Integrating an Unsupervised Transliteration Model into Statistical Machine Translation

Nadir Durrani  
University of Edinburgh  
dnadir@inf.ed.ac.uk

Hassan Sajjad  
Qatar Computing Research Institute  
hsajjad@qf.org.qa

Hieu Hoang  
University of Edinburgh  
hieu.hoang,pkoehn@inf.ed.ac.uk

Abstract

We investigate three methods for integrating an unsupervised transliteration model into an end-to-end SMT system. We induce a transliteration model from parallel data and use it to translate OOV words. Our approach is fully unsupervised and language independent. In the methods to integrate transliterations, we observed improvements from 0.23-0.75 (∆ 0.41) BLEU points across 7 language pairs. We also show that our mined transliteration corpora provide better rule coverage and translation quality compared to the gold standard transliteration corpora.

1 Introduction

All machine translation (MT) systems suffer from the existence of out-of-vocabulary (OOV) words, irrespective of the amount of data available for training. OOV words are mostly named entities, technical terms or foreign words that can be translated to the target language using transliteration. Much work (Al-Onaizan and Knight, 2002; Zhao et al., 2007; Kashani et al., 2007; Habash, 2009) has been done on transliterating named entities and OOVs, and transliteration has been shown to improve MT quality. Transliteration also shown to be useful for translating closely related language pairs (Durrani et al., 2010; Nakov and Tiedemann, 2012), and for disambiguation (Hermjakob et al., 2008; Azab et al., 2013). However, despite its utility, a transliteration module does not exist in the commonly used MT toolkits, such as Moses (Koehn et al., 2007). One of the main reasons is that the training data, a corpus of transliteration pairs, required to build a transliteration system, is not readily available for many language pairs. Even if such a training data is available, mechanisms to integrate transliterated words into MT pipelines are unavailable in these toolkits. Generally, a supervised transliteration system is trained separately outside of an MT pipeline, and a naïve approach, to replace OOV words with their 1-best transliterations in the post/pre-processing step of decoding is commonly used.

In this work i) we use an unsupervised model based on Expectation Maximization (EM) to induce transliteration corpus from word aligned parallel data, which is then used to train a transliteration model, ii) we investigate three different methods for integrating transliteration during decoding, that we implemented within the Moses toolkit. To the best of our knowledge, our work is the foremost attempt to integrate unsupervised transliteration model into SMT.

This paper is organized as follows. Section 2 describes the unsupervised transliteration mining system, which automatically mines transliteration pairs from the same word-aligned parallel corpus as used for training the MT system. Section 3 describes the transliteration model that is trained using the automatically extracted pairs. Section 4 presents three methods for incorporating transliteration into the MT pipeline, namely: i) replacing OOVs with the 1-best transliteration in a post-decoding step, ii) selecting the best transliteration from the list of n-best transliterations using transliteration and language model features in a post-decoding step, iii) providing a transliteration phrase-table to the decoder on the fly where it can consider all features to select the best transliteration of OOV words. Section 5 presents results. Our integrations achieved an average improvement of 0.41 BLEU points over a competitive baseline across 7 language pairs (Arabic, Bengali, Farsi, Hindi, Russian, Telugu and Urdu-into-English). An additional experiment showed that our system provides better rule coverage as opposed to another built from gold standard transliteration corpus and produces better translations.
2 Transliteration Mining

The main bottleneck in building a transliteration system is the lack of availability of transliteration training pairs. It is, however, fair to assume that any parallel data would contain a reasonable number of transliterated word pairs. Transliteration mining can be used to extract such word pairs from the parallel corpus. Most previous techniques on transliteration mining generally use supervised and semi-supervised methods (Sherif and Kondrak, 2007; Jiampojamarn et al., 2010; Darwish, 2010; Kahki et al., 2012). This constrains the mining solution to language pairs for which training data (seed data) is available. A few researchers proposed unsupervised approaches to mine transliterations (Lee and Choi, 1998; Sajjad et al., 2011; Lin et al., 2011). We adapted the work of Sajjad et al. (2012) as summarized below.

Model: The transliteration mining model is a mixture of two sub-models, namely: a transliteration and a non-transliteration sub-model. The idea is that the transliteration model would assign higher probabilities to transliteration pairs compared to the probabilities assigned by a non-transliteration model to the same pairs. Consider a word pair \((e, f)\), the transliteration model probability for the word pair is defined as follows:

\[
p_{tr}(e, f) = \sum_{a \in \text{Align}(e, f)} \prod_{j=1}^{[a]} p(q_j)
\]

where \(\text{Align}(e, f)\) is the set of all possible sequences of character alignments, \(a\) is one alignment sequence and \(q_j\) is a character alignment.

