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 demonstrate how this distance measure can be used as an evaluation criterion in machine translation. The correlation between this evaluation criterion and human judgment is systematically compared with that of other automatic evaluation measures on two translation tasks. In general, like other automatic evaluation measures, the criterion shows low correlation at sentence level, but good correlation at system level.

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*.

In a first approximation, 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 reordering of single words and blocks between two sentences can be expected, such as

- Grammar induction, see e.g. (Vilar, 2000)
- 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 demonstrate 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).

Classical edit distances do not correspond well with the consideration that two sentences are similar if a block of words just 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 and insert them after C (or vice versa), resulting in total costs of $2 \cdot \min\{|B|, |C|\}$. Nevertheless, a penalty of a block move by constant cost, i.e. a cost independent of the block length, might be sensible as the example in Section 3.2.1 will show.

2.2 Block Transposition as Edit Operation

As a solution to the problem described above, 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 admissible 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. 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) →  ((B) (C D))(A)
              →  ((C D) (B))(A)
   ```
2. Forbidden transpositions:
   ```
   (A)(B C D) →  (B C D)(A)
              ↛  (B)(D A)(C)
   ```

3 The Extended Distance Measure

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

3.1 Bracketing Transduction Grammars

A bracketing transduction grammar (BTG) (Wu, 1995) is a pair-of-string model that generates two output strings s and t. It consists of one common set of production rules for both output strings. A BTG always generates a pair of sentences. Terminals are pairs of symbols, where each may be the empty word ϵ.

Concatenation of the terminals and nonterminals on 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 s and t, and in the latter case it is to be read left-to-right in s and right-to-left in t. A BTG contains only the start symbol S and one nonterminal symbol A, and each production rule consists of either a string of As 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 production rules, associated with a cost function c:

1. Concatenation: $A \to [AA]$
   with $c([\alpha\beta]) = c(\alpha) + c(\beta)$

2. Inversion: $A \to \langle AA \rangle$
   with $c(\langle\alpha\beta\rangle) = c(\alpha) + c(\beta) + c_{inv}$

3. Identity: $A \to x/x$
   with $c(x/x) = 0$

4. Substitution: $A \to x/y$, where $x \neq y$
   with $c(x/y) = c_{sub}$

5. Deletion: $A \to x/\epsilon$
   with $c(x/\epsilon) = c_{del}$

6. Insertion: $A \to \epsilon/y$
   with $c(\epsilon/y) = c_{ins}$

7. Start: $S \to A$; $S \to \epsilon/\epsilon$
   with $c(\epsilon/\epsilon) = 0$

c_{inv}, c_{sub}, c_{del}, and c_{ins} are parameters of the edit distance; usually we set all of them to 1. α and β are parse subtrees, x and y are terminal symbols.

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 parse trees generated by the BTG for this sentence pair:

$$d_{inv}(s_1^I, t_1^J) := \min_{\tau \in T(s_1^I, t_1^J)} c(\tau) \qquad (1)$$

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

3.2.1 Example

Consider the sentence pair
`we will meet at noon in the lobby` /
`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 omitted):

```
[we/we will/will meet/meet ⟨
    [at/at noon/twelve ε/o'clock]
    [in/in the/the lobby/lobby]    ⟩]
```

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

3.2.2 Properties

If all costs are set to 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, it holds $(d_{inv}(s_1^I, t_1^J) = 0 \Leftrightarrow s_1^I = t_1^J)$. Finally, d_{inv} is symmetric, because all production rules and costs are symmetric.

The triangular inequality does not hold, as a counter-example proves: $d_{inv}(\texttt{abcd}, \texttt{abdc}) = 1$ and $d_{inv}(\texttt{abdc}, \texttt{bdac}) = 1$, but we have $d_{inv}(\texttt{abcd}, \texttt{bdac}) = 4 > 2$. Consequently d_{inv} is not a metric.

3.3 Algorithm

For the calculation of the distance of two partial symbol sequences $s_{i_0}^{i_1}$ and $t_{j_0}^{j_1}$, we have to determine the cost of the cheapest parse tree in all parses tree $T(s_{i_0}^{i_1}, t_{j_0}^{j_1})$ that generate these sequences.

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

- 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_{sub}, respectively.

