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

Adrien LARDILLEUX* and Yves LEPAGE**

*Fujitsu Technology Solutions / DGT, European Commission, Luxembourg
**IPS, Waseda University, Japan
adrien.lardilleux@ext.ec.europa.eu, yves.lepage@waseda.jp
Background: metrics for MT

Trade-off between ease of use and correlation with human judgment
Background: metrics for MT

Trade-off between ease of use and correlation with human judgment

- Light methods (e.g., BLEU, WER)
  - Are very easy to use
  - Are better fitted to languages with less resource
Background: metrics for MT

Trade-off between ease of use and correlation with human judgment

- **Light methods**
  (e.g., BLEU, WER)
  - Are very easy to use
  - Are better fitted to languages with less resource

- **Trained or knowledge-based metrics**
  (e.g., BEER, DPMFcomb, UoW.ReVaL)
  - Better correlate with human judgment
  - But need training or resources (e.g., paraphrase tables)
Background: metrics for MT

Trade-off between
use of characters and
use of words
Background: metrics for MT

Trade-off between
use of characters and
use of words

- Word-based methods
  (e.g., BLEU, WER)
  - Are well-fitted for languages like English or segmented Chinese
Introduction

Background

Background: metrics for MT

Trade-off between use of characters and use of words

- Word-based methods (e.g., BLEU, WER)
  - Are well-fitted for languages like English or segmented Chinese

- Character-based methods (e.g., chrF, CHARACTER)
  - Are usually subject to noise for languages using the Latin script
  - But are better fitted for morphologically rich languages
  - Better correlate with human judgments
Background: metrics for MT

Trade-off between
   ease of visualisation and
the scoring mechanism
Background: metrics for MT

Trade-off between ease of visualisation and the scoring mechanism

- Word-based methods (e.g., TER, METEOR)
  - Allow to naturally derive user-friendly visual correspondences between candidate and reference translations
Background: metrics for MT

Trade-off between ease of visualisation and the scoring mechanism

- Word-based methods (e.g., TER, METEOR)
  - Allow to naturally derive user-friendly visual correspondences between candidate and reference translations
- Overlapping N-gram-based approaches (e.g., BLEU or CHRF)
  - Are more difficult to visualise
Proposal

**CharCut**, a light character-based machine translation evaluation metric derived from a human-targeted segment difference visualisation algorithm.
Proposal

**CHARCut**, a light character-based machine translation evaluation metric derived from a human-targeted segment difference visualisation algorithm.

- Light automatic metric for MT output: no training, no use of extra knowledge
Proposal

CharCut, a light character-based machine translation evaluation metric derived from a human-targeted segment difference visualisation algorithm.

- Light automatic metric for MT output: no training, no use of extra knowledge
- High correlation with human judgment: on par with trained or knowledge-based metrics ≃ best “untrained” metrics and ≫ BLEU and TER
Proposal

**CharCut**, a light character-based machine translation evaluation metric derived from a human-targeted segment difference visualisation algorithm.

- **Light automatic metric** for MT output: no training, no use of extra knowledge
- **High correlation** with human judgment: on par with trained or knowledge-based metrics \( \approx \) best “untrained” metrics and \( \gg \) BLEU and TER
- **Meaningful visualisation** of MT output vs. human reference: scores directly reflect human-readable string differences
Method

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  ⇒ loose differences ⇒ less noisy character matches.
Method

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  ⇒ loose differences ⇒ less noisy character matches.

C: It was also remarkable for personal reasons.
R: It was noteworthy because of personal reasons.
Method

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  \(\Rightarrow\) loose differences \(\Rightarrow\) less noisy character matches.

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

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  ⇒ loose differences ⇒ less noisy character matches.

C: It was also remarkable for personal reasons.
R: It was noteworthy because of personal reasons.
Method and problem

Combination of

- iterative search for **longest common substrings** between candidate and reference translation
- simple length-based threshold
  ⇒ loose differences ⇒ less noisy character matches.

*C:* It was also remarkable for personal reasons.
*R:* It was noteworthy because of personal reasons.
Method and problem

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  \[\Rightarrow\] loose differences \[\Rightarrow\] less noisy character matches.

