CHARCUT: Human-Targeted Character-Based MT Evaluation with Loose Differences

Adrien Lardilleux, Yves Lepage

CharCut: Human-Targeted Character-Based MT Evaluation with Loose Differences

Adrien Lardilleux, Yves Lepage***

*Fujitsu Technology Solutions, Luxembourg
*DGT, European 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], CobaltF [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 tremendous margins [5, 6]. Nowadays, such metrics seem to be safe for use as drop-in replacements for BLEU.

Another key aspect of MT evaluation is the *display* of MT output and its comparison with human references. Highlighting differences between a candidate and a reference translation is a standard feature of many translation interfaces (e.g. MateCat's edit log [14] or SDL Studio's SDLXLIFF Compare[1]). Basically, any string comparison tool operating at a sub-segment level could be used to that end. The potential associated scores, however, typically simple word or character match percentages, may not reflect all aspects expected from a MT metric. One could just replace it with another metric, but inconsistencies between score and visible differences would ensue [15], which might be confusing for non-specialists. Advanced analysis environments proposing multiple measures along with word-based highlighting, such as Asiya [16] or MT-ComparEval [17], among others, are thus aimed at MT researchers rather than end users.

Some metrics allow to naturally derive user-friendly visual correspondences between candidate and reference translations. This is typically the case of word alignment based metrics (e.g. TER or METEOR [18]), as opposed to those based e.g. on overlapping n-grams (such as BLEU or chrF), or character-based approaches, which are often subject to noise [12].

We propose an approach that benefits from fine character-based differences while getting rid of their main drawback, namely noise. Initially designed as a mean for displaying differences to end users, it is also a full-fledged MT evaluation metric, as a score can be directly inferred from those human-targeted differences. In this view, a good metric is only the *consequence* of a good visual representation.

This paper is organized as follows: Section 2 describes how user perception of similarities led to the design of our metric; Section 3 builds up on those observations to describe the method in details; Section 4 evaluates it in terms of correlation with human judgements; and Section 5 concludes this work.

[1] http://appstore.sdl.com/app/sdlxliff-compare/89/

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: It was **also remarkable for** personal reasons.
R: 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␣f` (spaces have the same status as any other character), underlined below:

C: also remarkable for
R: noteworthy because of

Not only would highlighting those prove useless to the user, but it would also direct the eye 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 CharacTER 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 subword 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 substring (hereafter LCSubstr[2]) between a candidate C_0 and a reference R_0, and cutting off this LCSubstr from both segments:

$$\begin{aligned} C_{n+1} &= C_n - \text{LCSubstr}(C_n, R_n) \\ R_{n+1} &= R_n - \text{LCSubstr}(C_n, R_n) \end{aligned} \tag{1}$$

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_0 and R_0 when searching for the LCSubstr, by considering only those that match any of the three following regular expressions:

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

[2] Contrary to the Longest Common Subsequence (LCS), the LCSubstr is exclusively made up of adjacent characters.

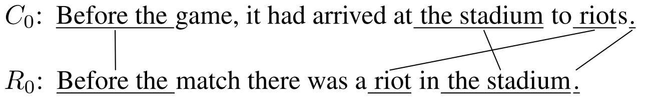
C: [. . .] der Europäischen Gemeinsamen Strategie zur Unterstützung Palästinas [. . .]

R: [. . .] der Gemeinsamen Europäischen Strategie zur Unterstützung Palästinas [. . .]

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.*

n	C_n	R_n	LCSubstr(C_n, R_n)	length
0	Before the game, it had arrived at the stadium to riots.	Before the match there was a riot in the stadium.	`␣the␣stadium`	12
1	Before the game, it had arrived at| to riots.	Before the match there was a riot in|.	`Before␣the␣`	11
2	|game, it had arrived at| to riots.	|match there was a riot in|.	`␣riot`	5
3	|game, it had arrived at| to|s.	|match there was a| in|.	`at`	2

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_0: Before the game, it had arrived at the stadium to riots.

