaches in sentence-level correlation. Results from experiments with word dependent substitution costs will demonstrate an additional increase of correlation between automatic evaluation measures and human judgment.

1 Introduction

Research in machine translation (MT) depends heavily on the evaluation of its results. Especially for the development of an MT system, an evaluation measure is needed which reliably assesses the quality of MT output. Such a measure will help analyze the strengths and weaknesses of different translation systems or different versions of the same system by comparing output at the sentence level. In most applications of MT, understandability for humans in terms of readability as well as semantical correctness should be the evaluation criterion. But as human evaluation is tedious and cost-intensive, automatic evaluation measures are used in most MT research tasks. A high correlation between these automatic evaluation measures and human evaluation is thus desirable.

State-of-the-art measures such as BLEU (Papineni et al., 2002) or NIST (Doddington, 2002) aim at measuring the translation quality rather on the document level[1] than on the level of single sentences. They are thus not well-suited for sentence-level evaluation. The introduction of smoothing (Lin and Och, 2004) solves this problem only partially.

In this paper, we will present a new automatic error measure for MT – the CDER – which is designed for assessing MT quality on the sentence level. It is based on edit distance – such as the well-known word error rate (WER) – but allows for reordering of blocks. Nevertheless, by defining reordering costs, the ordering of the words in a sentence is still relevant for the measure. In this, the new measure differs significantly from the position independent error rate (PER) by (Tillmann et al., 1997). Generally, finding an optimal solution for such a reordering problem is NP hard, as is shown in (Lopresti and Tomkins, 1997). In previous work, researchers have tried to reduce the complexity, for example by restricting the possible permutations on the block-level, or by approximation or heuristics during the calculation. Nevertheless, most of the resulting algorithms still have high run times and are hardly applied in practice, or give only a rough approximation. An overview of some better-known measures can be found in Section 3.1. In contrast to this, our new measure can be calculated very efficiently. This is achieved by requiring complete and disjoint coverage of the blocks only for the reference sentence, and not for the candidate translation. We will present an algorithm which computes the new error measure in quadratic time.

The new evaluation measure will be investigated and compared to state-of-the-art methods on two translation tasks. The correlation with human assessment will be measured for several different statistical MT systems. We will see that the new measure significantly outperforms the existing approaches.

[1] The n-gram precisions are measured at the sentence level and then combined into a score over the whole document.

As a further improvement, we will introduce word dependent substitution costs. This method will be applicable to the new measure as well as to established measures like WER and PER. Starting from the observation that the substitution of a word with a similar one is likely to affect translation quality less than the substitution with a completely different word, we will show how the similarity of words can be accounted for in automatic evaluation measures.

This paper is organized as follows: In Section 2, we will present the state of the art in MT evaluation and discuss the problem of block reordering. Section 3 will introduce the new error measure CDER and will show how it can be calculated efficiently. The concept of word-dependent substitution costs will be explained in Section 4. In Section 5, experimental results on the correlation of human judgment with the CDER and other well-known evaluation measures will be presented. Section 6 will conclude the paper and give an outlook on possible future work.

2 MT Evaluation

2.1 Block Reordering and State of the Art

In MT – as opposed to other natural language processing tasks like speech recognition – there is usually more than one correct outcome of a task. In many cases, alternative translations of a sentence differ from each other mostly by the ordering of blocks of words. Consequently, an evaluation measure for MT should be able to detect and allow for block reordering. Nevertheless, a higher "amount" of reordering between a candidate translation and a reference translation should still be reflected in a worse evaluation score. In other words, the more blocks there are to be reordered between reference and candidate sentence, the higher we want the measure to evaluate the distance between these sentences.