- If $i_1 < i_0$, $s_{i_0}^{i_1} = \epsilon$ 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 include the sum of two subtrees' costs. For concatenation of blocks, we obtain

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

and for inversion, we obtain

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

3.4 Dynamic Programming Recursion

We define an auxiliary quantity $Q(\cdot)$ for the recursive calculation of the cost of the cheapest parse tree:

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

Then we have the dynamic programming recursion:

$$Q(i_0, i_1; j_0, j_1) = \qquad (2)$$

$$= \begin{cases} (j_1 - j_0 + 1) \cdot c_{ins} & \text{if } i_1 < i_0 \\ (i_1 - i_0 + 1) \cdot c_{del} & \text{if } j_1 < j_0 \\ (1 - \delta(s_{i_0}, t_{j_0})) \cdot c_{sub} & \text{if } (i_1 = i_0) \\ & \qquad \wedge (j_1 = j_0) \\ \min_{\substack{i_0 \le i' \le i_1 \\ j_0 \le j' \le j_1}} \begin{cases} Q(i_0, i'; j_0, j') \\ +Q(i'+1, i_1; j'+1, j_1), \\ c_{inv} + Q(i_0, i'; j'+1, j_1) \\ +Q(i'+1, i_1; j_0, j') \end{cases} \\ \qquad\qquad\qquad \text{otherwise} \end{cases}$$

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

Note that $Q(\cdot)$ can be viewed as a four-dimensional extension of the two-dimensional CYK algorithm cost table. Since there is only a single nonterminal A, the two-dimensional parsing table $Q(\cdot)$ does not to be dependent on any nonterminal.

3.5 Complexity of the Algorithm

Analogously to the original CYK algorithm, the $Q(\cdot)$ table can be filled using dynamic programming. A table of size $O(I^2 J^2)$ has to be filled; $O(IJ)$ pairs of split points (i', j') must be taken into account 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 $Q(i_0, i'; j_0, j')$. We implemented a memoization approach (Norvig, 1991), i. e. caching of all previously calculated table entries of $Q(\cdot)$. This algorithm has the same worst case complexity $O(I^3 J^3)$, but performs much better in the 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.

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 $\{T_k\}_{k=1}^n$ 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 evaluation measures have been proposed, which provide cheap and reproducible results. To evaluate a candidate sentence set using an automatic evaluation measure, each sentence is compared to a set of reference translations $\mathcal{R}_k$. Usually, there is more than one

reference translation for a sentence, since there is more than one way to translate it correctly. The automatic evaluation measure either pools these reference translations, or it is calculated against the most similar reference sentence.

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

From these considerations, it is clear that MT research would benefit from an automatic evaluation measure which strongly correlates with human judgment.

4.2 Automatic Measures

4.2.1 BLEU

(Papineni et al., 2002) introduced an MT evaluation measure which they called BLEU *(BiLingual Evaluation Understudy)*. For each candidate sentence T_k, a modified n-gram precision is calculated with respect to its pooled reference sentences $\mathcal{R}_k$. The n-gram lengths range from 1 to 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 four 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 problem, 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 replacing the n-gram precision by information weight and the geometric mean by the arithmetic mean.

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 tasks 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 $d_L(\cdot)$

between each candidate sentence and its nearest reference sentence and normalizing this by the totalized reference length:

$$\mathrm{WER}(\{T_k\}, \{\mathcal{R}_k\}) = \frac{\sum_k \min_{r \in \mathcal{R}_k} d_L(T_k, r)}{\sum_k \frac{1}{|\mathcal{R}_k|} \cdot \sum_{r \in \mathcal{R}_k} |r|} \quad (3)$$

This definition 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 based distance) instead; i. e. the distance between a sentence and one of its permutations is always zero. Therefore, PER is technically not a distance measure.

4.2.4 Inversion Word Error Rate

The distance measure we have defined in Section 2.2, d_{inv}, is an extension of the Levenshtein distance. Thus we can introduce a new automatic evaluation measure as an extension of the WER by exchanging $d_L(t^{(i)}, r)$ by $d_{inv}(t^{(i)}, r)$ in Eq. 3. We call this measure *inversion word error rate (invWER)*.

It is interesting to compare the latter three measures with respect to their reordering constraints: WER does not admit any changes in order, PER does not put any constraints on reordering, and invWER takes an intermediate position between WER and PER by allowing recursive block inversions.

