CHARCUT: Human-Targeted Character-Based MT Evaluation with Loose Differences
Adrien Lardilleux, Yves Lepage

To cite this version:
Adrien Lardilleux, Yves Lepage. CHARCUT: Human-Targeted Character-Based MT Evaluation with Loose Differences. Proceedings of IWSLT 2017, Dec 2017, Tokyo, Japan. IWSLT 2017 International Workshop on Spoken Language Translation PROCEEDINGS. <hal-01726326>

HAL Id: hal-01726326
https://hal.archives-ouvertes.fr/hal-01726326
Submitted on 8 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
CHARCUT: Human-Targeted Character-Based MT Evaluation with Loose Differences

Adrien Lardilleux*, Yves Lepage**

*Fujitsu Technology Solutions, Luxembourg
*DGTEuropean Commission, Luxembourg
**IPS, Waseda University, Japan

adrien.lardilleux@ext.ec.europa.eu, yves.lepage@waseda.jp

Abstract

We present CHARCUT, a character-based machine translation evaluation metric derived from a human-targeted segment difference visualisation algorithm. It combines an iterative search for longest common substrings between the candidate and the reference translation with a simple length-based threshold, enabling loose differences that limit noisy character matches. Its main advantage is to produce scores that directly reflect human-readable string differences, making it a useful support tool for the manual analysis of MT output and its display to end users. Experiments on WMT16 metrics task data show that it is on par with the best "untrained" metrics in terms of correlation with human judgement, well above BLEU and TER baselines, on both system and segment tasks.

1. Introduction

A large number of metrics have been proposed in the past years for the task of objective evaluation of Machine Translation. To this day, trained or combined metrics (e.g. BEER [1], DPMFComb [2], UoW.ReVal [3], COBALT [4], among others), generally attain top results in terms of average correlation with human judgement, as was concluded in recent WMT conferences [5, 6].

On the other hand, endogenous metrics present the advantage of versatility, the most widely used remaining BLEU [7] and TER [8]. Such versatility is crucial in environments where MT systems are built and tuned to support numerous languages, some of which having little resources available.

Among those metrics, character-based ones have received more and more interest, starting from BLEU in characters [9, 10] to the recent CHRF [11] and CharacTER [12]. Operating at the character level is all the more important as MT systems working at a sub-word level are getting more widely used, e.g. with segmentation schemes like Byte Pair Encoding in Neural MT [13]. Since character-based metrics can implicitly account for sub-word linguistic phenomena, they have shown to correlate much better with human judgements than BLEU and TER—sometimes by tremen-
2. Defining noticeable similarities

From a translator’s point of view, a useful MT output is one that requires little time to comprehend and edit in order to turn it into a quality translation. When the MT contains too many mistakes, it is faster to rewrite a new translation from scratch than to attempt to correct it [15, 19].

A similar process takes place when humans compare two segments. No matter how many common substrings there are, they have no interest if they cannot be identified by the user due to their being lost in a flow of differences. As noted by Wang et al. [12], character-based comparisons typically suffer from noisy matches on languages using alphabets, because letters tend to be repeated frequently. Consider for instance the following segment pair taken from WMT16 evaluations (C denotes the candidate segment, R the reference):

\[
C: \text{It was also remarkable for personal reasons.} \\
R: \text{It was noteworthy because of personal reasons.}
\]

The strings in bold are likely to look like “atomic” replacements for most human eyes. Yet they have characters in common. Indeed, a longest common subsequence between the two strings in bold could be oerbe, ef (spaces have the same status as any other character), underlined below:

\[
C: \underline{\text{also notable for personal reasons.}} \\
R: \underline{\text{noteworthy because of personal reasons.}}
\]

Not only would highlighting those prove useless to the user, but it would also direct the user’s focus onto meaningless pieces of strings that would go naturally unnoticed without highlighting. It gets even worse when taking shifts of isolated characters into account, which is precisely why CharacT ER only considers shifts of words. Note that in this particular example, word-based differences would yield much more satisfying results. But those are generally too coarse in the general case, especially when words differ only by e.g. a single ending, or when dealing with morphologically rich languages.

We propose a simple approach to account for sub-word differences, while showing only meaningful character matches to the user. In order to keep comparisons intelligible, we reduce the number of highlighted substrings (be they matches or differences) within segments by allowing loose differences, i.e. differences that may still contain a few common characters. To this end, we rely on standard string difference operations, with the addition of a single constraint: only substrings longer than a given threshold are considered for matching. In our experiments (Sec. 4), we have found that the best value is generally around 3 characters for European languages, and manual investigations suggest that an optimum would be 1 or 2 characters for Chinese, which is in line with the findings of Li et al. [10]. This single constraint significantly reduces the amount of displayed information, helping the user focus more on meaningful differences.