State-of-the-art evaluation measures for MT penalize movement of blocks rather severely: n-gram based scores such as BLEU or NIST still yield a high unigram precision if blocks are reordered. For higher-order n-grams, though, the precision drops. As a consequence, this affects the overall score significantly. WER, which is based on Levenshtein distance, penalizes the reordering of blocks even more heavily. It measures the distance by substitution, deletion and insertion operations for *each* word in a relocated block. PER, on the other hand, ignores the ordering of the words in the sentences completely. This often leads to an overly optimistic assessment of translation quality.

2.2 Long Jumps

The approach we pursue in this paper is to extend the Levenshtein distance by an additional operation, namely block movement. The number of blocks in a sentence is equal to the number of gaps among the blocks plus one. Thus, the block movements can equivalently be expressed as *long jump* operations that jump over the gaps between two blocks. The costs of a long jump are constant. The blocks are read in the order of one of the sentences. These long jumps are combined with the "classical" Levenshtein edit operations, namely *insertion*, *deletion*, *substitution*, and the zero-cost operation *identity*. The resulting *long jump distance* d_{LJ} gives the minimum number of operations which are necessary to transform the candidate sentence into the reference sentence. Like the Levenshtein distance, the long jump distance can be depicted using an alignment grid as shown in Figure 1: Here, each grid point corresponds to a pair of inter-word positions in candidate and reference sentence, respectively. d_{LJ} is the minimum cost of a path between the lower left (first) and the upper right (last) alignment grid point which covers all reference and candidate words. Deletions and insertions correspond to horizontal and vertical edges, respectively. Substitutions and identity operations correspond to diagonal edges. Edges between arbitrary grid points from the same row correspond to long jump operations. It is easy to see that d_{LJ} is symmetrical.

In the example, the best path contains one deletion edge, one substitution edge, and three long jump edges. Therefore, the long jump distance between the sentences is five. In contrast, the best Levenshtein path contains one deletion edge, four identity and five consecutive substitution edges; the Levenshtein distance between the two sentences is six. The effect of reordering on the BLEU measure is even higher in this example: Whereas 8 of the 10 unigrams from the candidate sentence can be found in the reference sentence, this holds for only 4 bigrams, and 1 trigram. Not a single one of the 7 candidate four-grams occurs in the reference sentence.

3 CDER: A New Evaluation Measure

3.1 Approach

(Lopresti and Tomkins, 1997) showed that finding an optimal path in a long jump alignment grid is an NP-hard problem. Our experiments showed that the calculation of exact long jump distances becomes impractical for sentences longer than 20 words.

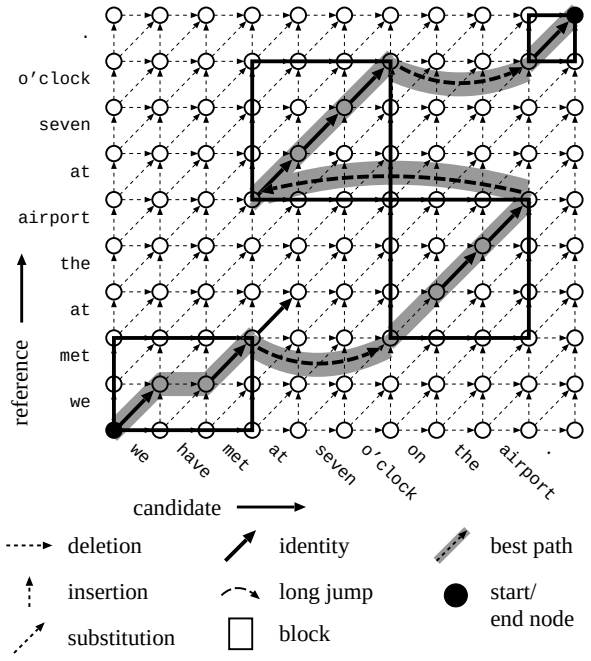

Figure 1: Example of a long jump alignment grid. All possible deletion, insertion, identity and substitution operations are depicted. Only long jump edges from the best path are drawn.

