Treatment of Markup in Statistical Machine Translation

Master's Thesis

Mathias Müller
Motivation

Researchers often give themselves the luxury of pretending that only pure text matters [...] (Joanis et al., 2013)

- tendency to overlook “applied” problems
One of those applied problems: markup handling

Ich bitte Sie, sich zu einer Schweigeminute zu erheben.
Ich bitte Sie, sich zu einer Schweigeminute zu **erheben**.

Ich bitte Sie, sich zu einer Schweigeminute zu **erheben**."}
One of those applied problems: markup handling

Ich bitte Sie, sich zu einer Schweigeminute zu <\textbf{erheben}</\textbf{e}>.
One of those applied problems: markup handling

Ich bitte Sie, sich zu einer Schweigeminute zu <b>erheben</b>.

Please <b>rise</b> then, for this minute of silence.
Ich bitte Sie, sich zu einer Schweigeminute zu erheben.

Please rise then, for this minute of silence.
Why is markup handling important?

(1) Translation industry has a need to translate content with markup

(2) Markup errors harm translator productivity

---

see e.g. O’Brien (2011); Joanis et al. (2013); Parra Escartín and Arcedillo (2015)
Existing solutions

Complete list of popular machine translation frameworks that have markup handling:
Existing solutions

- Is the code that implements markup handling available to the public?
- Are there empirical comparisons of the proposed method?

<table>
<thead>
<tr>
<th>publication</th>
<th>code</th>
<th>experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du et al. (2010)</td>
<td>❌</td>
<td>✅</td>
</tr>
<tr>
<td>Zhechev and van Genabith (2010)</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Hudík and Ruopp (2011)</td>
<td>✔</td>
<td>❌</td>
</tr>
<tr>
<td>Tezcan and Vandeghinste (2011)</td>
<td>❌</td>
<td>✅</td>
</tr>
<tr>
<td>Joanis et al. (2013)</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>
Existing solutions

- Is there pseudo code or a clear explanation of the algorithm?

<table>
<thead>
<tr>
<th>solution</th>
<th>code</th>
<th>experiments</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>D (2010)</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Z (2010)</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>H (2011)</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>T (2011)</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>J (2013)</td>
<td>×</td>
<td>(✓)</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>solution</th>
<th>code</th>
<th>experiments</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>thesis work</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
The mtrain framework

Code: https://gitlab.cl.uzh.ch/mt/mtrain
Implemented strategies

- two general classes of strategies, **masking** and **reinsertion**
- how do they solve the problem of markup handling?
How to solve markup handling?
How to solve markup handling?

Make tokenization more sophisticated.

After that, hide:

- replace markup with an innocuous string (masking)
- remove markup alltogether (reinsertion)

... in any case, the original content needs to be restored after translation (seek)
How Masking Works

List of British jokes updated by rowan@kinson.com

For business inquiries call +4140356 12 25.

<Liste von britischen Witzen aktualisiert von rowan@kinson.com>

Kontakt für Firmen: +4140356 12 25.

<Fios2 id="1">{}<ept>

<Fios2 id="1">{}<ept>
How Masking Works

List of British jokes updated by rowan@kinson.com

For business inquiries call +41 40356 12 25.

Kontakt für Firmen: TEL.

Liste von britischen Witzen aktualisiert von EMAIL

XML {} XML Fios2 XML {} XML
How Masking Works

List of British jokes updated by EMAIL

For business inquiries call TEL.

This is what the final training corpus looks like!
<table>
<thead>
<tr>
<th>List of British jokes updated by <a href="mailto:rowan@kinson.com">rowan@kinson.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>For business inquiries call +4140356 12 25.</td>
</tr>
</tbody>
</table>

```xml
<bpt id="1" ctype="bold">{}
Fios2<ept id="1">{}
```

<table>
<thead>
<tr>
<th>Liste von britischen Witzen aktualisiert von <a href="mailto:rowan@kinson.com">rowan@kinson.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Kontakt für Firmen: +4140356 12 25.</td>
</tr>
</tbody>
</table>

```xml
<bpt id="1" ctype="bold">{}
Fios2<ept id="1">{}
```
How Reinsertion Works

This is what the final training corpus looks like!
Undoing masking or markup removal

Liste von britischen Witzen aktualisiert von rowan@kinson.com

original source

Liste von britischen Witzen aktualisiert von

preprocessed source

List of British jokes updated by

machine translation

？

after reinsertion
Using word alignment or similar correspondence

Liste von britischen Witzen aktualisiert von rowan@kinson.com

original source

List of British jokes updated by rowan@kinson.com

machine translation

Liste von britischen Witzen aktualisiert von

preprocessed source

List of British jokes updated by rowan@kinson.com

after reinsertion
Experiments in a nutshell

- test 5 markup handling methods
- data sets where markup is abundant
- train standard Moses SMT systems, vary only the markup handling method
- manual evaluation of performance:
  - correct
  - misplaced
  - malformed
Manual evaluation of 568 tags in XLIFF data

IM = identity masking, AM = alignment masking, SR = segmentation reinsertion, AR = alignment reinsertion, HR = hybrid reinsertion.
Manual evaluation of 584 tags in Euromarkup data

IM = identity masking, AM = alignment masking, SR = segmentation reinsertion, AR = alignment reinsertion, HR = hybrid reinsertion.
Wrapping up

The main contributions of my thesis are:

- a survey of existing solutions for markup handling in SMT
- implementations of novel and existing methods in a unified framework that is available for free
- recommendations regarding the choice markup strategy

(... and other things I could not mention in all brevity : ( ) )
Calling the \texttt{mtrain} API example

```python
>>> from mtrain.preprocessing.reinsertion import Reinserter
>>> reinserter = Reinserter('alignment')
>>> source_segment = 'Hello <g id="1" ctype="x-bold;"> World ! </g>'
# markup removal, then translation...
>>> translated_segment = 'Hallo Welt !'
>>> alignment = [(0,0), (1,1), (2,2)]
>>> reinserter.reinsert(source_segment, translated_segment, alignment)
'Hallo <g ctype="x-bold;" id="1"> Welt ! </g>'
```
<b>Thanks</b> for listening!

First question for a lively discussion:
Yeah, but does this work for neural machine translation?


Bibliography II