R_0: 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_0 and R_0 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_n and R_n, 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_0 into R_0; 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_0 (hereafter noted C_{match}) and that from R_0 (hereafter R_{match}), 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_{match} = `Before␣the␣|␣the␣stadium|␣riot|.`
R_{match} = `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 CharacTER 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-edition 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:

$$\text{score}_{\text{orig}} = \frac{\#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{|C_0| + |R_0|} \quad (2)$$

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:

$$\text{score}_C = \min\left(1, \frac{\#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{2 \times |C_0|}\right) \quad (3)$$

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 $\text{score}_{\text{orig}} = \frac{27+21+5}{56+49} \simeq 0.50$ and $\text{score}_C = \frac{27+21+5}{2\times56} \simeq 0.47$.

4. Experiments

4.1. Task description

We evaluate CharCut 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.[3]

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 tracks are texts from the medical domain from English into Czech, German, Romanian, and Polish.

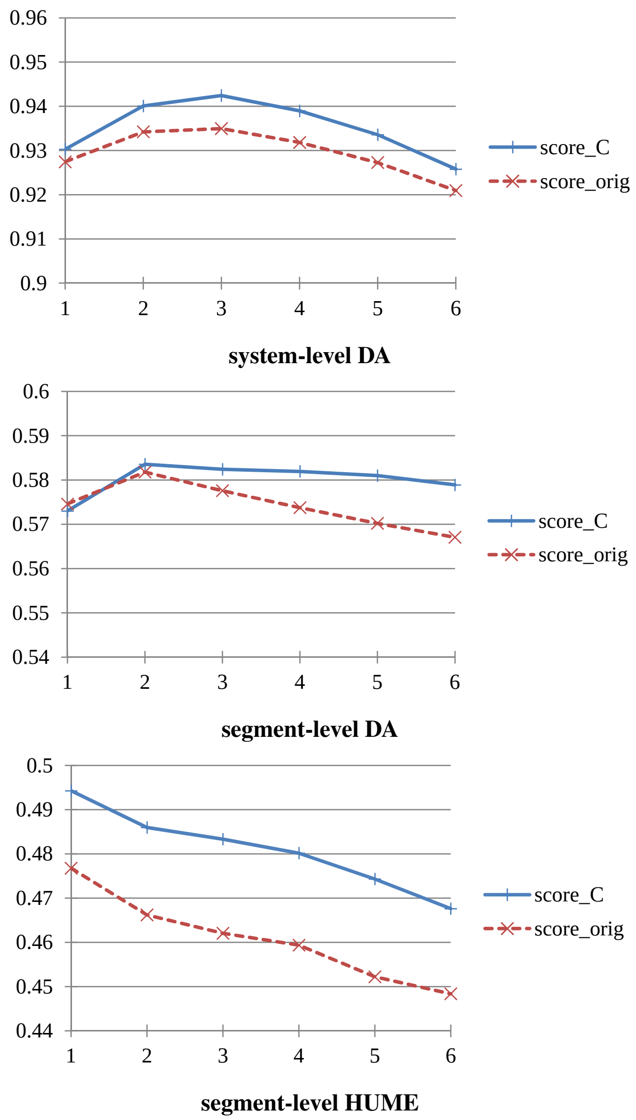
4.2. Optimizing for correlation with human judgement

Figure 4 reports the average Pearson correlations between human judgements and various set-ups of CharCut. On average, the correlations obtained with the 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

[3] 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.



system-level DA

segment-level DA

segment-level HUME

Figure 4: *Average correlations between* CHARCUT *scores and human judgements on WMT16's metrics tasks. In ordinates, the absolute Pearson correlation coefficient. In abscissae, the minimum match size in characters (length-based threshold). The reported numbers are averages over all language pairs. Normalizing with the candidate segment length only (score$_C$) consistently outperforms using both the candidate and reference lengths (score$_{orig}$).*