A possible way to achieve polynomial run-time is to restrict the number of admissible block permutations. This has been implemented by (Leusch et al., 2003) in the *inversion word error rate*. Alternatively, a heuristic or approximative distance can be calculated, as in GTM by (Turian et al., 2003). An implementation of both approaches at the same time can be found in TER by (Snover et al., 2005). In this paper, we will present another approach which has a suitable run-time, while still maintaining completeness of the calculated measure. The idea of the proposed method is to drop some restrictions on the alignment path.

The long jump distance as well as the Levenshtein distance require both reference and candidate translation to be covered *completely* and *disjointly*. When extending the metric by block movements, we drop this constraint for the candidate translation. That is, only the words in the reference sentence have to be covered exactly once, whereas those in the candidate sentence can be covered zero, one, or multiple times. Dropping the constraints makes an efficient computation of the distance possible. We drop the constraints for the candidate sentence and not for the reference sentence because we do not want any information contained in the reference to be omitted. Moreover, the reference translation will not contain unnecessary repetitions of blocks.

The new measure – which will be called **CDER** in the following – can thus be seen as a measure oriented towards *recall*, while measures like BLEU are guided by *precision*. The CDER is based on the $\overline{CD}CD$ *distance*[2] introduced in (Lopresti and Tomkins, 1997). The authors show there that the problem of finding the optimal solution can be solved in $O(I^2 \cdot L)$ time, where I is the length of the candidate sentence and L the length of the reference sentence. Within this paper, we will refer to this distance as d_{CD} . In the next subsection, we will show how it can be computed in $O(I \cdot L)$ time using a modification of the Levenshtein algorithm.

We also studied the reverse direction of the described measure; that is, we dropped the coverage constraints for the reference sentence instead of the candidate sentence. Additionally, the maximum of both directions has been considered as distance measure. The results in Section 5.2 will show that the measure using the originally proposed direction has a significantly higher correlation with human evaluation than the other directions.

3.2 Algorithm

Our algorithm for calculating d_{CD} is based on the dynamic programming algorithm for the Levenshtein distance (Levenshtein, 1966). The Levenshtein distance $d_{\mathrm{Lev}}(e_1^I, \tilde{e}_1^L)$ between two strings e_1^I and $\tilde{e}_1^L$ can be calculated in constant time if the Levenshtein distances of the substrings, $d_{\mathrm{Lev}}(e_1^{I-1}, \tilde{e}_1^L)$, $d_{\mathrm{Lev}}(e_1^I, \tilde{e}_1^{L-1})$, and $d_{\mathrm{Lev}}(e_1^{I-1}, \tilde{e}_1^{L-1})$, are known.

Consequently, an auxiliary quantity

$$D_{\mathrm{Lev}}(i,l) := d_{\mathrm{Lev}}\big(e_1^i, \tilde{e}_1^l\big)$$

is stored in an $I \times L$ table. This auxiliary quantity can then be calculated recursively from $D_{\mathrm{Lev}}(i-1,l)$, $D_{\mathrm{Lev}}(i,l-1)$, and $D_{\mathrm{Lev}}(i-1,l-1)$. Consequently, the Levenshtein distance can be calculated in time $O(I \cdot L)$.

This algorithm can easily be extended for the calculation of d_{CD} as follows: Again we define an auxiliary quantity $D(i,l)$ as

$$D(i,l) := d_{\mathrm{CD}}\big(e_1^i, \tilde{e}_1^l\big)$$

Insertions, deletions, and substitutions are handled the same way as in the Levenshtein algorithm. Now assume that an optimal d_{CD} path has been found: Then, each long jump edge within

[2] C stands for *cover* and D for *disjoint*. We adopted this notion for our measures.

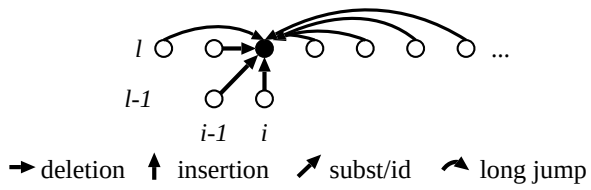

Figure 2: Predecessors of a grid point (i, l) in Equation 1

this path will always start at a node with the lowest D value in its row[3].

