

A Novel String-to-String Distance Measure With Applications to Machine Translation Evaluation

Gregor Leusch, Nicola Ueffing, Hermann Ney

Lehrstuhl für Informatik VI
RWTH Aachen – University of Technology
D-52056 Aachen, Germany,
{leusch, ueffing, ney}@i6.informatik.rwth-aachen.de

Abstract

We introduce a string-to-string distance measure which extends the edit distance by block transpositions as constant cost edit operation. An algorithm for the calculation of this distance measure in polynomial time is presented. We then show how this distance measure can be used as an evaluation criterion in machine translation. Its correlation with human judgment is systematically compared with that of other automatic evaluation measures on two translation tasks.

1 Introduction

One basic task in natural language processing (NLP), as well as other disciplines like computational biology (Waterman, 1995), is comparing sequences of symbols with each other, deciding about their similarity. In NLP, sequences are designated as *sentences*, consisting of *words*.

By common sense, sentences are considered to be the more similar the more words they share and the more their word orders resemble each other. Whereas for applications in speech recognition or optical character recognition reordering is of no consideration, there are applications where one expects reordering of single words and blocks between two sentences, like

- Grammar induction, see e.g. (Vilar, 2000)
- Document retrieval
- Evaluation in Machine Translation (MT)

In this paper, we will propose a distance measure, the *inversion edit distance*, that takes block reordering into account. We will show an application to MT evaluation. The paper will be organized as follows: Section 2 will introduce conventional edit operations and their extension by block transpositions. In Section 3, a formal definition of the inversion edit distance on the basis of bracketing transduction grammars will be given. Furthermore, the search algorithm and its complexity will be described. An application of the inversion edit distance to machine translation evaluation will be presented in Section 4, and experiments on two different corpora will be

given in Section 5. These experiments will be discussed in Section 6, and we will conclude in Section 7.

2 Edit Operations

2.1 Conventional Edit Operations

A common approach to distance measures defines a set of *edit operations*, such as *insertion* or *deletion* of a word, together with a cost for each operation. The distance between two sentences then is defined to be the sum of the costs in the cheapest chain of edit operations transforming one sentence into the other. Having *insertion*, *deletion* and *substitution* as operations, each at the cost of 1, yields the *Levenshtein distance* (Levenshtein, 1966).

Unfortunately, classical edit distances do not correspond well with the consideration that two sentences are similar if just a block changes position:

Consider A, B, C, D to be blocks of words. Assume, B and C do not share words. Then, in order to transform the sentence $ABCD$ into $ACBD$ with Levenshtein operations only, we have to delete all words of B before and insert them behind C (or vice versa), resulting in total costs of $2 \cdot \min\{|B|, |C|\}$. Nevertheless, a penalization of a block move by constant cost only might be sensible as the example in Section 3.2.1 will show.

2.2 Block Transposition as Edit Operation

As a solution to this, we define a *block transposition*, i.e. changing the order of two arbitrary successive blocks, to be a constant-cost edit operation. In

the example presented above, $ABCD$ can then be transformed into $ACBD$ at constant cost.

In order to reduce the complexity of the search, we restrict consequent block transpositions to be *bracketed*, i.e. the two blocks to be swapped must both lie either completely within or completely out of any block from previous operations.

The following examples illustrate allowed and forbidden block transpositions. The brackets indicate the blocks that are swapped. In the transformation of $ABCD$ into $CDBA$ in (1), only transpositions within these blocks are performed. Whereas in (2), the transformation from $BCDA$ into $BDAC$ crosses the blocks BCD and A from the previous transposition and is therefore forbidden.

1. Allowed transpositions:
 $(A)(B\ C\ D) \rightarrow ((B)\ (C\ D))(A)$
 $\rightarrow ((C\ D)\ (B))(A)$
2. Forbidden transpositions:
 $(A)(B\ C\ D) \rightarrow (B\ C\ D)(A)$
 $\rightarrow (B)(D\ A)(C)$

3 The Extended Distance Measure

A concise definition of the edit operations introduced in Section 2 can be formulated using bracketing transduction grammars.