C: It was also remarkable for personal reasons.
R: It was noteworthy because of personal reasons.
Method and problem

Combination of

- iterative search for longest common substrings between candidate and reference translation
- simple length-based threshold
  \[\Rightarrow\text{loose differences}\Rightarrow\text{less noisy character matches.}\]

C: It was also remarkable for personal reasons.
R: It was noteworthy because of personal reasons.
<table>
<thead>
<tr>
<th>Seg. id</th>
<th>Score</th>
<th>Segment comparison: Deletion Insertion Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33/109=30%</td>
<td>Src: 28-летний повар найден мертвым в торговом центре Сан-Франциско</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MT: <em>28-year-old</em> chef found <em>dead</em> in San Francisco <em>shopping centre</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ref: <em>28-Year-Old Chef Found Dead at San Francisco Mall</em></td>
</tr>
<tr>
<td>2</td>
<td>31/249=12%</td>
<td>Src: 28-летний повар, который недавно переехал в Сан-Франциско, был найден мертвым в лестничном пролете местного торгового центра на этой неделе.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MT: <em>the 28-year-old chef, who has recently moved to San Francisco, was found dead in the stairwell of a local shopping centre this week.</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ref: <em>A 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week.</em></td>
</tr>
<tr>
<td>3</td>
<td>111/262=42%</td>
<td>Src: Однако брат жертвы говорит, что он не может вообразить кого-то, кто желал бы причинить ему боль, отмечая: &quot;Наконец-то дела у него шли на лад&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MT: <em>However, the victim's brother says he can't imagine anyone who would wish to cause him pain, noting: &quot;Finally he went on the lad.&quot;</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ref: <em>But the victim's brother says he can't think of anyone who would want to hurt him, saying, &quot;Things were finally going well for him.&quot;</em></td>
</tr>
</tbody>
</table>
Actual visualisation output: English–German

<table>
<thead>
<tr>
<th>Src:</th>
<th>MT:</th>
<th>Ref:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The victim's brother, Louis Galicia, told ABC station KGO in San Francisco that Frank, previously a line cook in Boston, had landed his dream job as line chef at San Francisco's Sons &amp; Daughters restaurant six months ago.</td>
<td>Der Bruder des Opfers, Louis Galicien, erzählte ABC-Station KGO in San Francisco, dass Frank, zuvor ein Line-Koch in Boston, seinen Traumjob als Linienchef im Restaurant Sons &amp; Daughters von San Francisco vor sechs Monaten gelandet hatte.</td>
<td>Der Bruder des Opfers, Louis Galicia, teilte dem ABS Sender KGO in San Francisco mit, dass Frank, der früher als Koch in Boston gearbeitet hat, vor sechs Monaten seinen Traumjob als Koch im Sons &amp; Daughters Restaurant in San Francisco ergriffen hatte.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>150/489=31%</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Src:</th>
<th>MT:</th>
<th>Ref:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A spokesperson for Sons &amp; Daughters said they were &quot;shocked and devastated&quot; by his death.</td>
<td>Eine Sprecherin von Sons &amp; Daughters sagte, sie seien durch seinen Tod &quot;geschokt und verwüstet&quot; worden.</td>
<td>Ein Sprecher des Sons &amp; Daughters sagte, dass sie über seinen Tod &quot;schockiert und am Boden zerstört seien&quot;.</td>
</tr>
</tbody>
</table>

| 7 | 69/211=33% |
Method description

**CharCut** consists of three phases:

1. an iterative segmentation
   by 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.
Introduction
  Background
  Method description

Proposed method
  Iterative segmentation
  Identification of string shifts
  Scoring scheme

Comparison with other metrics

Conclusion
Recursive character-based longest-first approach, starting with $C_0 =$ the MT output segment and $R_0 =$ the human reference segment.

\[
\begin{align*}
C_{n+1} &= C_n - \text{LCSSubstr}(C_n, R_n) \\
R_{n+1} &= R_n - \text{LCSSubstr}(C_n, R_n)
\end{align*}
\]
Problem with character-based longest-first approach

Problem: Counter-intuitive segmentation.