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 pretence 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 (MTEVALBLEU, 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.*

system-level DA

Metric	Avg. corr. ± stddev.
UoW.ReVaL	(0.972 ± 0.013)
MPEDA	0.945 ± 0.044
*CharCut	0.942 ± 0.037
chrF2	0.934 ± 0.038
chrF3	0.934 ± 0.035
*Lev. distance	0.930 ± 0.049
BEER	0.928 ± 0.054
chrF1	0.927 ± 0.051
CharacTer	0.922 ± 0.055
mtevalNIST	0.886 ± 0.068
mtevalBLEU	0.867 ± 0.060
mosesCDER	0.861 ± 0.061
mosesTER	0.851 ± 0.061
mosesPER	0.842 ± 0.096
wordF3	0.836 ± 0.069
wordF2	0.836 ± 0.069
wordF1	0.831 ± 0.071
mosesWER	0.812 ± 0.099
mosesBLEU	0.810 ± 0.082

segment-level DA

Metric	Avg. corr. ± stddev.
DPMFcomb	(0.633 ± 0.048)
metrics-f	(0.631 ± 0.049)
cobalt-f.	(0.617 ± 0.040)
MPEDA	0.584 ± 0.053
*CharCut	0.582 ± 0.076
UPF-cobalt	(0.582 ± 0.060)
chrF3	0.560 ± 0.082
chrF2	0.559 ± 0.081
*Lev. distance	0.556 ± 0.065
BEER	0.556 ± 0.082
chrF1	0.548 ± 0.079
*CharacTer	0.537 ± 0.074
UoW.ReVaL	0.530 ± 0.035
wordF3	0.524 ± 0.055
wordF2	0.522 ± 0.055
wordF1	0.514 ± 0.055
sentBLEU	0.510 ± 0.039
*TER	0.485 ± 0.052
DTED	0.330 ± 0.058

segment-level HUME

Metric	Avg. corr. ± stddev.
chrF3	0.519 ± 0.096
chrF2	0.517 ± 0.092
BEER	0.513 ± 0.079
chrF1	0.503 ± 0.079
MPEDA	0.492 ± 0.073
*CharCut	0.483 ± 0.121
wordF3	0.452 ± 0.092
wordF2	0.450 ± 0.091
wordF1	0.439 ± 0.088
*CharacTer	0.438 ± 0.126
*Lev. distance	0.437 ± 0.109
sentBLEU	0.401 ± 0.101
*TER	0.394 ± 0.125

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.