3.1 Bracketing Transduction Grammars

A bracketing transduction grammar (BTG) (Wu, 1995) is a bilingual model that generates two output streams s and t in two languages, called source and target language, respectively. It consists of one common set of production rules for both languages. A BTG always generates a pair of sentences. Terminals are pairs of source and target language symbols, where each may be the empty word ϵ .

Concatenation of the terminals and nonterminals in the right hand side of a production rule is either *straight*, denoted by $[\cdot]$, or *inverted*, denoted by $\langle \cdot \rangle$. In the former case, the parse subtree is to be read left-to-right in both languages, and in the latter case it is to be read left-to-right in the source language and right-to-left in the target language. A BTG contains only the start symbol S and one nonterminal symbol A , and each production rule consists of either a string of A s or a terminal pair.

3.2 Edit Operations as BTG Production Rules

Using the BTG formalism, we can describe the edit operations we have defined in Section 2 as a production rules, associated with a cost function c :

1. Concatenation: $A \rightarrow [AA]$
with $c([\alpha\beta]) = c(\alpha) + c(\beta)$
2. Inversion: $A \rightarrow \langle AA \rangle$
with $c(\langle \alpha\beta \rangle) = c(\alpha) + c(\beta) + c_{inv}$
3. Identity: $A \rightarrow x/x$
with $c(x/x) = 0$
4. Substitution: $A \rightarrow x/y$, where $x \neq y$
with $c(x/y) = c_{subst}$
5. Deletion: $A \rightarrow x/\epsilon$
with $c(x/\epsilon) = c_{del}$
6. Insertion: $A \rightarrow \epsilon/y$
with $c(\epsilon/y) = c_{ins}$
7. Start: $S \rightarrow A; S \rightarrow \epsilon/\epsilon$
with $c(\epsilon/\epsilon) = 0$

The costs c_{inv} , c_{subst} , c_{del} , and c_{ins} are parameters of the edit distance; usually we set all of them to 1.

We define the *inversion edit distance* between a source sentence s_1^I and a target sentence t_1^J to be the minimum cost of the set $T(s_1^I, t_1^J)$ of all trees for this sentence pair:

$$d_{inv}(s_1^I, t_1^J) := \min_{\tau \in T(s_1^I, t_1^J)} c(\tau)$$

Note that, without the inversion rule (2), the minimum production cost equals the Levenshtein distance.

3.2.1 Example

Consider the sentences we will meet at noon in the lobby and we will meet in the lobby at twelve o'clock. Then, $d_{inv} = 3$, as these sentences can be parsed as follows (trivial concatenation brackets not shown):

$$\left[\text{we/we will/will meet/meet} \langle \right. \\ \left. \left[\text{at/at noon/twelve } \epsilon/\text{o'clock} \right] \right. \\ \left. \left[\text{in/in the/the lobby/lobby} \right] \right\rangle$$

We see that the insertion rule, the substitution rule, and the inversion rule are each applied once. The Levenshtein distance d_L of this sentence pair is 5.

3.2.2 Properties

The inversion edit distance has the following properties:

Property 1 For $c_{inv} = c_{subst} = c_{del} = c_{ins} = 1$, d_{inv} is a distance measure.

As no cost is negative, we have $d_{inv}(s_1^I, t_1^J) \geq 0$. Since concatenation and identity are for free, but each other operation has positive cost, ($d_{inv}(s_1^I, t_1^J) = 0 \Leftrightarrow s_1^I = t_1^J$) follows. d_{inv} is symmetric, because all production rules and costs are symmetric. Thus, d_{inv} is a distance measure.

Property 2 For $c_{inv} = c_{subst} = c_{del} = c_{ins} = 1$, d_{inv} is not a metric.

It holds $d_{inv}(abcd, abdc) = 1$ and $d_{inv}(abdc, bdac) = 1$, but we have $d_{inv}(abcd, bdac) = 4 > 2$. Thus, the triangular inequation does not hold, and d_{inv} is not a metric.