\[
C: \ldots \text{der\Europäischen\Gemeinsamen Strategie zur Unterstützung Palästinas \ldots} \\
R: \ldots \text{der\Gemeinsamen\Europäischen Strategie zur Unterstützung Palästinas \ldots}
\]
Problem with character-based longest-first approach

Problem: Counter-intuitive segmentation.

\[
C: \quad \ldots \text{der Europäischen Gemeinsamen Strategie zur Unterstützung Palästinas} \ldots \\
R: \quad \ldots \text{der Gemeinsamen Europäischen Strategie zur Unterstützung Palästinas} \ldots \\
\]

- The same ending is shared by the two swapped words \textit{Europäischen} and \textit{Gemeinsamen};
- This ending has been integrated into the LCSubstr;
- This prevents the more natural full word matches.
Problem with character-based longest-first approach

Problem: Counter-intuitive segmentation.

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

▶ The same ending is shared by the two swapped words Europäischen and Gemeinsamen; 
▶ This ending has been integrated into the LCSubstr; 
▶ This prevents the more natural full word matches.

Answer: Making the method aware of word separators.
Proposed method

Making the method aware of word separators

When searching for LCSubstr, consider only substr. of $C_0$ and $R_0$ of the three following types:

- Substring inside one word only, including spaces and punctuations
  - Ex.: Hello, world!!!

- Several entire words, including beginning and end spaces or punctuations
  - Ex.: Hello, world!!!

- Run of non-word characters
  - Ex.: Hello, world!!!
Making the method aware of word separators

When searching for LCSubstr, consider only substr. of $C_0$ and $R_0$ of the three following types:

- Substring inside one word only, including spaces and punctuations
  
  Ex.: Hello,\_world!!!
Making the method aware of word separators

When searching for LCSubstr, consider only substr. of $C_0$ and $R_0$ of the three following types:

- Substring inside one word only, including spaces and punctuations
  
  Ex.: Hello, world!!

- Several entire words, including beginning and end spaces or punctuations
  
  Ex.: Hello, world!!!
Making the method aware of word separators

When searching for LCSubstr, consider only substr. of $C_0$ and $R_0$ of the three following types:

- Substring inside one word only, including spaces and punctuations
  
  Ex.: Hello, world!!!

- Several entire words, including beginning and end spaces or punctuations
  
  Ex.: Hello, world!!!

- Run of non-word characters
  
  Ex.: Hello, world!!!
Proposed method

Iterative segmentation

Longest common prefixes and suffixes

- The longest common prefix and the longest common suffix between $C_0$ and $R_0$ are added 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 fixes frequent cases of true negatives
  - such as final punctuations or
  - segments shorter than the minimum match size which are usually felt as matches.

Lardilleux and Lepage

IWLST 2017
Longest common prefixes and suffixes

- The longest common prefix and the longest common suffix between $C_0$ and $R_0$ are added 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 fixes frequent cases of true negatives
  - such as final punctuations or
  - segments shorter than the minimum match size which are usually felt as matches.

- Experiments showed no impact in terms of correlation with human judgement.
End of iterative segmentation

- Stop when \( \text{length}(\text{LCSSubstr}(C_n, R_n)) < \) some threshold (typically 3)
- Add longest common prefixes and suffixes.
- The set of LCSSubstr’s extracted up to last step \( n \)
  (including longest common prefix and suffix) are matches;
- The remaining strings,
  i.e., the last computed \( C_n \) and \( R_n \), are loose differences.
**Example of iterative search for longest common substrings**