5 Experimental results

We performed experiments on two different test corpora. For both of them, several candidate sets were generated by different MT systems, which were then manually evaluated sentence-wise. We calculated PER, WER, invWER, and BLEU for each candidate set. These automatic evaluation scores were compared with the manual evaluation scores. This comparison was performed at the sentence level and at the system level, i.e. at the level of of whole candidate sets. In the latter case, we compared the unweighted averages of the scores of the sentences as well as the averages weighted by sentence length (which the automatic evaluation measures do implicitly; see Section 4.2). BLEU and the manual evaluation scores are accuracy measures, whereas PER, WER and invWER are error measures. Thus we rescaled the latter three 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 (Wahlster, 2000). 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 words.

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

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. A manual evaluation score was calculated as the average sentence evaluation score, weighted by the average reference sentence length.

In Figure 1, the distribution of the automatic versus manual evaluation scores at sentence level is shown. Bars indicate the standard deviation. All automatic evaluation scores correlate well with the manual score, though the standard deviation of the automatic evaluation scores within each manual evaluation class is rather large.

Figure 2 represents the distribution of the automatic evaluation scores at system level. Again, the three scores show a similar behavior; and the correlation with human judgment is very high. It stands out that the system level correlation coefficient is significantly higher than the sentence level correlation coefficient for all automatic evaluation measures.

Figure 1: *German–English (VERBMOBIL):* Sentence level comparison of automatic and manual scores (averaged for each manual score interval).

Figure 2: *German–English (VERBMOBIL):* System level comparison of automatic and manual scores. Scores are weighted by sentence length.

Table 1: *German–English (VERBMOBIL):* Correlation coefficients between automatic and manual scores at the sentence and system level (weighted and unweighted scores at the system level).

evaluation measure	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

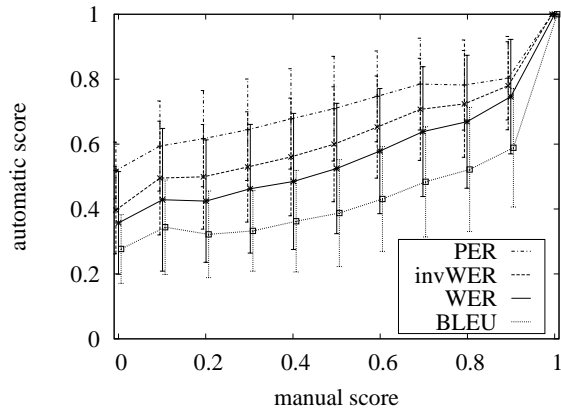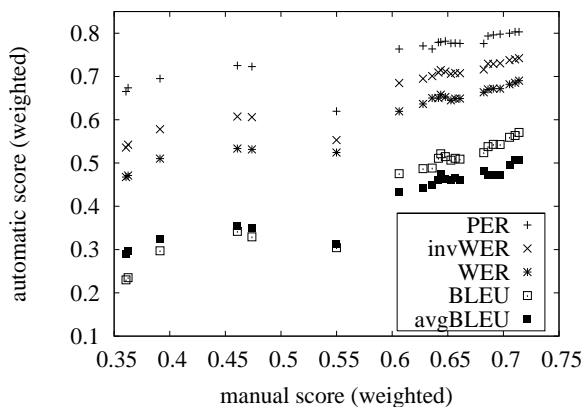Comparing the correlation between automatic and manual scores numerically, as presented in Table 1, we see that the sentence level correlation coefficients range between 0.61 and 0.70 for all systems. BLEU has the highest correlation, followed by invWER. At system level, all correlation coefficients except for the PER coefficient range between 0.95 and 0.98, here WER being the best, followed by BLEU. Neither in tendency nor in correlation values do we find a remarkable difference between the weighted and the unweighted 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.

Table 2: *German–English (VERBMOBIL):* System ranking according to automatic scores. Systems $S_1, \ldots S_{22}$ are numbered from best to worst according to manual evaluation. c_R denotes the rank correlation coefficient (Kendall, 1970).

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

5.2 Chinese-English

The TIDES 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, adding up to 878 sentences. Out of these sentences, we selected all sentences for which the maximum length of all candidate and reference sentences is 50 words or below. 657 test sentences hold this condition. Each sentence has been provided with four reference translations, generated by different human translators. The average reference sentence length is 23.5 words.

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 human evaluators. Both fluency and adequacy of a candidate sentence were judged separately, each from 1 (worst) to 5 (best) in steps of 1. For each sentence, we determined the mean fluency and the mean adequacy over its two or three judgments. We summed the average fluency and adequacy into a single manual score. Experiments showed that there was no significant difference in the correlation of each automatic evaluation measure with respect to fluency and adequacy respectively.