3.3 Algorithm

For the calculation of the distance of two sentences $s_{i_0}^{i_1}$ and $t_{j_0}^{j_1}$, we have to determine the cost of the cheapest parse tree in $T(s_{i_0}^{i_1}, t_{j_0}^{j_1})$ generating them.

We can extend the CYK algorithm (Younger, 1967) to the two-dimensional (i. e. bilingual) case. Then, the costs are calculated according to:

- If $i_0 = i_1$ and $j_0 = j_1$, that is $s_{i_0}^{i_1}$ and $t_{j_0}^{j_1}$ both are single words, either the identity or the substitution production will be applied; thus $d_{inv}(s_{i_0}, t_{j_0})$ is zero or c_{subst} , respectively.
- Similar holds if $i_1 < i_0$, that is $s_{i_0}^{i_1} = \epsilon$. Then, ϵ and $t_{j_0}^{j_1}$ can only be generated by $j_1 - j_0 + 1$ applications of the concatenation and the insertion rule, thus $d_{inv}(\epsilon, t_{j_0}^{j_1}) = (j_1 - j_0 + 1) \cdot c_{ins}$.
- Analogously, if $j_1 < j_0$, the deletion rule has to be applied $i_1 - i_0 + 1$ times, thus $d_{inv}(s_{i_0}^{i_1}, \epsilon) = (i_1 - i_0 + 1) \cdot c_{del}$.
- In all other cases, either the concatenation or the inversion production rule will be applied, hence the tree's cost is the sum of two subtrees' costs. For concatenation of blocks, we obtain

$$d_{inv}(s_{i_0}^{i_1}, t_{j_0}^{j_1}) = \min_{i', j'} \min_{\substack{\tau \in T(s_{i_0}^{i'}, t_{j_0}^{j'}) \\ \tau' \in T(s_{i'+1}^{i_1}, t_{j'+1}^{j_1})}} \{c(\tau) + c(\tau')\}$$

and for inversion, we obtain

$$d_{inv}(s_{i_0}^{i_1}, t_{j_0}^{j_1}) = \min_{i', j'} \min_{\substack{\tau \in T(s_{i_0}^{i'}, t_{j'+1}^{j_1}) \\ \tau' \in T(s_{i'+1}^{i_1}, t_{j_0}^{j'})}} \{c(\tau) + c(\tau') + c_{inv}\}$$

3.4 Auxiliary Quantity

To apply dynamic programming, we define an auxiliary quantity Q for the recursive calculation of the cost of the cheapest parse tree:

$$Q(i_0, i_1, j_0, j_1) := \text{minimum cost for transforming block } s_{i_0}^{i_1} \text{ into } t_{j_0}^{j_1}$$

$$\text{Then, } Q(i_0, i_1, j_0, j_1) = \left(\begin{array}{l} (j_1 - j_0 + 1) \cdot c_{ins} \quad \text{if } i_1 < i_0 \\ (i_1 - i_0 + 1) \cdot c_{del} \quad \text{if } j_1 < j_0 \\ (1 - \delta(s_{i_0}, t_{j_0})) \cdot c_{subst} \quad \text{if } (i_1 = i_0) \\ \quad \quad \quad \quad \quad \quad \quad \quad \wedge (j_1 = j_0) \\ \min_{\substack{i_0 \leq i' \leq i_1 \\ j_0 \leq j' \leq j_1}} \left\{ \begin{array}{l} Q(i_0, i', j_0, j') \\ \quad + Q(i'+1, i_1, j'+1, j_1), \\ c_{inv} + Q(i_0, i', j'+1, j_1) \\ \quad + Q(i'+1, i_1, j_0, j') \\ \text{otherwise} \end{array} \right\} \end{array} \right) \quad (1)$$

where $\delta(\cdot, \cdot)$ is the Kronecker function. Finally, $d_{inv}(s_1^I, t_1^J) = Q(1, I, 1, J)$.

Note that Q can be viewed as a two-dimensional extension of the two-dimensional CYK algorithm cost table. A nonterminal matching table is not necessary here, as A will always match and is the only nonterminal¹.