To our knowledge, this approach was first used as a similarity measure by [20] in a clinical context for patient record matching. More recently, it was successfully applied to toponym matching [21]. It also presents similarities with the more complex MUMmer, a genome alignment system first introduced in [22], where what we call loose differences are the counterpart of what is known in bioinformatics as “highly polymorphic regions,” i.e. short regions of DNA that have undergone many mutations.

3. Method description

CHARCUT consists of three phases:

1. an iterative search for longest common substrings between the candidate and the reference translations;
2. the identification of string shifts;
3. a scoring phase based on the lengths of remaining differences.

3.1. Iterative segmentation algorithm

In the first phase, we identify a set of non-overlapping matches by applying an iterative search for the longest common subsequence (hereafter LCSubstr\(^2\)) between a candidate \(C_n\) and a reference \(R_n\), and cutting off this LCSubstr from both segments:

\[
C_{n+1} = C_n - \text{LCSubstr}(C_n, R_n) \\
R_{n+1} = R_n - \text{LCSubstr}(C_n, R_n)
\]

When several LCSubstr’s are possible (same length), the leftmost one in \(C_n\) is processed first, and is paired with the leftmost corresponding match in \(R_n\). A LCSubstr removed is replaced with a (zero-length) hard boundary that subsequent LCSubstr’s cannot cross. We iterate until the length of LCSubstr\((C_n, R_n)\), which monotonously decreases at each step, is below a certain threshold (typically around 3 characters).

Our first investigations have revealed that pure character-based matching, treating spaces as any other character, could lead to misinformed segmentations in presence of shifts of words with identical prefixes or suffixes (see Fig. 1 for an example). For this reason, we consider only a subset of all possible substrings of \(C_n\) and \(R_n\) when searching for the LCSubstr, by considering only those that match any of the three following regular expressions:

- \(\backslash w* \backslash w+ \backslash w*\) (intra-word substring, does not span multiple words);
- \(\backslash w* \backslash b+ \backslash b \backslash w*\) (inter-word substring, stops at word boundaries or non-word characters);
- \(\backslash w+\) (run of non-word characters).

\(^2\)Contrary to the Longest Common Subsequence (LCS), the LCSubstr is exclusively made up of adjacent characters.
Figure 1: A common pitfall where the raw character based longest-first approach can lead to a counter-intuitive segmentation. The first LCSubstr is underlined. Because the two swapped German words Europäischen and Gemeinsamen share the same ending, this ending has been integrated into the LCSubstr, preventing the more natural full word matches. We circumvent this issue by making our algorithm aware of word separators.

<table>
<thead>
<tr>
<th>n</th>
<th>C&lt;sub&gt;n&lt;/sub&gt;</th>
<th>R&lt;sub&gt;n&lt;/sub&gt;</th>
<th>LCSubstr(C&lt;sub&gt;n&lt;/sub&gt;, R&lt;sub&gt;n&lt;/sub&gt;)</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Before the game, it had arrived at the stadium to riots.</td>
<td>Before the match there was a riot in the stadium.</td>
<td>the stadium</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>Before the game, it had arrived at</td>
<td>to riots.</td>
<td>Before the match there was a riot in</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[game, it had arrived at</td>
<td>to riots.</td>
<td>[match there was a riot in</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[game, it had arrived at</td>
<td>to</td>
<td>s.</td>
<td>[match there was a</td>
</tr>
</tbody>
</table>

Figure 2: Example of iterative search for longest common substrings (LCSubstr). At each step, the LCSubstr (underlined) is cut off and replaced with a zero-length boundary (noted with a pipe character “|”) that subsequent LCSubstr’s may not cross. The process stops when the length of the LCSubstr is below a given threshold—here, 3 characters, preventing smaller common substrings, starting with at: at step 3, to be considered as matches. The longest common suffix (single full stop) is eventually added to the list of LCSubstr’s, while the longest common prefix was already extracted as a regular LCSubstr.

C<sub>0</sub>: Before the game, it had arrived at the stadium to riots.
R<sub>0</sub>: Before the match there was a riot in the stadium.

Figure 3: Segmentation resulting from the iterative search of Fig. 2. Matches (= LCSubstr’s) are underlined, and the remaining substrings are loose differences. Here, those differences still have around 68% of characters in common (16), while no meaningful lexical correspondences are visible: the length-based threshold has successfully prevented a large amount of noise that would otherwise make the output unreadable.