Consequently, we use the following modification of the Levenshtein recursion:

$$D(0,0) = 0$$

(1)

$$D(i,l) = \min \left\{ \begin{array}{c} D(i-1,l-1) + (1-\delta(e_i, \tilde{e}_l)), \\ D(i-1,l) + 1, \\ D(i,l-1) + 1, \\ \min_{i'} D(i',l) + 1 \end{array} \right\}$$

where δ is the Kronecker delta. Figure 2 shows the possible predecessors of a grid point.

The calculation of $D(i,l)$ requires all values of $D(i',l)$ to be known, even for $i' > i$. Thus, the calculation takes three steps for each l:

1. For each i, calculate the minimum of the first three terms.

2. Calculate $\min_{i'} D(i',l)$.

3. For each i, calculate the minimum according to Equation 1.

Each of these steps can be done in time $O(I)$. Therefore, this algorithm calculates d_{CD} in time $O(I \cdot L)$ and space $O(I)$.

3.3 Hypothesis Length and Penalties

As the CDER does not penalize candidate translations which are too long, we studied the use of a *length penalty* or *miscoverage penalty*. This determines the difference in sentence lengths between candidate and reference. Two definitions of such a penalty have been studied for this work.

[3]Proof: Assume that the long jump edge goes from (i',l) to (i,l), and that there exists an i'' such that $D(i'',l) < D(i',l)$. This means that the path from $(0,0)$ to (i'',l) is less expensive than the path from $(0,0)$ to (i',l). Thus, the path from $(0,0)$ through (i'',l) to (i,l) is less expensive than the path through (i',l). This contradicts the assumption.

Length Difference

There is always an optimal d_{CD} alignment path that does not contain any deletion edges, because each deletion can be replaced by a long jump, at the same costs. This is different for a d_{LJ} path, because here each candidate word must be covered exactly once. Assume now that the candidate sentence consists of I words and the reference sentence consists of L words, with $I > L$. Then, at most L candidate words can be covered by substitution or identity edges. Therefore, the remaining candidate words (at least $I - L$) must be covered by deletion edges. This means that at least $I - L$ deletion edges will be found in any d_{LJ} path, which leads to $d_{\mathrm{LJ}} - d_{\mathrm{CD}} \geq I - L$ in this case.

Consequently, the *length difference* between the two sentences gives us a useful miscoverage penalty lp_{len}:

$$lp_{\mathrm{len}} := \max\big(I - L, 0\big)$$

This penalty is independent of the d_{CD} alignment path. Thus, an optimal d_{CD} alignment path is optimal for $d_{\mathrm{CD}} + lp_{\mathrm{len}}$ as well. Therefore the search algorithm in Section 3.2 will find the optimum for this sum.

Absolute Miscoverage

Let $\mathrm{coverage}(i)$ be the number of substitution, identity, and deletion edges that cover a candidate word e_i in a d_{CD} path. If we had a complete and disjoint alignment for the candidate word (i.e., a d_{LJ} path), $\mathrm{coverage}(i)$ would be 1 for each i.

In general this is not the case. We can use the *absolute miscoverage* as a penalty lp_{misc} for d_{CD}:

$$lp_{\mathrm{misc}} := \sum_i |1 - \mathrm{coverage}(i)|$$

This miscoverage penalty is not independent of the alignment path. Consequently, the proposed search algorithm will not necessarily find an optimal solution for the sum of d_{CD} and lp_{misc}.

The idea behind the absolute miscoverage is that one can construct a valid – but not necessarily optimal – d_{LJ} path from a given d_{CD} path. This procedure is illustrated in Figure 3 and takes place in two steps:

1. For each block of over-covered candidate words, replace the aligned substitution and/or identity edges by insertion edges; move the long jump at the beginning of the block accordingly.