3.5 Complexity of the Algorithm

A naïve approach to the calculation of $Q(1, I, 1, J)$ would be the recursive calculation according to the formula in Eq. 1. This procedure is much too expensive. Instead, we can – analogously to the original CYK algorithm – fill the Q table using dynamic programming, dovetailing on the block lengths $i_1 - i_0$ and $j_1 - j_0$. We have to fill a table of size $O(I^2 J^2)$, running over $O(IJ)$ pairs of split points (i', j') for each table entry. This yields a time complexity of $O(I^3 J^3)$ for this approach.

We found that in most cases it is not necessary to calculate all values of $Q(i_0, i', j_0, j')$. Thus, we implemented a recursive approach with memoization (Norvig, 1991), i. e. caching of all previously calculated table entries of Q . This algorithm has the same worst case complexity $O(I^3 J^3)$, but performs much better in average case. This is due to the fact that we can prune many subtrees of the search tree after having estimated or calculated the first term in the sum.

¹except for S , of course

4 An Application to MT Evaluation

4.1 Introduction

Research in MT depends on the evaluation of MT system results. The progress in the development of a system is to be measured or different systems are to be compared on the basis of test corpora.

In most applications, the translations generated by an MT system are eventually intended to be used by humans. Consequently, manually assigned scores are considered as gold standard for evaluation. In order to evaluate an MT system, a set $\{t^{(i)}\}_{i=1}^n$ of translations generated by the system, called *candidate sentence set*, is evaluated by human experts. Unfortunately, manual evaluation is very expensive in time and money. Several suggestions have been made to simplify and accelerate this task, while at the same time reproducibility and reliability are improved. But manual evaluation still requires 30 to 60 seconds *per sentence* even for easy tasks (Nießen et al., 2000). Thus, the manual evaluation of a candidate sentence set, which usually contains hundreds or even thousands of sentences, takes several hours.

For this reason, a number of automatic measures have been proposed, which provide cheap and reproducible results. To evaluate a candidate sentence set using an automatic measure, each sentence is compared to a set of reference translations $\mathcal{R}^{(i)}$. Usually, there is more than one reference translation for a sentence, as there is more than one way to translate it correctly. The evaluation measure either pools these reference translations, or it is calculated against the most similar reference sentence.

Unfortunately, automatic evaluation measures depend heavily on the choice of reference translations. Furthermore, automatic measures can only decide on words and phrases, and not whether the meaning of sentences is captured or not.

Following these considerations, MT research would benefit from an automatic measure which strongly correlates with human judgment.

4.2 Automatic Measures

4.2.1 BLEU

(Papineni et al., 2001) introduced an MT evaluation measure which they called BLEU (*BiLingual Evaluation Understudy*). For each candidate sentence $t^{(i)}$, a modified n -gram precision is calculated with respect to its pooled reference sentences $\mathcal{R}^{(i)}$. The n -gram lengths range from 1 to N , where typically $N = 4$. To penalize overgeneration of common

n -grams in a candidate sentence, the n -gram count is limited to the corresponding maximum n -gram count in its reference sentences. Then, the geometric mean of these N precisions is calculated.

The precision alone would favor systems that produce short and simple sentences, even if parts of the translation are omitted. To avoid this, sentences which are shorter than the next-in-length reference are assigned a brevity penalty.

The calculation of the geometric mean and the penalizing is carried out on the whole candidate set (and not sentence-wise), thus implicitly weighting each sentence by its length. To investigate the effect of this implicit weighting, we also calculated the arithmetic mean of BLEU of each sentence (weighted and unweighted). We denote this measure by *avgBLEU*.

(NIST, 2002a) proposed a measure similar to BLEU, introducing a different brevity penalty and exchanging the n -gram precision by information weight. We have not conducted further experiments regarding this measure, but we expect it to behave similarly.