The non-transliteration model deals with the word pairs that have no character relationship between them. It is modeled by multiplying source and target character unigram models:

\[
p_{ntr}(e, f) = \prod_{i=1}^{[e]} p_E(e_i) \prod_{i=1}^{[f]} p_F(f_i)
\]

The transliteration mining model is defined as an interpolation of the transliteration sub-model and the non-transliteration sub-model:

\[
p(e, f) = (1 - \lambda)p_{tr}(e, f) + \lambda p_{ntr}(e, f)
\]

\(\lambda\) is the prior probability of non-transliteration.

The non-transliteration model does not change during training. We compute it in a pre-processing step. The transliteration model learns character alignment using expectation maximization (EM). See Sajjad et al. (2012) for more details.

3 Transliteration Model

Now that we have transliteration word pairs, we can learn a transliteration model. We segment the training corpus into characters and learn a phrase-based system over character pairs. The transliteration model assumes that source and target characters are generated monotonically.\(^1\) Therefore we do not use any reordering models. We use 4 basic phrase-translation features (direct, inverse phrase-translation, and lexical weighting features), language model feature (built from the target-side of mined transliteration corpus), and word and phrase penalties. The feature weights are tuned\(^2\) on a dev-set of 1000 transliteration pairs.

4 Integration to Machine Translation

We experimented with three methods for integrating transliterations, described below:

Method 1: involves replacing OOVs in the output with the 1-best transliteration. The success of Method 1 is solely contingent on the accuracy of the transliteration model. Also, it ignores context which may lead to incorrect transliteration. For example, the Arabic word 
\(\text{بِل} \) transliterates to “Bill” when followed by “Clinton” and “Bell” if preceded by “Alexander Graham”.

Method 2: provides n-best transliterations to a monotonic decoder that uses a monolingual language model and a transliteration phrase-translation table to re-score transliterations. We carry forward the 4 translation model features used in the transliteration system to build a transliteration phrase-table. We additionally use an LM-OOV feature which counts the number of words in a hypothesis that are unknown to the language model. Smoothing methods such as Kneser-Ney assign significant probability mass to unseen events, which may cause the decoder to make incorrect transliteration selection. The LM-OOV feature acts as a prior to penalize such hypotheses.

Method 3: Method 2 can not benefit from all decoding features and phenomenon like reordering. It transliterates Urdu compound 
\(\text{حرب} \) (Arabian Sea) to “Sea Arabian”, if \(\text{حرب} \) is an unknown word. In method 3, we feed the transliteration phrase-table directly into the first-pass decoding which allows reordering of UNK words.

\(^1\)Mining algorithm also makes this assumption.

\(^2\)Tuning data is subtracted from the training corpus while tuning to avoid over-fitting. After the weights are tuned, we add it back, retrain GIZA, and estimate new models.
use the \textit{decoding-graph-backoff} option in Moses, that allows multiple translation phrase tables and back-off models. As in method 2, we also use the LM-OOV feature in method 3.3

5 Evaluation

Data: We experimented with 7 language pairs, namely: Arabic, Bengali, Farsi, Hindi, Russian, Telugu and Urdu-into-English. For Arabic\textsuperscript{4} and Farsi, we used the TED talks data (Cettolo et al., 2012) made available for IWSLT-13, and we used the dev2010 set for tuning and the test2011 and test2012 sets for evaluation. For Indian languages we used the Indic multi-parallel corpus (Post et al., 2012), and we used the dev and test sets provided with the parallel corpus. For Russian, we used WMT-13 data (Bojar et al., 2013), and we used half of the news-test2012 for tuning and other half for testing. We also evaluated on the news-test2013 set. For all, we trained the language model using the monolingual WMT-13 data. See Table 1 for data statistics.

<table>
<thead>
<tr>
<th>Lang</th>
<th>Train\textsubscript{lm}</th>
<th>Train\textsubscript{tr}</th>
<th>Dev</th>
<th>Test\textsubscript{1}</th>
<th>Test\textsubscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>152K</td>
<td>6795</td>
<td>887</td>
<td>1434</td>
<td>1704</td>
</tr>
<tr>
<td>BN</td>
<td>24K</td>
<td>1916</td>
<td>775</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>FA</td>
<td>79K</td>
<td>4039</td>
<td>852</td>
<td>1185</td>
<td>1116</td>
</tr>
<tr>
<td>HI</td>
<td>39K</td>
<td>4719</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>RU</td>
<td>2M</td>
<td>302K</td>
<td>1501</td>
<td>1502</td>
<td>3000</td>
</tr>
<tr>
<td>TE</td>
<td>45K</td>
<td>4924</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>UR</td>
<td>87K</td>
<td>9131</td>
<td>980</td>
<td>883</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: No. of sentences in Training Data and Mined Transliteration Corpus (Types) (Train\textsubscript{tr})