2. For each block of under-covered candidate words, add the corresponding number of

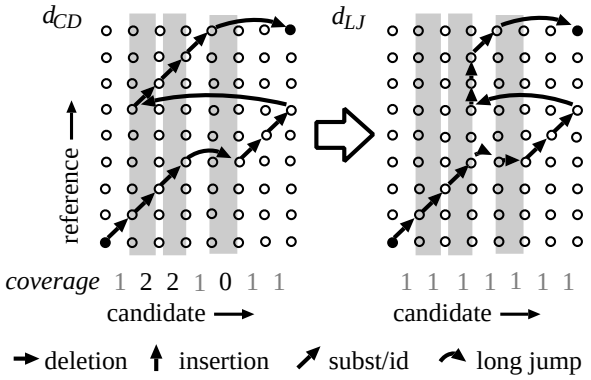

Figure 3: Transformation of a d_{CD} path into a d_{LJ} path.

deletion edges; move the long jump at the beginning of the block accordingly.

This also shows that there cannot be[4] a polynomial time algorithm that calculates the minimum of $d_{\mathrm{CD}} + lp_{\mathrm{misc}}$ for arbitrary pairs of sentences, because this minimum is equal to d_{LJ}.

With these miscoverage penalties, inexpensive lower and upper bounds for d_{LJ} can be calculated, because the following inequality holds:

$$(2) \quad d_{\mathrm{CD}} + lp_{\mathrm{len}} \;\leq\; d_{\mathrm{LJ}} \;\leq\; d_{\mathrm{CD}} + lp_{\mathrm{misc}}$$

4 Word-dependent Substitution Costs

4.1 Idea

All automatic error measures which are based on the edit distance (i.e. WER, PER, and CDER) apply fixed costs for the substitution of words. However, this is counter-intuitive, as replacing a word with another one which has a similar meaning will rarely change the meaning of a sentence significantly. On the other hand, replacing the same word with a completely different one probably will. Therefore, it seems advisable to make substitution costs dependent on the semantical and/or syntactical dissimilarity of the words.

To avoid awkward case distinctions, we assume that a substitution cost function c_{SUB} for two words e, $\tilde{e}$ meets the following requirements:

1. c_{SUB} depends only on e and $\tilde{e}$.
2. c_{SUB} is a metric; especially
   (a) The costs are zero if $e = \tilde{e}$, and larger than zero otherwise.
   (b) The triangular inequation holds: it is always cheaper to replace e by $\tilde{e}$ than to replace e by e' and then e' by $\tilde{e}$.
3. The costs of substituting a word e by $\tilde{e}$ are always equal or lower than those of deleting e and then inserting $\tilde{e}$. In short, $c_{\mathrm{SUB}} \leq 2$.

Under these conditions the algorithms for WER and CDER can easily be modified to use word-dependent substitution costs. For example, the only necessary modification in the CDER algorithm in Equation 1 is to replace $1 - \delta(e, \tilde{e})$ by $c_{\mathrm{SUB}}(e, \tilde{e})$.

For the PER, it is no longer possible to use a linear time algorithm in the general case. Instead, a modification of the Hungarian algorithm (Knuth, 1993) can be used.

The question is now how to define the word-dependent substitution costs. We have studied two different approaches.

4.2 Character-based Levenshtein Distance

A pragmatic approach is to compare the spelling of the words to be substituted with each other. The more similar the spelling is, the more similar we consider the words to be, and the lower we want the substitution costs between them. In English, this works well with similar tenses of the same verb, or with genitives or plurals of the same noun. Nevertheless, a similar spelling is no guarantee for a similar meaning, because prefixes such as "`mis-`", "`in-`", or "`un-`" can change the meaning of a word significantly.

An obvious way of comparing the spelling is the Levenshtein distance. Here, words are compared on character level. To normalize this distance into a range from 0 (for identical words) to 1 (for completely different words), we divide the absolute distance by the length of the Levenshtein alignment path.