4.2.2 Word Error Rate

The word error rate (WER), which is calculated as the length-normalized Levenshtein distance to a reference sentence, has been used in several NLP areas and related disciplines. (Nießen et al., 2000) presented an application to MT evaluation using the multiple reference technique described in Section 4.1. The WER of a test set is calculated by determining the totalized Levenshtein distance between each candidate sentence and its nearest reference sentence and normalizing this by the totalized reference length:

$$\text{WER}(\{t^{(i)}\}, \{\mathcal{R}^{(i)}\}) = \frac{\sum_i \min_{r \in \mathcal{R}^{(i)}} d_L(t^{(i)}, r)}{\sum_i \frac{1}{|\mathcal{R}^{(i)}|} \cdot \sum_{r \in \mathcal{R}^{(i)}} |r|}$$

This implicitly weights each sentence by its length as well.

4.2.3 Position-Independent Error Rate

The position-independent error rate (PER) is similar to the WER, but uses a position independent Levenshtein distance (bag-of-word difference) instead; i. e. the distance between a sentence and one of its permutations is always 0. Note that therefore this is technically not a distance measure.

4.2.4 Inversion Word Error Rate

As the distance measure we have defined in Section 2.2 is an extension of the Levenshtein distance, we can introduce the new evaluation measure as an extension of the WER, where $d_L(t^{(i)}, r)$ is exchanged by $d_{inv}(t^{(i)}, r)$.

5 Experiments

We performed experiments on two different test corpora. For both of them, several candidate sets were produced by different MT systems, which were then manually evaluated sentence-wise. We calculated the PER, the WER, the invWER, and BLEU for each candidate set. These automatic evaluation scores were compared with the manual evaluation. This comparison was done as well on the sentence level as on the level of the whole test set. In the latter case, we compared the unweighted averages of the evaluation scores of the sentences as well as the averages weighted by sentence length (which the automatic measures do implicitly; see Section 4.2). BLEU and the human scores are accuracy measures, whereas PER, WER and invWER are error measures. Thus we inverted the latter three and rescaled such that all measures range from 0.0 (worst) to 1.0 (best).

5.1 German-English

We performed experiments on a German-English test corpus from the Verbmobil project. This corpus contains 342 sentences from the domain of tourism and appointment scheduling. It consists of transcriptions of spontaneously spoken dialogues, and the sentences often lack correct syntactic structure. We collected 898 reference translations from different translators, averaging to 2.63 reference translations per sentence. The average reference sentence length is 12.2.

We evaluated 22 candidate sets from two MT research systems, which were produced using different parameter sets, pre-/postprocessing steps and training corpus size.

Figure 1 shows the distribution of the automatic evaluation scores versus human judgment on the sentence level. The human evaluators assigned 11 quality classes ranging from 0.0 (worst) to 1.0 (best) in steps of 0.1; see (Nießen et al., 2000) for a description of this measure. We see that all automatic evaluation scores correlate well with the manual score. Nevertheless, the standard deviation of the automatic evaluation scores within each manual evaluation class is rather large.

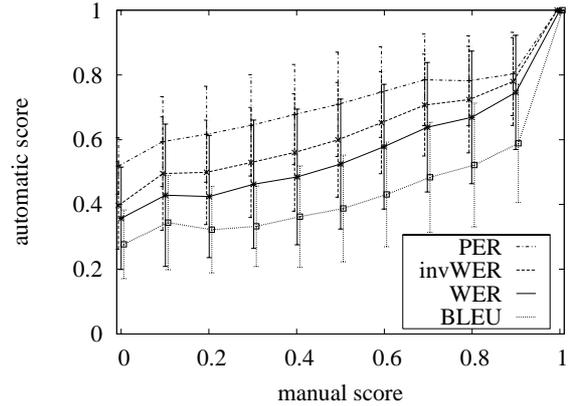


Figure 1: *German-English*: Sentence level comparison of different automatic evaluation scores versus manual evaluation; averaged. Each bar shows the standard deviation within the averaged range.

Table 1: *German-English*: Correlation between the manual and automatic scores; calculated at sentence level and system level. At system level, all scores were compared both weighted by sentence length and unweighted.

	sentence	system	
		weighted	unweighted
PER	0.61	0.85	0.85
WER	0.65	0.98	0.98
invWER	0.68	0.95	0.95
BLEU	0.70	0.97	0.98
avgBLEU	-	0.96	0.96

Figure 2 shows the distribution of the automatic evaluation scores on the system level. The manual evaluation score was calculated as the average sentence evaluation score, weighted by the (average) reference sentence length. Again, the three scores show a similar behavior; and the correlation with human judgment is very high.