This leads to a mix of word- and character-based LCSubstr’s which we felt more natural than pure character-based ones in our experiments. In the case of scripts without word separators such as Chinese, most LCSubstr’s match the first expression.

Eventually, we also add the longest common prefix and the longest common suffix between C<sub>0</sub> and R<sub>0</sub> to the list of LCSubstr’s, independently of their length, providing they match the second or third regular expression and were not already extracted as a regular LCSubstr. This addition had almost no impact in terms of correlation with human judgement in our experiments, but it improves highlighting by fixing frequent cases of true negatives, such as final punctuations or segments shorter than the minimum match size, that most users would expect to be considered as matches.

Then:

- the set of LCSubstr’s extracted up to this point (including longest common prefix and suffix) are matches;
- the remaining strings, i.e. the last computed C<sub>n</sub> and R<sub>n</sub>, are loose differences.

Figures 2 and 3 give an example. Contrary to edit distances, our approach does not yield a minimal sequence of operations that would turn C<sub>0</sub> into R<sub>0</sub>; instead, it seeks to lower the number of matches and differences, hence the user reading effort.

3.2. Identifying string shifts

CHARCUT naturally handles string shifts, as the position change between  the stadium and  riot in Fig. 3 illustrates. For the purpose of highlighting and scoring, we mark the shortest one ( riot) as a shift, and the other one as a regular match.

More generally, when faced with multiple alternative shifts, we identify the longest common subsequence, in total number of characters, between the sequence of LCSubstr’s from C<sub>0</sub> (hereafter noted C<sub>match</sub>) and that from R<sub>0</sub> (hereafter R<sub>match</sub>), and any LCSubstr left out is marked as a shift. The two input sequences have exactly the same tokens, but in a different order (here 4 tokens = 4 LCSubstr’s, delimited by a pipe “|”):

C<sub>match</sub> = Before the, the stadium|riot|.
R<sub>match</sub> = Before the|riot|the stadium|.

The longest common subsequence has three tokens: Before the, the stadium, for a total of 12 + 11 + 1 = 24 characters. Those tokens will be referred to as regular matches in the following. Tokens left out (here, the single token riot) are marked as shifts, and will be scored and highlighted accordingly later on. Note that our definition
of the longest common subsequence deviates from the general LCS definition, since we do not base its computation on the number of tokens, but on the sum of their lengths.

3.3. Scoring scheme

The result of the previous phases is a segmentation of the input segments in three types of substrings: regular matches, shifts, and loose differences. Loose differences include deletions (from the candidate segment) and insertions (into the reference segment). We derive a score from those substrings by assigning a cost to each character of the candidate and reference segments:

- characters from regular matches have no cost;
- shifted, deleted, and inserted characters have a cost of 1 (shifted characters are counted only once although they appear in both segments).

We do not consider the combination deletion + insertion = replacement as a single operation because, by definition, there is no correspondence inside loose differences. While this combination appears natural when dealing with word units, it makes much less sense on characters, as identification of replacement pairs within differences would be arbitrary, especially if their lengths highly differ between the candidate and the reference.

While CharacT

ER assigns a shift cost equal to the average word length of the shifted phrase, we use the total number of shifted characters instead, thus keeping the computation rather straightforward. There is little risk of over-evaluating the cost of shifts because, by definition (Sec. 3.2), their length is minimal.

In our setting, the cost of post-editation is thus the total number of edited characters. An intuitive normalization scheme would be to divide this number by the total length of the candidate and reference segments in order to produce a score between 0 and 1:

\[ \text{score}_{\text{orig}} = \frac{\#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{|C_0| + |R_0|} \]  

(2)

However, following Wang et al. [12], we tried using only the length of the candidate, and we could confirm that it generally leads to higher correlation with human judgements (see experiments in next section). We thus consider also the following variant, where we divide by twice the candidate length instead, and limit the final score to 1 in case the number of edited characters exceeds the denominator:

\[ \text{score}_{C} = \min \left( \frac{1, \#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{2 \times |C_0|} \right) \]  

(3)

The lower the scores, the better. A score of zero means that the candidate and reference segments are identical. In the example of Fig. 3, we obtain \( \text{score}_{\text{orig}} = \frac{27 + 21 + 5}{156} \approx 0.50 \) and \( \text{score}_{C} = \frac{27 + 21 + 5}{2 \times 56} \approx 0.47 \).