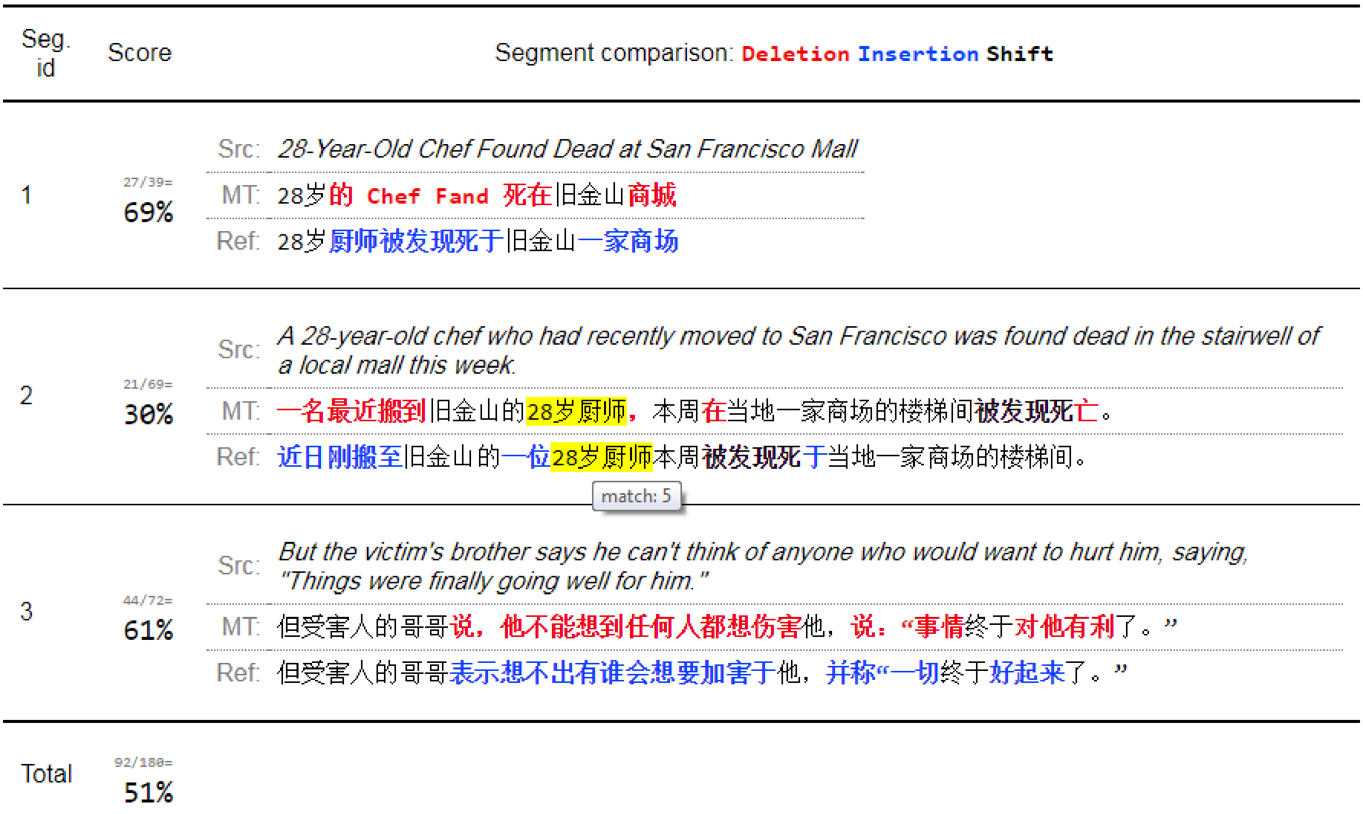
7. References

[1] M. Stanojević and K. Sima'an, "BEER 1.1: ILLC UvA submission to metrics and tuning task," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 396–401. [Online]. Available: http://aclweb.org/anthology/W15-3050

[2] H. Yu, Q. Ma, X. Wu, and Q. Liu, "CASICT-DCU Participation in WMT2015 Metrics Task," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 417–421. [Online]. Available: http://aclweb.org/anthology/W15-3053

[3] R. Gupta, C. Orasan, and J. van Genabith, "Machine Translation Evaluation using Recurrent Neural Networks," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 380–384. [Online]. Available: http://aclweb.org/anthology/W15-3047

[4] M. Fomicheva, N. Bel, L. Specia, I. da Cunha, and A. Malinovskiy, "CobaltF: A Fluent Metric for MT Evaluation," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 483–490. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2339



Seg. id	Score	Segment comparison: Deletion Insertion Shift
1	27/39= 69%	Src: *28-Year-Old Chef Found Dead at San Francisco Mall* MT: 28岁的 Chef Fand 死在旧金山商城 Ref: 28岁厨师被发现死于旧金山一家商场
2	21/69= 30%	Src: *A 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week.* MT: 一名最近搬到旧金山的28岁厨师，本周在当地一家商场的楼梯间**被发现死**亡。 Ref: 近日刚搬至旧金山的一位28岁厨师本周**被发现死**于当地一家商场的楼梯间。 match: 5
3	44/72= 61%	Src: *But the victim's brother says he can't think of anyone who would want to hurt him, saying, "Things were finally going well for him."* MT: 但受害人的哥哥说，他不能想到任何人都想伤害他，说：“事情终于对他有利了。” Ref: 但受害人的哥哥表示想不出有谁会想要加害于他，并称“一切终于好起来了。”
Total	92/180= 51%	

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.*

[5] M. Stanojević, A. Kamran, P. Koehn, and O. Bojar, “Results of the WMT15 Metrics Shared Task,” in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 256–273. [Online]. Available: http://aclweb.org/anthology/W15-3031

[6] O. Bojar, Y. Graham, A. Kamran, and M. Stanojević, “Results of the WMT16 Metrics Shared Task,” in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 199–231. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2302

[7] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. [Online]. Available: http://www.aclweb.org/anthology/P02-1040