Comparing the correlation between the automatic and manual scores numerically, as presented in Table 1, we see that the sentence level correlation values range between 0.65 and 0.70 for all systems. BLEU has the highest correlation, followed by the invWER. On system level, all correlation values range between 0.95 and 0.98, here the WER being the best, followed by BLEU. Neither in tendency nor in the correlation values we find a remarkable difference between the weighted and the unweighted

Table 2: *German–English*: Ranking of the systems according to the different (weighted) automatic scores. S_1, \dots, S_{22} are numbered according to (weighted) manual evaluation: S_1 is the best system, S_{22} is worst. c_R is the ranking correlation

Measure	Ranking	c_R
PER	$S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_{11} S_{12} S_{13} S_9 S_8 S_{10} S_{15} S_{14} S_{16} S_{19} S_{18} S_{20} S_{21} S_{22} S_{17}$	0.92
WER	$S_1 S_2 S_3 S_5 S_4 S_6 S_7 S_{12} S_{11} S_{14} S_{13} S_8 S_9 S_{10} S_{15} S_{16} S_{19} S_{18} S_{17} S_{20} S_{21} S_{22}$	0.95
invWER	$S_1 S_2 S_3 S_4 S_6 S_5 S_7 S_{12} S_{11} S_{13} S_9 S_8 S_{10} S_{14} S_{15} S_{16} S_{19} S_{18} S_{20} S_{17} S_{21} S_{22}$	0.96
BLEU	$S_1 S_2 S_3 S_5 S_4 S_6 S_7 S_{12} S_{11} S_{13} S_9 S_8 S_{10} S_{14} S_{15} S_{16} S_{19} S_{18} S_{17} S_{20} S_{21} S_{22}$	0.96
avgBLEU	$S_1 S_2 S_3 S_7 S_{12} S_6 S_4 S_5 S_9 S_{11} S_8 S_{10} S_{13} S_{14} S_{15} S_{16} S_{19} S_{18} S_{20} S_{17} S_{21} S_{22}$	0.94

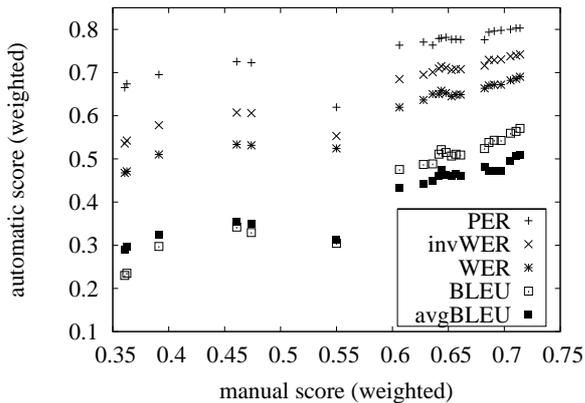


Figure 2: *German–English*: System level comparison of different automatic evaluation scores versus manual evaluation. Each score is weighted by sentence length (implicitly or explicitly).

system-level scores.

In Table 2, we see that the rankings of the 22 systems implied by the automatic scores highly correlate with the manual ranking. On the other hand, small scale differences of similar systems need not be judged equally by the automatic and manual evaluation scores. This may cause problems if small changes in the parameter setting of an MT system are to be evaluated: An improvement according to manual evaluation might be a deterioration according to an automatic score and vice versa.

5.2 Chinese-English

The Chinese–English test corpus along with manual evaluation scores was obtained from the NIST MT evaluation 2002 (NIST, 2002b). Originally, the test corpus consists of 100 Chinese newspaper articles, summing up to 878 sentences. Out of these sentences, we selected all sentences for which the maximum length of all candidate and reference sen-

tences is 50 words or below; leaving us with 657 test sentences in total. Each sentence has been provided with four reference translations, generated by different human translators. The average reference sentence length is 23.5.