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

The Chinese–English task 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 one of its reference translation.[1]

5.2.1 Experiment A

In the first experiment, case information was omitted. Manual evaluation scores were adopted without changes. Table 3A shows the correlation coefficients

---

[1]Most interestingly, this translation was rated 3.5 out of 5 in fluency and 4.5 out of 5 in adequacy by human evaluators.

Table 3: *Chinese–English (TIDES):* Correlation coefficients between automatic and manual scores at the sentence and system level (weighted and unweighted scores at the system level): without (A) and with (B) case sensitive automatic scoring and normalization of manual scores.

	evaluation measure	sentence	system	
			weighted	unweighted
A	PER	0.24	0.09	0.09
	WER	0.23	0.02	0.01
	invWER	0.25	0.02	0.03
	BLEU	0.23	0.21	0.24
	avgBLEU	-	0.08	0.11
B	PER	0.23	0.59	0.58
	WER	0.23	0.28	0.29
	invWER	0.24	0.40	0.41
	BLEU	0.20	0.77	0.76
	avgBLEU	-	0.58	0.49

between manual and automatic evaluation. We see that a weak correlation is present at sentence level. At system level, no acceptable correlation between manual and automatic evaluation measures can be found.

5.2.2 Experiment B

In the second experiment, first we normalized all the fluency and adequacy judgments of each evaluator, such that the mean of these judgments is 0.0 and the variance 1.0 (for each evaluator, over all sentences, documents and systems). Then we averaged the judgments of each sentence. Furthermore, the automatic evaluation measures were configured to include case information (Doddington, 2003) .

Table 3B shows that at system level BLEU performs better than the other automatic evaluation methods; PER and invWER show acceptable correlation. Figure 3 supports this observation.

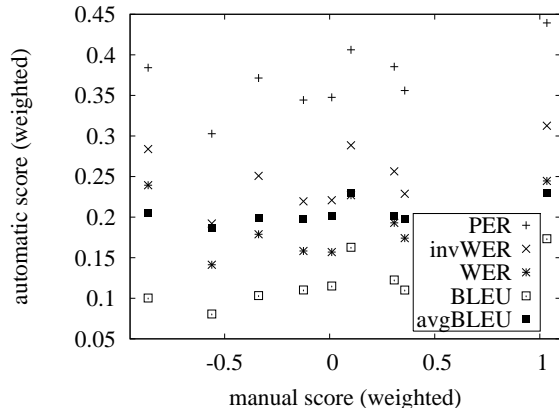However, for this low number of samples (i.e. 9),

Figure 3: *Chinese–English (TIDES), Experiment B:* System level comparison of automatic and manual scores.

Figure 4: *Chinese–English (TIDES), Experiment B:* Sentence level comparison of automatic and manual scores (averaged for each manual score inverval).

Table 4: *Chinese–English (TIDES), Experiment B:* System ranking according to automatic scores; case information included. Systems $S_1, \ldots S_9$ are numbered from best to worst according to normalized manual evaluation.

Measure	Ranking	c_R
PER	$S_1\ S_4\ S_3\ S_9\ S_7\ S_2\ S_5\ S_6\ S_8$	0.52
WER	$S_1\ S_9\ S_4\ S_3\ S_7\ S_2\ S_6\ S_5\ S_8$	0.32
invWER	$S_1\ S_4\ S_9\ S_3\ S_7\ S_2\ S_5\ S_6\ S_8$	0.42
BLEU	$S_1\ S_4\ S_3\ S_5\ S_6\ S_2\ S_7\ S_9\ S_8$	0.80
avgBLEU	$S_1\ S_4\ S_9\ S_3\ S_5\ S_7\ S_6\ S_2\ S_8$	0.33

the correlation can change significantly due to small sample effects. E.g. omitting the system with the lowest manual evaluation score increases the correlation coefficient to a value between 0.75 and 0.82 for *all* automatic evaluation measures.

When looking at Figure 4, it must be stressed that there is an implicit weighting by the number of sentences for each data point. Nevertheless it shows that PER, WER and invWER behave rather similarly to each other at sentence level, even though the difference between them is bigger than in Figure 1. In comparison with the other automatic evaluation measures, the BLEU measure shows a rather small dynamic range.

As we have noticed at the German–English task, the correlation coefficient is significantly higher than the sentence level correlation coefficient for all automatic evaluation measures, especially for BLEU.