4. Experiments

4.1. Task description

We evaluate CharacT

ER on the WMT16 system- and segment-level news metrics tasks [6], using the official evaluation scripts. We report results obtained with the “direct assessment” golden truth (hereafter DA), as it was concluded in WMT16 that it was more reliable than relative ranking, and it was also chosen as the official human evaluation at WMT17. Under this evaluation scheme, humans evaluate the adequacy of translations on an absolute scale in isolation from other translations, and the correlation with automatic scores is measured by means of absolute Pearson correlation coefficient. The data consist of news texts from Czech, Finnish, German, Romanian, and Turkish, into English, plus the Russian-English language pair in both directions.

In addition, we report results of the segment-level tasks under the “HUMEseg” evaluation scheme [23], which was also an official human evaluation of WMT17. Similarly to DA, scores are compared using the Pearson correlation coefficient, but with human judgements of semantic nodes aggregated over each sentence rather than single absolute scores. The data used for those tasks are texts from the medical domain from English into Czech, German, Romanian, and Polish.

4.2. Optimizing for correlation with human judgement

Figure 4 reports the average Pearson correlations between human judgements and various set-ups of CharacT

ER. On average, the correlations obtained with the \( \text{score}_{C} \) scheme (eq. 3) is greater than that obtained with \( \text{score}_{\text{orig}} \) (eq. 2) by 0.01, which confirms the findings of Wang et al. [12].

Although in practice varying the minimum match size leads to visually very different outputs, especially with low values, they seem to have a limited impact on correlations with human judgements: the average range of the absolute Pearson coefficient (difference between maximum and minimum) is 0.01. The system- and segment-level DA graphs show curves that tend to increase slightly then decrease, with a maximum correlation when the minimum match size equals 2 or 3 characters. This is consistent with the sense we get from the corresponding highlighting, which “looks right” to the eye—too small values leading to noisy matches, and too high values to silence.

On the contrary, the monotonously decreasing curves of the segment-level HUME graph suggest that smaller minimum match sizes would be better, which is in contradiction with the other results. We will nevertheless restrict our following experiments to a minimum match size of 3 characters, as it constitutes a good compromise between the above three

\[ \text{The DA evaluations of WMT17 cover more diverse target languages, in particular Chinese, which constitutes a good test for character based approaches, but the official evaluation scripts were not publicly released at the time this paper was written. For consistency, we therefore chose to stick to the WMT16 evaluations.} \]
evaluations and our manual investigations, in which we observed that a 2 character threshold still produced too many noisy matches.

4.3. Comparison with other metrics

Table 1 compares CHARCUT with metrics that took part in the WMT16 evaluations. For conciseness, we only report average correlations over all language pairs. The rankings differ from the official WMT16 results since we chose to focus on the Direct Assessment and HUME evaluations, while the official evaluations were based on Relative Ranking. The reported results make therefore no pretense to (re-)define the “best metrics;” rather, they are only meant to show that the scores produced by CHARCUT, which are first and foremost intended to be presented to users along with segment highlighting, are globally as good as other recent metrics, well above well-known baselines.

In these experiments, CHARCUT uses the score_C normalization scheme and a minimum match size of 3-characters. We also report additional correlations obtained with the Levenshtein distance, normalized with the sum of the source and target segment lengths, to serve as a character-based baseline; as well as with TER and CharacTER on segment-level tasks.

Globally, CHARCUT’s results are very close to those of MPEDA [24], which relies yet on additional training corpora. Compared with other endogenous metrics (chrF, wordF, CharacTER, variants of BLEU and TER, Levenshtein distance), CHARCUT produces top average correlations on the system- and segment-level DA evaluations, and is only superseded by chrF on the HUME evaluation. A fortiori, its correlations are much higher than those of the BLEU and TER baselines: from +9% relative Pearson correlation (MTEVAL.BLEU, system-level DA) up to +23% (TER, segment-level HUME).

Unexpectedly, the simple normalized character-based Levenshtein distance performs quite well, outperforming even metrics like BEER and CharacTER on the DA evaluations. CHARCUT nevertheless represents a consistent improvement over it, by +0.03 absolute Pearson correlation on average.

4.4. Processing time

We used a random sample of 10,000 segment pairs from WMT16 to measure the speed of CHARCUT. The average reference length in this sample was 113 characters. On a 2.8 GHz processor, our Python implementation could process 260 segment pairs per second, using a minimum match size of 3 characters, which is faster than required in most situations. For comparison, CharacTER and chrF, also Python implementations, could process respectively 54 and 600 segment pairs per second with default settings on the same machine.