Six different research MT systems and three commercial MT systems generated nine candidate sets for this test corpus. Each sentence was evaluated by two or three out of eleven evaluators, judging fluency and adequacy, each from 1 to 5 in steps of 1. For each evaluator, we normalized the fluency and adequacy judgements such that the mean of all judgements was 0.0 and the variance 1.0 (over all sentences, documents and systems). Then, for each sentence, we compared the mean fluency and mean adequacy out of its two or three judgements.

We normalized each reference and candidate sentence by case conversion, whitespace trimming and punctuation separation before the automatic evaluation process.

This test corpus is a lot more difficult for the MT systems than the German–English task, as is reflected in the fact that only one (!) of the 5913 candidate sentences matches its reference translation. Most interestingly, this translation was rated 3.5 out of 5 in fluency and 4.5 out of 5 in adequacy by the human evaluators, showing that the choice of appropriate reference translations must be cared for.

Figure 3 shows the distribution of the automatic evaluation scores versus manual evaluation on the sentence level. Again, WER and invWER behave rather similarly, even though the difference is bigger than in Figure 1. Both WER and invWER have a rather large standard deviation which even increases for higher manual scores. BLEU has a lower standard deviation, but we notice a very small total rise from 0.22 to 0.46 in the BLEU score over all manual evaluation classes.

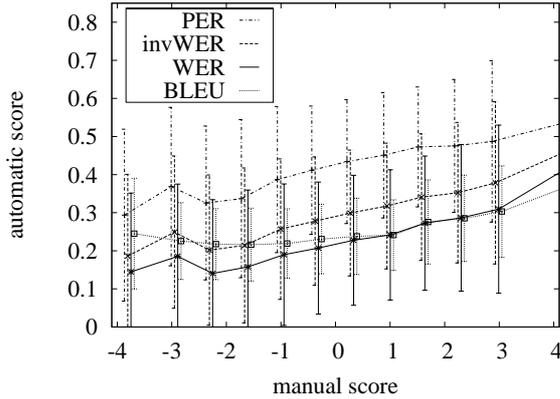


Figure 3: *Chinese-English*: Sentence level comparison of different automatic evaluation scores versus manual evaluation; averaged. Each bar shows the standard deviation within the averaged range.

Figure 4 shows the distribution of the automatic evaluation scores on the system level. The manual evaluation score was calculated as the average sentence evaluation scores, weighted by the (average) reference sentence length. This time, we notice that the correlation between the automatic and the manual evaluation score is very small.

Table 3: *Chinese-English*: Correlation between the manual and automatic scores; calculated at sentence level and system level. At system level, all scores were compared both weighted by sentence length and unweighted.

	sentence	system	
		weighted	unweighted
BLEU	0.26	0.28	0.25
avgBLEU	-	0.15	0.13
PER	0.26	0.15	0.14
WER	0.27	0.04	0.02
invWER	0.28	0.09	0.07

Table 3 confirms our observations: The correlation on the sentence level is acceptable with all three scores, where this time the invWER is leading. On the system level, the correlation is close to zero for the WER and the invWER. For nine observed values, we would expect a much higher correlation if the scores were related. Assuming a normal distribution for both types of score, even an empirical correlation of 0.28 would by far not prove (with a confi-

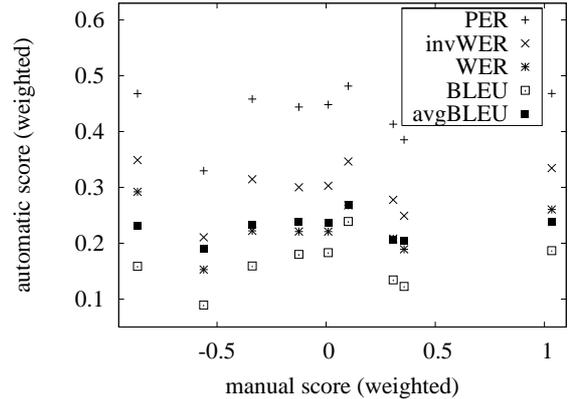


Figure 4: *Chinese-English*: System level comparison of different automatic evaluation scores versus manual evaluation. Each score is weighted by sentence length (implicitly or explicitly).

dence of 90%) that BLEU and the manual score are not independent variables.