In Table 4, the ranking of the systems according to the five automatic evaluation scores is listed. We see that the automatic evaluation measures produce similar rankings. The BLEU ranking has the highest correlation with manual ranking, followed by the PER ranking and the invWER ranking. Again, the degree of correlation is very much affected by removing the data point of a single MT system.

6 Discussion

Before drawing conclusions from the experiments, it is important to sum up the differences between the two translation tasks:

- The VERBMOBIL task has a limited domain and a rather small vocabulary size (e.g. 5000 words), whereas the Chinese news articles cover various different domains and have a large vocabulary (e.g. 50000 words).
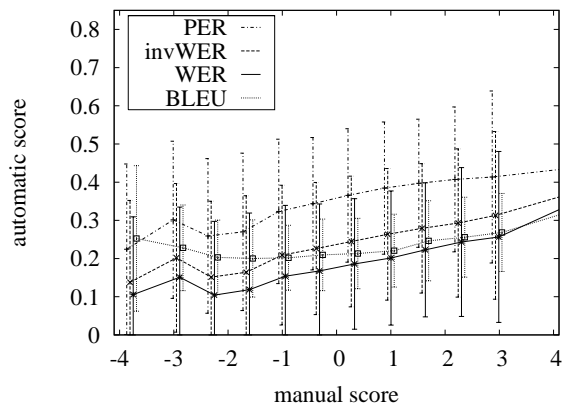
- The manual evaluation process for the VERB-MOBIL task had been designed especially to provide for reproducible results. Each evaluation was compared with all other evaluations of a sentence by the human evaluator (Nießen et al., 2000). The Chinese–English evaluation was achieved by averaging the evaluation of different human evaluators.

- While the translations for the VERBMOBIL task were generated by different variants of the RWTH MT system, the TIDES evaluation is based on MT systems of different research groups and companies.

Comparing the correlation coefficient between automatic and manual evaluation, we find remarkable differences between the two translation tasks presented above: On the German–English corpus, all automatic evaluation scores correlate strongly with human judgment. On the Chinese–English task, the correlation between automatic evaluation scores and human judgment is still significant, but notably lower.

The results of the Chinese–English experiments A and B indicate that, for a good correlation between automatic and manual scores, the utilization of case information in automatic evaluation can be essential. We observed that, on the Chinese-English task, reference translations are very inconsistent concerning capitalization, especially in the translation of headlines. A similar inconsistency holds for numerical and temporal named entities and the handling of untranslated words.

When comparing performance at sentence and system level, a potential problem with our experiments is that BLEU makes use of a geometric mean as opposed to arithmetic averaging. Therefore the NIST score might be more appropriate.

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 symbol blocks at constant cost.

We have defined the inversion edit distance as the parse cost of a sentence pair within a simple inversion grammar, and showed it to be a metric. 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 presented an application of the inversion edit distance in the evaluation of machine translation systems. Its correlation with human judgment has been compared with other automatic evaluation measures on two different translation tasks. The experiments show that the correlation of this automatic evaluation measure with human judgment is appropriate at system level. The experiments also show that, for the automatic evaluation measures studied, there is a significant difference in correlation with human judgment between sentence level evaluation and system level evaluation.

8 Acknowledgments

9 Bibliographical References

G. Doddington. 2003. NIST MT Evaluation Workshop. Personal communication. Gaithersburg, MD, July.

M. G. Kendall. 1970. *Rank Correlation Methods*. Charles Griffin & Co Ltd, London.

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

S. Nießen, F. J. Och, G. Leusch, and H. Ney. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proc. Second Int. Conf. on Language Resources and Evaluation*, pp. 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/`.

P. Norvig. 1991. Techniques for automatic memoization with applications to context-free parsing. *Computational Linguistics*, 17(1), pp. 93–98.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. 40th Annual Meeting of the Assoc. for Computational Linguistics*, pp. 311–318, Philadelphia, PA, July.

J. M. Vilar. 2000. Improve the learning of subsequential transducers by using alignments and dictionaries. In A. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, vol. 1891 of *LNAI*, pp. 298–311, Lisbon, Portugal, September. Springer.

W. Wahlster, editor, 2000. *Verbmobil: Foundations of speech-to-speech translations*, pp. 22–30. Springer.

M. S. Waterman, 1995. *Introduction to computational biology. Maps, sequences and genomes*, pp. 185–232. Chapman and Hall, 16th edition.

D. Wu. 1995. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proc. 33th Annual Meeting of the Assoc. for Computational Linguistics*, pp. 244–251.

D. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2), pp. 189–208.