5. Conclusion

We have presented CHARCUT, a character-based machine translation evaluation metric. It relies on loose differences, residuals from an iterative search for longest common substrings. Initially designed for displaying differences between reference and candidate segments to end users, it also produces scores that should look consistent to most, since they directly reflect those differences. In this view, good correlation with human judgement is only a consequence of a good
Table 1: Comparison of CHARCut’s performances with metrics that took part in the system-level DA, segment-level DA, and segment-level HUME tasks of WMT16. We report the average Pearson correlation coefficients over all language pairs. Averages within brackets refer to metrics that did not participate in the English-to-Russian evaluation, so they are based on one less figure. Asterisks indicate our own runs; all other averages are based on figures from [6]. CHARCut is globally on par with the best metrics in those evaluations.

<table>
<thead>
<tr>
<th>system-level DA</th>
<th>segment-level DA</th>
<th>segment-level HUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>UoW.REVAL</td>
<td>(0.972 ± 0.013)</td>
<td>DPMFCOMP</td>
</tr>
<tr>
<td>MPEDA</td>
<td>0.945 ± 0.044</td>
<td>METRICS-F</td>
</tr>
<tr>
<td>*CHARCut</td>
<td>0.942 ± 0.037</td>
<td>COBALT-F.</td>
</tr>
<tr>
<td>CHRF2</td>
<td>0.934 ± 0.038</td>
<td>MPEDA</td>
</tr>
<tr>
<td>CHRF3</td>
<td>0.934 ± 0.035</td>
<td>*CHARCut</td>
</tr>
<tr>
<td>*Lev. distance</td>
<td>0.930 ± 0.049</td>
<td>UPP-COBALT</td>
</tr>
<tr>
<td>BEER</td>
<td>0.928 ± 0.054</td>
<td>CHRF3</td>
</tr>
<tr>
<td>CHRF1</td>
<td>0.927 ± 0.051</td>
<td>CHRF2</td>
</tr>
<tr>
<td>CHARActER</td>
<td>0.922 ± 0.055</td>
<td>*Lev. distance</td>
</tr>
<tr>
<td>MTEVALNIST</td>
<td>0.886 ± 0.068</td>
<td>BEER</td>
</tr>
<tr>
<td>MTEVALBLEU</td>
<td>0.867 ± 0.060</td>
<td>chrF1</td>
</tr>
<tr>
<td>MOSESCDER</td>
<td>0.861 ± 0.061</td>
<td>chrF2</td>
</tr>
<tr>
<td>MOSESTER</td>
<td>0.851 ± 0.061</td>
<td>WORDF3</td>
</tr>
<tr>
<td>MOSEPESR</td>
<td>0.842 ± 0.096</td>
<td>WORDF2</td>
</tr>
<tr>
<td>WORDF3</td>
<td>0.836 ± 0.069</td>
<td>WORDF1</td>
</tr>
<tr>
<td>WORDF2</td>
<td>0.836 ± 0.069</td>
<td>WORDF1</td>
</tr>
<tr>
<td>WORDF1</td>
<td>0.831 ± 0.071</td>
<td>SENTBLEU</td>
</tr>
<tr>
<td>MOSESWER</td>
<td>0.812 ± 0.099</td>
<td>*TER</td>
</tr>
<tr>
<td>MOSESBLEU</td>
<td>0.810 ± 0.082</td>
<td>DTED</td>
</tr>
</tbody>
</table>

visual representation. Experiments on WMT16 metrics tasks have thus shown that those scores are well correlated with human judgements, globally on par with other recent metrics like CHRF and MPEDA, ahead of BLEU and TER baselines by up to 23% relative Pearson correlation in our experiments. It is also language independent and requires no additional resource or training. Possible improvements include better handling of shifts, as CHARCut is currently unaware of shift distance; or again automatically correlate the minimum match size with the number of highlighted substrings in order to keep outputs readable even with very different input segments.

6. Availability

CHARCut is open source and available at https://github.com/alardill/CharCut. It consists of a single Python script that computes scores and highlights differences (HTML outputs). Figure 5 shows a sample output.

7. References


Figure 5: Sample HTML output obtained on the first three English-Chinese segments of WMT17, using a 2-character minimum match size. The interface is kept slick on purpose and uses only classical colours: red for deletions, blue for insertions, plus bold for shifts (here, only one in second segment). The scores reflect directly the number of highlighted characters. The background of matching substrings turns yellow when pointed with the mouse.