[8] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, “A Study of Translation Edit Rate with Targeted Human Annotation,” in *Proceedings of Association for Machine Translation in the Americas*. Cambridge, Massachusetts, USA: Association for Computational Linguistics, August 2006, pp. 223–231. [Online]. Available: http://www.mt-archive.info/AMTA-2006-Snover.pdf

[9] Étienne Denoual and Y. Lepage, “BLEU in Characters: Towards Automatic MT Evaluation in Languages without Word Delimiters,” in *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)*, Jeju Island, Republic of Korea, October 2005, pp. 79–84. [Online]. Available: http://www.aclweb.org/anthology/I/I05/I05-2014.pdf

[10] M. Li, C. Zong, and H. T. Ng, “Automatic Evaluation of Chinese Translation Output: Word-Level or Character-Level?” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 159–164. [Online]. Available: http://www.aclweb.org/anthology/P11-2028

[11] M. Popović, "chrF: character n-gram F-score for automatic MT evaluation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 392–395. [Online]. Available: http://aclweb.org/anthology/W15-3049

[12] W. Wang, J.-T. Peter, H. Rosendahl, and H. Ney, "CharacTer: Translation Edit Rate on Character Level," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 505–510. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2342

[13] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 1715–1725. [Online]. Available: http://www.aclweb.org/anthology/P16-1162

[14] M. Federico, N. Bertoldi, M. Cettolo, M. Negri, M. Turchi, M. Trombetti, A. Cattelan, A. Farina, D. Lupinetti, A. Martines, A. Massidda, H. Schwenk, L. Barrault, F. Blain, P. Koehn, C. Buck, and U. Germann, "THE MATECAT TOOL," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, August 2014, pp. 129–132. [Online]. Available: http://www.aclweb.org/anthology/C14-2028

[15] S. O'Brien, "Towards predicting post-editing productivity," *Machine Translation*, vol. 25, no. 3, p. 197, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10590-011-9096-7

[16] J. Giménez and L. Màrquez, "Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation," *The Prague Bulletin of Mathematical Linguistics*, no. 94, pp. 77–86, 2010. [Online]. Available: http://ufal.mff.cuni.cz/pbml/94/art-gimenez-marques-evaluation.pdf

[17] O. Klejch, E. Avramidis, A. Burchardt, and M. Popel, "MT-ComparEval: Graphical evaluation interface for Machine Translation development," *The Prague Bulletin of Mathematical Linguistics*, no. 104, pp. 63–74, 2015. [Online]. Available: http://ufal.mff.cuni.cz/pbml/104/art-klejch-et-al.pdf

[18] A. Lavie and A. Agarwal, "METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 228–231. [Online]. Available: http://www.aclweb.org/anthology/W/W07/W07-0734

[19] M. Turchi, M. Negri, and M. Federico, "MT Quality Estimation for Computer-assisted Translation: Does it Really Help?" in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 530–535. [Online]. Available: http://www.aclweb.org/anthology/P15-2087

[20] C. Friedman and R. Sideli, "Tolerating spelling errors during patient validation," *Computers and Biomedical Research*, vol. 25, no. 5, pp. 486–509, 1992.

[21] G. Recchia and M. Louwerse, "A Comparison of String Similarity Measures for Toponym Matching," in *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place*, Orlando, USA, November 2013, pp. 54–61. [Online]. Available: http://doi.acm.org/10.1145/2534848.2534850

[22] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg, "Alignment of whole genomes," *Nucleic Acids Research*, vol. 27, no. 11, pp. 2369–2376, 1999. [Online]. Available: http://dx.doi.org/10.1093/nar/27.11.2369

[23] A. Birch, O. Abend, O. Bojar, and B. Haddow, "HUME: Human UCCA-Based Evaluation of Machine Translation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 1264–1274. [Online]. Available: https://aclweb.org/anthology/D16-1134

[24] L. Zhang, Z. Weng, W. Xiao, J. Wan, Z. Chen, Y. Tan, M. Li, and M. Wang, "Extract Domain-specific Paraphrase from Monolingual Corpus for Automatic Evaluation of Machine Translation," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 511–517. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2343