<table>
<thead>
<tr>
<th>n</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>$\text{LCSstr}(C_n, R_n)$</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></td>
<td></td>
</tr>
</tbody>
</table>
Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>n</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>LCSSubstr($C_n$, $R_n$)</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></td>
<td></td>
</tr>
<tr>
<td>1</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>
</tbody>
</table>
### Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>n</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>$\text{LCSubstr}(C_n, R_n)$</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></td>
<td></td>
</tr>
<tr>
<td>1</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>2</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>Before_the_</td>
<td>11</td>
</tr>
</tbody>
</table>
Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>n</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>LCSSubstr($C_n, R_n$)</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></td>
<td></td>
</tr>
<tr>
<td>1</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>2</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>Before_the_</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>_game, it had arrived at the stadium to riots.</td>
<td>_match there was a riot in the stadium.</td>
<td>_riot</td>
<td>5</td>
</tr>
</tbody>
</table>
Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>( \text{LCSSubstr}(C_n, R_n) )</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></td>
<td></td>
</tr>
<tr>
<td>1</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>2</td>
<td>Before_the_game, it had arrived at</td>
<td>to_riots.</td>
<td>Before_the_match there was a_riot</td>
<td>Before_the_</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>match there was a_riot_in</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>match there was a</td>
<td>at</td>
</tr>
</tbody>
</table>
Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>( n )</th>
<th>( C_n )</th>
<th>( R_n )</th>
<th>( \text{LCS} \text{Substr}(C_n, R_n) )</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></td>
<td></td>
</tr>
<tr>
<td>1</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>2</td>
<td>Before_the_game, it had arrived at the stadium to riots.</td>
<td>Before_the_match_there_was_a_riot in.</td>
<td>Before_the_</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>game, it had arrived at the stadium to riots.</td>
<td>match_there_was_a_riot_in.</td>
<td>riot</td>
<td>5</td>
</tr>
</tbody>
</table>
Example of iterative search for longest common substrings

<table>
<thead>
<tr>
<th>$n$</th>
<th>$C_n$</th>
<th>$R_n$</th>
<th>$\text{LCS} \left( C_n, R_n \right)$</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></td>
<td></td>
</tr>
<tr>
<td>1</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>2</td>
<td>Before_the_game, it had arrived at to riots.</td>
<td>Before_the_match_there_was_a_riot in.</td>
<td>Before_the_</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>match_there_was_a_riot in.</td>
<td>riot</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>match_there_was_a_in</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>
Example of segmentation

\[ C_0: \text{Before the game, it had arrived at the stadium to riots.} \]
\[ R_0: \text{Before the match there was a riot in the stadium.} \]

- LCSBstr’s are in black.
- Remaining substrings (in red and blue) are loose differences.
Visualising string shifts

\[ C_0: \text{Before the game, it had arrived at the stadium to riot.} \]

\[ R_0: \text{Before the match there was a riot in the stadium.} \]

- Here, the stadium and riot are crossed.
- For the purpose of visualisation,
  - the shortest one (riot) is marked as a shift,
  - and the other one as a regular match.
Proposed method

Identification of string shifts

Identifying string shifts

\[ C_{\text{match}} = \text{Before} \text{"the"} \text{"the"} \text{"stadium"} \text{"riot"}. \]
\[ R_{\text{match}} = \text{Before} \text{"the"} \text{"riot"} \text{"the"} \text{"stadium"}. \]

To identify string shifts:

- determine longest subsequence of tokens (LCStr’s)
- longest is defined in number of chars, not tokens.

Here: \( C_{\text{match}} = \text{Before} \text{"the"} \text{"the"} \text{"stadium"} \text{"riot"}. \) (12+11+1=24 chars)
Identifying string shifts

\[
C_{\text{match}} = \text{Before} \| \text{the} \| \text{the stadium} \| \text{riot} \|.
\]
\[
R_{\text{match}} = \text{Before} \| \text{riot} \| \text{the stadium} \|.
\]

To identify string shifts:

- determine longest subsequence of tokens (LCStr’s)
- longest is defined in number of chars, not tokens.
  Here: \text{Before} \| \text{the} \| \text{the stadium} \|. (12+11+1=24 \text{ chars})

Regular matches / shifts:

- Tokens in longest subsequence are regular matches.
- Tokens outside of longest subsequence are \textit{shifts}.
  Here: \textit{riot}.
Scoring scheme

Result of the iterative segmentation and identification of shifts: segmentation of input segments in 3 types of substrings:

- regular matches
- shifts
- loose differences, i.e.,
  - deletions from the candidate segment
  - insertions into the reference segment
Scoring scheme

Result of the iterative segmentation and identification of shifts: segmentation of input segments in 3 types of substrings:

\[
\text{Score} \propto \#\text{deletions} + \#\text{insertions} + \#\text{shifts}
\]