4.3 Common Prefix Length

Another character-based substitution cost function we studied is based on the common prefix length of both words. In English, different tenses of the same verb share the same prefix; which is usually the stem. The same holds for different cases, numbers and genders of most nouns and adjectives. However, it does not hold if verb prefixes are changed or removed. On the other hand, the common prefix length is sensitive to critical prefixes such as "`mis-`" for the same reason. Consequently, the common prefix length, normalized by the average length of both words, gives a reasonable measure for the similarity of two words. To transform the normalized common prefix length into costs, this fraction is then subtracted from 1.

Table 1 gives an example of these two word-dependent substitution costs.

[4] provided that $P \neq NP$, of course.

Table 1: Example of word-dependent substitution costs.

e	$\tilde{e}$	Levenshtein		prefix	
		distance	substitution cost	similarity	substitution cost
usual	unusual	2	$\frac{2}{7} = 0.29$	1	$1 - \frac{1}{6} = 0.83$
understanding	misunderstanding	3	$\frac{3}{16} = 0.19$	0	1.00
talk	talks	1	$\frac{1}{5} = 0.20$	4	$1 - \frac{4}{4.5} = 0.11$

4.4 Perspectives

More sophisticated methods could be considered for word-dependent substitution costs as well. Examples of such methods are the introduction of information weights as in the NIST measure or the comparison of stems or synonyms, as in METEOR (Banerjee and Lavie, 2005).

5 Experimental Results

5.1 Experimental Setting

The different evaluation measures were assessed experimentally on data from the Chinese–English and the Arabic–English task of the NIST 2004 evaluation workshop (Przybocki, 2004). In this evaluation campaign, 4460 and 1735 candidate translations, respectively, generated by different research MT systems were evaluated by human judges with regard to fluency and adequacy. Four reference translations are provided for each candidate translation. Detailed corpus statistics are listed in Table 2.

For the experiments in this study, the candidate translations from these tasks were evaluated using different automatic evaluation measures. Pearson's correlation coefficient r between automatic evaluation and the sum of fluency and adequacy was calculated. As it could be arguable whether Pearson's r is meaningful for categorical data like human MT evaluation, we have also calculated Kendall's correlation coefficient τ. Because of the high number of samples (= sentences, 4460) versus the low number of categories (= outcomes of *adequacy+fluency*, 9), we calculated τ separately for each source sentence. These experiments showed that Kendall's τ reflects the same tendencies as Pearson's r regarding the ranking of the evaluation measures. But only the latter allows for an efficient calculation of confidence intervals. Consequently, figures of τ are omitted in this paper.

Due to the small number of samples for evaluation on system level (10 and 5, respectively), all correlation coefficients between automatic and human evaluation on system level are very close to 1. Therefore, they do not show any significant differences for the different evaluation measures. Additional experiments on data from the NIST 2002 and 2003 workshops and from the IWSLT 2004 evaluation workshop confirm the findings from the NIST 2004 experiments; for the sake of clarity they are not included here. All correlation coefficients presented here were calculated for sentence level evaluation. For comparison with state-of-the-art evaluation measures, we have also calculated the correlation between human evaluation and WER and BLEU, which were both measures of choice in several international MT evaluation campaigns. Furthermore, we included TER (Snover et al., 2005) as a recent heuristic block movement measure in some of our experiments for comparison with our measure. As the BLEU score is unsuitable for sentence level evaluation in its original definition, BLEU-S smoothing as described by (Lin and Och, 2004) is performed. Additionally, we added sentence boundary symbols for BLEU, and a different reference length calculation scheme for TER, because these changes improved the correlation between human evaluation and the two automatic measures. Details on this have been described in (Leusch et al., 2005).

Table 2: Corpus statistics. TIDES corpora, NIST 2004 evaluation.