Baseline Settings: We trained a Moses system replicating the settings used in competition-grade systems (Durrani et al., 2013b; Birch et al., 2013): a maximum sentence length of 80, GDF\textsuperscript{5} symmetrization of GIZA++ alignments (Och and Ney, 2003), an interpolated Kneser-Ney smoothed 5-gram language model with KenLM (Heafield, 2011) used at runtime, a 5-gram OSM (Post et al., 2012), and we used the k-best batch MIRA (Cherry and Foster, 2012).5

Transliteration Miner: The miner extracts transliterations from a word-aligned parallel corpus. We only used word pairs with 1-to-1 alignments.6 Before feeding the list into the miner, we cleaned it by removing digits, symbols, word pairs where source or target is composed from less than 3 characters, and words containing foreign characters that do not belong to this scripts. We ran the miner with 10 iterations of EM. The number of transliteration pairs (types) extracted for each language pair is shown in Table 1 (Train\textsubscript{tr}).

Transliteration System: Before evaluating our integrations into the SMT system, we performed an intrinsic evaluation of the transliteration system that we built from the mined pairs. We formed test data for Arabic–English (1799 pairs), Hindi–English (2394 pairs) and Russian–English (1859 pairs) by concatenating the seed data and gold standard transliteration pairs both provided for the Shared Task on Transliteration mining (Kumaran et al., 2010). Table 2 shows precision and recall of the mined transliteration system (MTS).

<table>
<thead>
<tr>
<th></th>
<th>AR</th>
<th>HI</th>
<th>RU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prec (1-best Accuracy)</td>
<td>20.0%</td>
<td>25.3%</td>
<td>46.1%</td>
</tr>
<tr>
<td>Recall (100-best Accuracy)</td>
<td>80.2%</td>
<td>79.3%</td>
<td>87.5%</td>
</tr>
</tbody>
</table>

Table 2: Precision and Recall of MTS

The precision (1-best accuracy) of the transliteration model is quite low. This is because the transliteration corpus is noisy and contains imperfect transliteration pairs. For example, the miner extracted the pair \textit{Arabia} \rightarrow \textit{استراليا} (Australia), while the correct transliteration is “Australia”. We can improve the precision by tightening the mining threshold probability. However, our end goal is to improve end-to-end MT and not the transliteration system. We observed that recall is more important than precision for overall MT quality. We provide an empirical justification for this when discussing the final experiments.

\textsuperscript{3}Method 3 is desirable in cases where the decoder can translate or transliterate a word. For example Hindi word \textit{रीमा} can be translated to “Border” and also transliterated to name “Seema”. Identifying such candidates that can be translated or transliterated is a challenge. Machine learning techniques (Goldwasser and Roth, 2008; Kirschenbaum and Wintner, 2009) and named entity recognizers (Klementiev and Roth, 2006; Hermjakob et al., 2008) have been used for this purpose. Though, we only focus on OOV words, method 3 can be used if such a classifier/NE tagger is available.

\textsuperscript{4}Arabic and Urdu are segmented using MADA (Habash and Sadat, 2006) and UWS (Durrani and Hussain, 2010).

\textsuperscript{5}Retuning the transliteration features was not helpful, default weights are used.

\textsuperscript{6}M-N/1-N alignments are less likely to be transliterations.
MT Experiments: Table 3 gives a comprehensive evaluation of the three methods of integration discussed in Section 4 along with the number7 of OOV words (types) in different tests. We report BLEU gains (Papineni et al., 2002) obtained by each method. Method 1 (M1), that replaces OOV words with 1-best transliteration gave an average improvement of +0.13. This result can be attributed to the low precision of the transliteration system (Table 2). Method 2 (M2), that transliterates OOVs in second pass monotonic decoding, gave an average improvement of +0.39. Slightly higher gains were obtained using Method 3 (M3), that integrates transliteration phrase-table inside decoder on the fly. However, the efficacy of M3 in comparison to M2 is not as apparent, as M2 produced better results than M3 in half of the cases.