- regular matches
- shifts
- loose differences, i.e.,
  - deletions from the candidate segment
  - insertions into the reference segment
Scoring scheme

Result of the iterative segmentation and identification of shifts: segmentation of input segments in 3 types of substrings:

\[
\text{Score} \propto \#\text{deletions} + \#\text{insertions} + \#\text{shifts}
\]

Indiv. score

- regular matches 0
- shifts
- loose differences, i.e.,
  - deletions from the candidate segment
  - insertions into the reference segment
Scoring scheme

Result of the iterative segmentation and identification of shifts: segmentation of input segments in 3 types of substrings:

\[ \text{Score } \propto \#\text{deletions} + \#\text{insertions} + \#\text{shifts} \]

Indiv. score

- regular matches 0
- shifts
- loose differences, i.e.,
  - deletions from the candidate segment 1
  - insertions into the reference segment 1
Result of the iterative segmentation and identification of shifts: segmentation of input segments in 3 types of substrings:

Score $\propto \#\text{deletions} + \#\text{insertions} + \#\text{shifts}$

Indiv. score

- regular matches $0$
- shifts (counted once although appear in both segments) $1$
- loose differences, i.e.,
  - deletions from the candidate segment $1$
  - insertions into the reference segment $1$
Optimizing for correlation with human judgement

Two different normalisations:

- **total length of candidate and reference** (intuitive)
  - ⇒ score between 0 and 1:
    
    \[
    \text{score}_{\text{orig}} = \frac{\#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{|C_0| + |R_0|} \quad (2)
    \]

- **length of candidate only** (Wang et al., 2016)
  - ⇒ higher correlation with human judgements
    
    \[
    \text{score}_C = \min \left( 1, \frac{\#\text{deletions} + \#\text{insertions} + \#\text{shifts}}{2 \times |C_0|} \right) \quad (3)
    \]
Pearson correlation for the two scoring schemes

Absolute Pearson correlation against minimum match size in characters (length-based threshold) (system DA, segment-DA, segment-HUME)
Comparison with other metrics

Introduction
  Background
  Method description

Proposed method
  Iterative segmentation
  Identification of string shifts
  Scoring scheme

Comparison with other metrics

Conclusion
Comparison with other metrics

Comparison

▶ With metrics that took part in WMT16 tasks
  ▶ system-level DA
  ▶ segment-level DA
  ▶ segment-level HUME

▶ Criterion: average Pearson correlation coefficients over all language pairs.
Comparison with other metrics

Comparison

- With metrics that took part in WMT16 tasks
  - system-level DA
  - segment-level DA
  - segment-level HUME
- Criterion: average Pearson correlation coefficients over all language pairs.

Notations:
- Brackets = metrics that did not participate in the English-to-Russian evaluation (i.e., one less figure used);
- Asterisks = our own runs;
- Everything else = figures from (Bojar et al., 2016).
## System-level DA

<table>
<thead>
<tr>
<th>Metric</th>
<th>Avg. corr. ± stddev.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UoW.ReVaL</strong></td>
<td>(0.972 ± 0.013)</td>
</tr>
<tr>
<td>MPEDA</td>
<td>0.945 ± 0.044</td>
</tr>
<tr>
<td><em>CharCut</em></td>
<td>0.942 ± 0.037</td>
</tr>
<tr>
<td>chrF2</td>
<td>0.934 ± 0.038</td>
</tr>
<tr>
<td>chrF3</td>
<td>0.934 ± 0.035</td>
</tr>
<tr>
<td>*Lev. distance</td>
<td>0.930 ± 0.049</td>
</tr>
<tr>
<td>BEER</td>
<td>0.928 ± 0.054</td>
</tr>
<tr>
<td>chrF1</td>
<td>0.927 ± 0.051</td>
</tr>
<tr>
<td>CharacTER</td>
<td>0.922 ± 0.055</td>
</tr>
<tr>
<td>mtevalNIST</td>
<td>0.886 ± 0.068</td>
</tr>
<tr>
<td>mtevalBLEU</td>
<td>0.867 ± 0.060</td>
</tr>
<tr>
<td>mosesCDER</td>
<td>0.861 ± 0.061</td>
</tr>
<tr>
<td>mosesTER</td>
<td>0.851 ± 0.061</td>
</tr>
<tr>
<td>mosesPER</td>
<td>0.842 ± 0.096</td>
</tr>
<tr>
<td>wordF3</td>
<td>0.836 ± 0.069</td>
</tr>
<tr>
<td>wordF2</td>
<td>0.836 ± 0.069</td>
</tr>
<tr>
<td>wordF1</td>
<td>0.831 ± 0.071</td>
</tr>
<tr>
<td>mosesWER</td>
<td>0.812 ± 0.099</td>
</tr>
<tr>
<td>mosesBLEU</td>
<td>0.810 ± 0.082</td>
</tr>
</tbody>
</table>
## Segment-level DA