Table 4 shows the ranking of the systems according to different automatic evaluation scores. We see that the rankings are similar, but on this task, all three are significantly different from the ranking according to manual evaluation.

6 Discussion

Comparing the correlation between automatic and manual evaluation, we find significant differences between the two translation tasks presented above: On the German-English corpus, all three automatic evaluation scores have a high correlation with human judgment, whereas on the Chinese-English task, the correlation is very poor.

We assume that one reason for this is that the

Table 4: *Chinese-English*: Ranking of the systems according to the different (weighted) automatic scores. S_1, \dots, S_9 are numbered according to (weighted) manual evaluation: S_1 is the best system, S_9 is worst. c_R is the ranking correlation

Measure	Ranking	c_R
BLEU	$S_4 S_1 S_5 S_6 S_7 S_9 S_3 S_2 S_8$	0.30
avgBLEU	$S_4 S_5 S_1 S_7 S_6 S_9 S_2 S_3 S_8$	0.28
PER	$S_4 S_1 S_9 S_7 S_5 S_6 S_3 S_2 S_8$	0.10
WER	$S_9 S_4 S_1 S_7 S_5 S_6 S_3 S_2 S_8$	0.13
invWER	$S_9 S_4 S_1 S_7 S_5 S_6 S_3 S_2 S_8$	0.12

German–English corpus is relatively easy compared to Chinese–English. We conclude that for tasks with an acceptable system performance, the automatic measures are well suited for judging translation quality. Another reason for the high correlation might be that the candidate sets investigated for German–English were generated by very similar systems. This suggests that automatic evaluation is appropriate for the comparison of improvement within one system or between similar systems. On the other hand, for the comparison of different MT approaches, manual evaluation should not be replaced by automatic evaluation, as especially the ranking for the Chinese–English task in Table 4 shows.

7 Conclusion

We have presented a new distance measure, the inversion edit distance d_{inv} , for comparison of sequences of symbols. The classical Levenshtein distance has been extended by block transpositions in order to allow for moves of word blocks at constant cost.

By the definition of the inversion edit distance as the parse cost of the sentence pair within a simple bilingual grammar, we have given it a sound theoretical background. Using this definition, we have shown that all distance axioms hold. Furthermore, we have introduced an algorithm to calculate the inversion edit distance in worst-case polynomial time, which is related to the CYK algorithm.

We then have shown an application of the inversion edit distance in evaluation of machine translation. Its behavior versus human judgment has been compared with other automatic evaluation measures on two different translation tasks. We have demonstrated that the correlation of this evaluation measure with human judgment is comparable with that of other automatic evaluation measures.

8 Acknowledgments

This work was partly supported by the projects LC-STAR (IST-2001-32216), PF-STAR (IST-2001-37599) and TransType2 (IST-2001-32091) by the European Community.

We would like to thank NIST for the provision of test data and manual evaluation data.

9 References

- V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–71, February.
- Sonja Nießen, Franz J. Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. In *Proc. of the Second Int. Conf. on Language Resources and Evaluation*, pages 39–45, Athens, Greece, May.
- NIST. 2002a. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. <http://nist.gov/speech/tests/mt/>.
- NIST. 2002b. MT Evaluation Chinese–English. <http://nist.gov/speech/tests/mt/>.
- Peter Norvig. 1991. Techniques for automatic memoization with applications to context-free parsing. *Computational Linguistics*, 17(1):93–98.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, September.
- Juan Miguel Vilar. 2000. Improve the learning of subsequential transducers by using alignments and dictionaries. In Arlindo de Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Artificial Intelligence*, pages 298–311, Lisbon, Portugal, September. Springer-Verlag.
- Michael S. Waterman. 1995. *Introduction to computational biology. Maps, sequences and genomes*. Chapman and Hall, XVI edition.
- Dekai Wu. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proc. 33th Annual Meeting of the Assoc. for Computational Linguistics*, pages 244–251.
- D. Younger. 1967. Recognition and parsing of context-free languages in time n^3 . *Information and Control*, 10(2):189–208.