<table>
<thead>
<tr>
<th>Lang</th>
<th>Test</th>
<th>B0</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>iwslt11</td>
<td>26.75</td>
<td>+0.12</td>
<td>+0.36</td>
<td>+0.25</td>
<td>587</td>
</tr>
<tr>
<td></td>
<td>iwslt12</td>
<td>29.03</td>
<td>+0.10</td>
<td>+0.30</td>
<td>+0.27</td>
<td>682</td>
</tr>
<tr>
<td>BN</td>
<td>jhu12</td>
<td>16.29</td>
<td>+0.12</td>
<td>+0.42</td>
<td>+0.46</td>
<td>1239</td>
</tr>
<tr>
<td>FA</td>
<td>iwslt11</td>
<td>20.85</td>
<td>+0.10</td>
<td>+0.40</td>
<td>+0.31</td>
<td>559</td>
</tr>
<tr>
<td></td>
<td>iwslt12</td>
<td>16.26</td>
<td>+0.04</td>
<td>+0.20</td>
<td>+0.26</td>
<td>400</td>
</tr>
<tr>
<td>HI</td>
<td>jhu12</td>
<td>15.64</td>
<td>+0.21</td>
<td>+0.35</td>
<td>+0.47</td>
<td>1629</td>
</tr>
<tr>
<td>RU</td>
<td>wmt12</td>
<td>33.95</td>
<td>+0.24</td>
<td>+0.55</td>
<td>+0.49</td>
<td>434</td>
</tr>
<tr>
<td></td>
<td>wmt13</td>
<td>25.98</td>
<td>+0.25</td>
<td>+0.40</td>
<td>+0.23</td>
<td>799</td>
</tr>
<tr>
<td>TE</td>
<td>jhu12</td>
<td>11.04</td>
<td>-0.09</td>
<td>+0.40</td>
<td>+0.75</td>
<td>2343</td>
</tr>
<tr>
<td>UR</td>
<td>jhu12</td>
<td>23.25</td>
<td>+0.24</td>
<td>+0.54</td>
<td>+0.60</td>
<td>827</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>21.9</td>
<td>+0.13</td>
<td>+0.39</td>
<td>+0.41</td>
<td>950</td>
</tr>
</tbody>
</table>

Table 3: End-to-End MT Evaluation – B0 = Baseline, M1 = Method1, M2 = Method2, M3 = Method3, BLEU gains shown for each method

In an effort to test whether improving transliteration precision would improve end-to-end SMT results, we carried out another experiment. Instead of building a transliteration system from mined corpus, we built it using the gold standard corpus (for Arabic, Hindi and Russian), that we also used previously to do an intrinsic evaluation. We then replaced our mined transliteration systems with its counterpart that uses gold standard transliteration system (GTS), except in Hindi–English where both systems were equal.

<table>
<thead>
<tr>
<th>Lang</th>
<th>Test</th>
<th>MTS GST</th>
<th>MTS MTS</th>
<th>H1 H1</th>
<th>RU RU</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>iwslt11</td>
<td>27.11</td>
<td>26.99</td>
<td>29.33</td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>iwslt12</td>
<td>29.33</td>
<td>29.20</td>
<td>16.11</td>
<td>34.33</td>
</tr>
<tr>
<td>BN</td>
<td>jhu12</td>
<td>34.50</td>
<td>34.00</td>
<td>16.29</td>
<td>29.03</td>
</tr>
<tr>
<td>FA</td>
<td>iwslt11</td>
<td>26.38</td>
<td>26.85</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td></td>
<td>iwslt12</td>
<td>26.75</td>
<td>26.75</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td>HI</td>
<td>jhu12</td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td>RU</td>
<td>wmt12</td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td></td>
<td>wmt13</td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td>TE</td>
<td>jhu12</td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td>UR</td>
<td>jhu12</td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>34.33</td>
<td>34.33</td>
<td>16.26</td>
<td>29.03</td>
</tr>
</tbody>
</table>

Table 4: Comparing Gold Standard Transliteration (GST) and Mined Transliteration Systems

In the error analysis we found that the GST system suffered from sparsity and did not provide enough coverage of rules to produce right transliterations. For example, Arabic drops the determiner كل (al), but such additions were not observed in gold transliteration pairs. Arabic word الغايكسل (Gigapixel) is therefore transliterated to “algegeaksl”. Similarly the GST system learned no transliteration pairs to account for the rule “b → p” and therefore erroneously transliterated سيروك (Spurlock) to “Sbrlok”. Similar observations were true for the case of Russian–English. The rules “a → u” and “y → ϵ” were not observed in the gold set, and hence كانت (hurricane) was transliterated to “herricane” and танбо́ты (Talbot) to “Talboty”. This shows that better recall obtained from the mined pairs led to overall improvement.

6 Conclusion

We incorporated unsupervised transliteration mining model into standard MT pipeline to automatically transliterate OOV words without needing additional resources. We evaluated three methods for integrating transliterations on 7 language pairs and showed improvements ranging from 0.23-0.75 (Δ 0.41) BLEU points. We also showed that our mined transliteration corpus provide better recall and overall translation quality compared to the gold standard transliteration corpus. The unsupervised transliteration miner and its integration to SMT has been made available to the research community via the Moses toolkit.

Acknowledgments

We wish to thank the anonymous reviewers and Kareem Darwish for their valuable feedback on an earlier draft of this paper. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287658. This publication only reflects the authors’ views.

7Note that not all OOVs can be transliterated. This number is therefore an upper bound what can be transliterated.
References