<table>
<thead>
<tr>
<th>Metric</th>
<th>Avg. corr. ± stddev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPMFcomb</td>
<td>(0.633 ± 0.048)</td>
</tr>
<tr>
<td>METRICS-F</td>
<td>(0.631 ± 0.049)</td>
</tr>
<tr>
<td>COBALT-F.</td>
<td>(0.617 ± 0.040)</td>
</tr>
<tr>
<td>MPEDA</td>
<td>0.584 ± 0.053</td>
</tr>
<tr>
<td>*CharCut</td>
<td>0.582 ± 0.076</td>
</tr>
<tr>
<td>UPF-cobalt</td>
<td>(0.582 ± 0.060)</td>
</tr>
<tr>
<td>chrF3</td>
<td>0.560 ± 0.082</td>
</tr>
<tr>
<td>chrF2</td>
<td>0.559 ± 0.081</td>
</tr>
<tr>
<td>*Lev. distance</td>
<td>0.556 ± 0.065</td>
</tr>
<tr>
<td>BEER</td>
<td>0.556 ± 0.082</td>
</tr>
<tr>
<td>chrF1</td>
<td>0.548 ± 0.079</td>
</tr>
<tr>
<td>*CharacTER</td>
<td>0.537 ± 0.074</td>
</tr>
<tr>
<td>UoW.ReVal</td>
<td>0.530 ± 0.035</td>
</tr>
</tbody>
</table>

...
## Segment-level HUME

<table>
<thead>
<tr>
<th>Metric</th>
<th>Avg. corr. ± stddev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrF3</td>
<td>0.519 ± 0.096</td>
</tr>
<tr>
<td>chrF2</td>
<td>0.517 ± 0.092</td>
</tr>
<tr>
<td>BEER</td>
<td>0.513 ± 0.079</td>
</tr>
<tr>
<td>chrF1</td>
<td>0.503 ± 0.079</td>
</tr>
<tr>
<td>MPEDA</td>
<td>0.492 ± 0.073</td>
</tr>
<tr>
<td><em>CharCut</em></td>
<td>0.483 ± 0.121</td>
</tr>
<tr>
<td>wordF3</td>
<td>0.452 ± 0.092</td>
</tr>
<tr>
<td>wordF2</td>
<td>0.450 ± 0.091</td>
</tr>
<tr>
<td>wordF1</td>
<td>0.439 ± 0.088</td>
</tr>
<tr>
<td><em>CharacTer</em></td>
<td>0.438 ± 0.126</td>
</tr>
<tr>
<td><em>Lev. distance</em></td>
<td>0.437 ± 0.109</td>
</tr>
<tr>
<td>sentBLEU</td>
<td>0.401 ± 0.101</td>
</tr>
<tr>
<td><em>TER</em></td>
<td>0.394 ± 0.125</td>
</tr>
</tbody>
</table>
Analysis of the comparison with other metrics

▶ High correlation with human judgment
Analysis of the comparison with other metrics

- **High correlation** with human judgment

- **Comparison with light metrics:**
  - Top average correl. on system- and segment-level DA eval. compared with \texttt{CHRF}, \texttt{WORDF}, \texttt{CharacTER}
  - Much higher correl. than \texttt{BLEU} and \texttt{TER}
Analysis of the comparison with other metrics