Source language	Chinese	Arabic
Target language	English	English
Sentences	446	347
Running words	13 016	10 892
Ref. translations	4	4
Avg. ref. length	29.2	31.4
Candidate systems	10	5

5.2 CDER

Table 3 presents the correlation of BLEU, WER, and CDER with human assessment. It can be seen that CDER shows better correlation than BLEU and WER on both corpora. On the Chinese–English task, the smoothed BLEU score has a higher sentence-level correlation than WER. However, this is not the case for the Arabic–

Table 3: Correlation (r) between human evaluation (*adequacy + fluency*) and automatic evaluation with BLEU, WER, and CDER (NIST 2004 evaluation; sentence level).

Automatic measure	*Chin.–E.*	*Arab.–E.*
BLEU	0.615	0.603
WER	0.559	0.589
CDER	**0.625**	**0.623**
CDER reversed[a]	0.222	0.393
CDER maximum[b]	0.594	0.599

[a] CD constraints for candidate instead of reference.
[b] Sentence-wise maximum of normal and reversed CDER

Table 4: Correlation (r) between human evaluation (*adequacy + fluency*) and automatic evaluation for CDER with different penalties.

Penalty	*Chin.–E.*	*Arab.–E.*
–	**0.625**	**0.623**
lp_{len}	0.581	0.567
lp_{misc}	0.466	0.528
$(lp_{\text{len}} + lp_{\text{misc}})/2$	0.534	0.557

English task. So none of these two measures is superior to the other one, but they are both outperformed by CDER.

If the direction of CDER is reversed (i.e, the CD constraints are required for the candidate instead of the reference, such that the measure has *precision* instead of *recall* characteristics), the correlation with human evaluation is much lower.

Additionally we studied the use of the maximum of the distances in both directions. This has a lower correlation than taking the original CDER, as Table 3 shows. Nevertheless, the maximum still performs slightly better than BLEU and WER.

5.3 Hypothesis Length and Penalties

The problem of how to avoid a preference of overly long candidate sentences by CDER remains unsolved, as can be found in Table 4: Each of the proposed penalties infers a significant decrease of correlation between the (extended) CDER and human evaluation. Future research will aim at finding a suitable length penalty. Especially if CDER is applied in system development, such a penalty will be needed, as preliminary optimization experiments have shown.

5.4 Substitution Costs

Table 5 reveals that the inclusion of word-dependent substitution costs yields a raise by more than 1% absolute in the correlation of CDER with human evaluation. The same is true for

Table 5: Correlation (r) between human evaluation (*adequacy + fluency*) and automatic evaluation for WER and CDER with word-dependent substitution costs.

Measure	*Subst. costs*	*Chin.–E.*	*Arab.–E.*
WER	const (1)	0.559	0.589
	prefix	0.571	0.605
	Levenshtein	**0.580**	**0.611**
CDER	const (1)	0.625	0.623
	prefix	0.637	0.634
	Levenshtein	**0.638**	**0.637**

WER: the correlation with human judgment is increased by about 2% absolute on both language pairs. The Levenshtein-based substitution costs are better suited for WER than the scheme based on common prefix length. For CDER, there is hardly any difference between the two methods. Experiments on five more corpora did not give any significant evidence which of the two substitution costs correlates better with human evaluation. But as the prefix-based substitution costs improved correlation more consistently across all corpora, we employed this method in our next experiment.

5.5 Combination of CDER and PER

An interesting topic in MT evaluation research is the question whether a linear combination of two MT evaluation measures can improve the correlation between automatic and human evaluation. Particularly, we expected the combination of CDER and PER to have a significantly higher correlation with human evaluation than the measures alone. CDER (as opposed to PER) has the ability to reward correct local ordering, whereas PER (as opposed to CDER) penalizes overly long candidate sentences. The two measures were combined with linear interpolation. In order to determine the weights, we performed data analysis on seven different corpora. The result was consistent across all different data collections and language pairs: a linear combination of about 60% CDER and 40% PER has a significantly higher correlation with human evaluation than each of the measures alone. For the two corpora studied here, the results of the combination can be found in Table 6: On the Chinese–English task, there is an additional gain of more than 1% absolute in correlation over CDER alone. The combined error measure is the best method in both cases.