- **High correlation** with human judgment
- **Comparison with light metrics:**
  - Top average correl. on system- and segment-level DA eval. compared with \textit{chrF}, \textit{wordF}, Charac\textit{TER}
  - Much higher correl. than BLEU and TER
- **Comparison with trained metrics:**
  - On par with MPEDA (relies on additional training corpora)
Comparison with other metrics

Speed

On a 2.8 GHz processor, for Python implementations:

<table>
<thead>
<tr>
<th>Metric</th>
<th>segments/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrF</td>
<td>600</td>
</tr>
<tr>
<td>CharCut</td>
<td>260</td>
</tr>
<tr>
<td>CharacTER</td>
<td>54</td>
</tr>
</tbody>
</table>
Conclusion

Introduction
  Background
  Method description

Proposed method
  Iterative segmentation
  Identification of string shifts
  Scoring scheme

Comparison with other metrics

Conclusion
Conclusion

**CHARCUT**: character-based machine translation evaluation metric.

- Is language independent.
- Requires no additional resource or training.
Conclusion

**CHARCUT**: character-based machine translation evaluation metric.

- Is language independent.
- Requires no additional resource or training.
- Relies on **loose differences**, residuals of iterative search for longest common substrings.
- Was initially designed for displaying differences between reference and candidate segments to end users.
- Produces scores that directly reflect differences.
Conclusion

**CharCut**: character-based machine translation evaluation metric.

- Is language independent.
- Requires no additional resource or training.
- Relies on *loose differences*, residuals of iterative search for longest common substrings.
- Was initially designed for displaying differences between reference and candidate segments to end users.
- Produces scores that directly reflect differences.
- Exhibits good correlation with human judgement.
Conclusion

**CharCut:** character-based machine translation evaluation metric.

- Is language independent.
- Requires no additional resource or training.
- Relies on loose differences, residuals of iterative search for longest common substrings.
- Was initially designed for displaying differences between reference and candidate segments to end users.
- Produces scores that directly reflect differences.
- Exhibits good correlation with human judgement.

Good visual representation $\Rightarrow$ High correlation with human judgement
Future work

- Finer handling of shifts as CharCut is currently unaware of shift distance;
- Automatic correlation of the minimum match size with the number of highlighted substrings in order to keep outputs readable even with very different input segments.
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).
<table>
<thead>
<tr>
<th>Seg. id</th>
<th>Score</th>
<th>Segment comparison: Deletion Insertion Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19/50= 38%</td>
<td>MT: Thank you for listening. Ref: Thanks for your attention.</td>
</tr>
<tr>
<td>Total</td>
<td>19/50= 38%</td>
<td></td>
</tr>
</tbody>
</table>
Interface convention

- The interface is kept slick on purpose.
- It uses only classical colours:
  - red for deletions,
  - blue for insertions,
  - bold for shifts,
  - yellow background for matching substrings when pointed with the mouse.
- The scores directly reflect the number of highlighted characters.
### HTML sample output (WMT17 English-Chinese, 2-char min match size)

<table>
<thead>
<tr>
<th>Seg. id</th>
<th>Score</th>
<th>Segment comparison: Deletion Insertion Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Src:</strong> 28-Year-Old Chef Found Dead at San Francisco Mall</td>
</tr>
<tr>
<td>1</td>
<td>69%</td>
<td><strong>MT:</strong> 28岁的 Chef Fand 死在旧金山商城</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Ref:</strong> 28岁厨师被发现死于旧金山一家商场</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Src:</strong> A 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week.</td>
</tr>
<tr>
<td>2</td>
<td>30%</td>
<td><strong>MT:</strong> 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。</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Ref:</strong> 近日刚搬到旧金山的一位28岁厨师本周被发现死于当地一家商场的楼梯间。</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Src:</strong> But the victim's brother says he can't think of anyone who would want to hurt him, saying, &quot;Things were finally going well for him.&quot;</td>
</tr>
<tr>
<td>3</td>
<td>61%</td>
<td><strong>MT:</strong> 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。”</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Ref:</strong> 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。”</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total:</strong> 51%</td>
</tr>
</tbody>
</table>

Lardilleux and Lepage

IWLST 2017
References