The last line in Table 6 shows the 95%-confidence interval for the correlation. We see that the new measure CDER, combined with PER, has a significantly higher correlation with human evaluation than the existing measures BLEU, TER,

Table 6: Correlation (r) between human evaluation (*adequacy* + *fluency*) and automatic evaluation for different automatic evaluation measures.

Automatic measure	*Chin.–E.*	*Arab.–E.*
BLEU	0.615	0.603
TER	0.548	0.582
WER	0.559	0.589
WER + Lev. subst.	0.580	0.611
CDER	0.625	0.623
CDER +prefix subst.	0.637	0.634
CDER +prefix+PER	**0.649**	**0.635**
95%-confidence	*±0.018*	*±0.028*

and WER on both corpora.

6 Conclusion and Outlook

We presented CDER, a new automatic evaluation measure for MT, which is based on edit distance extended by block movements. CDER allows for reordering blocks of words at constant cost. Unlike previous block movement measures, CDER can be exactly calculated in quadratic time. Experimental evaluation on two different translation tasks shows a significantly improved correlation with human judgment in comparison with state-of-the-art measures such as BLEU.

Additionally, we showed how word-dependent substitution costs can be applied to enhance the new error measure as well as existing approaches. The highest correlation with human assessment was achieved through linear interpolation of the new CDER with PER.

Future work will aim at finding a suitable length penalty for CDER. In addition, more sophisticated definitions of the word-dependent substitution costs will be investigated. Furthermore, it will be interesting to see how this new error measure affects system development: We expect it to allow for a better sentence-wise error analysis. For system optimization, preliminary experiments have shown the need for a suitable length penalty.

Acknowledgement

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023, and was partly funded by the European Union under the integrated project TC-STAR – Technology and Corpora for Speech to Speech Translation

References

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, MI, Jun.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *ARPA Workshop on Human Language Technology*.

D. E. Knuth, 1993. *The Stanford GraphBase: a platform for combinatorial computing*, pages 74–87. ACM Press, New York, NY.

G. Leusch, N. Ueffing, and H. Ney. 2003. A novel string-to-string distance measure with applications to machine translation evaluation. *MT Summit IX*, pages 240–247, New Orleans, LA, Sep.

G. Leusch, N. Ueffing, D. Vilar, and H. Ney. 2005. Preprocessing and normalization for automatic evaluation of machine translation. *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 17–24, Ann Arbor, MI, Jun.

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

C.-Y. Lin and F. J. Och. 2004. Orange: a method for evaluation automatic evaluation metrics for machine translation. *COLING 2004*, pages 501–507, Geneva, Switzerland, Aug.

D. Lopresti and A. Tomkins. 1997. Block edit models for approximate string matching. *Theoretical Computer Science*, 181(1):159–179, Jul.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, PA, Jul.

M. Przybocki. 2004. NIST machine translation 2004 evaluation: Summary of results. *DARPA Machine Translation Evaluation Workshop*, Alexandria, VA.

M. Snover, B. J. Dorr, R. Schwartz, J. Makhoul, L. Micciulla, and R. Weischedel. 2005. A study of translation error rate with targeted human annotation. Technical Report LAMP-TR-126, CS-TR-4755, UMIACS-TR-2005-58, University of Maryland, College Park, MD.

C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf. 1997. Accelerated DP based search for statistical translation. *European Conf. on Speech Communication and Technology*, pages 2667–2670, Rhodes, Greece, Sep.

J. P. Turian, L. Shen, and I. D. Melamed. 2003. Evaluation of machine translation and its evaluation. *MT Summit IX*, pages 23–28, New Orleans, LA, Sep.