PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 93 JANUARY 2010 107–116

Integrating Output from Specialized Modules in Machine Translation

Transliterations in Joshua

Ann Irvine[a], Mike Kayser[b], Zhifei Li[a], Wren Thornton[c], Chris Callison-Burch[a]

[a] Center for Language and Speech Processing, Johns Hopkins University
[b] BBN Technologies
[c] Cognitive Science Program, Indiana University

Abstract

In many cases in SMT we want to allow specialized modules to propose translation fragments to the decoder and allow them to compete with translations contained in the phrase table. Transliteration is one module that may produce such specialized output. In this paper, as an example, we build a specialized Urdu transliteration module and integrate its output into an Urdu–English MT system. The module marks-up the test text using an XML format, and the decoder allows alternate translations (transliterations) to compete.

1. Introduction

The phrase tables used in statistical machine translation (SMT) systems are often incomplete, and they may not take full advantage of the linguistic knowledge that we have about a language. However, many data-driven NLP tools exist for specific linguistic tasks. For this reason, it is often useful to create specialized modules that employ other methods of translation not so dependent on the training text. Such modules may include noun phrase taggers and translators (Koehn and Knight, 2004), morphological analyzers, modality taggers and translators,[1] and transliteration systems. These modules may then be integrated into the MT pipeline (Dugast et al., 2007; Yang and Kirchhoff, 2006).

[1]Verbal modality expresses the notions of possibility, necessity, permission, and obligation

Corresponding author: annirvine@gmail.com

Previous SMT systems have integrated the subtask output of specialized modules using an XML-markup on input text (Koehn, 2004; Senellart et al., 2003). Here we present an XML format to integrate the output of our specialized transliteration module into the Joshua decoder (Li et al., 2009a). It necessarily differs from the XML schemes used in phrase-based decoders because Joshua is a parsing-based decoder. We illustrate its use with an example transliteration module.

2. Decoding Constraints

Joshua (Li et al., 2009a) is an open source[2] SMT system that uses synchronous context free grammars (SCFGs) as its underlying formalism. SCFGs provide a convenient and theoretically grounded way of incorporating linguistic information into statistical models of translation. Joshua implements all the essential algorithms described in (Chiang, 2007) and supports Hiero-style rules (Chiang, 2005) as well as richer syntax augmented rules (Zollmann et al., 2008). The version of Joshua that we have used in this work incorporates the grammar extraction software that comes as part of their open source SAMT toolkit.[3] Joshua translates by applying the extracted SCFG rules to the source language text using a general chart parsing framework (Li et al., 2009b).

A probabilistic SCFG consists of a set of source-language terminal symbols T_S, a set of target-language terminal symbols T_T, a set of nonterminals N that is shared between both languages, and set of production rules of the form

$$X \rightarrow \langle \gamma, \alpha, \sim, w \rangle$$

where $X \in N, \gamma \in [N \cup T_S]*$ is a (possibly mixed) sequence of nonterminals and source terminals that form the lefthand side of the rule, $\alpha \in [N \cup T_T]*$ is a (again possibly mixed) sequence of nonterminals and target-language terminals that form the right-hand side of the rule, and $\sim$ is a one-to-one correspondence between the nonterminals of γ and α. w is a weight for the production rule.

To support integrating specialized modules we introduced the ability to specify alternate translation rules in the document to be translated. In order to support both the use of alternate translation rules and regular decoding (without alternate rules), we introduced a new parameter in Joshua's configuration file for specifying which parser to use on the input file. Each input file parser reads in the file and emits a sequence of segments to be translated. In order to avoid storing the whole file in memory, the sequence is returned as a co-iterator.[4] By using a co-iterator, the sequence is produced lazily and consumed on-line, invoking the chart parser as a co-routine.

[2]http://cs.jhu.edu/~ccb/joshua/

[3]http://www.cs.cmu.edu/zollmann/samt/

[4]We use some object instance which implements the `joshua.util.CoIterator` interface. Both iterators and co-iterators are examples of abstractions over enumerations. Whereas an iterator captures the notion of producing the elements, a co-iterator captures the notion of consuming those elements.

To avoid interrupting translation in the middle of a file, we want to detect malformed files before translation begins. So in order to detect malformed file errors eagerly, the file is read once with a co-iterator that consumes the segments but does nothing with them, and then re-read to produce segments for the chart parser.

Each translation segment consists of an ID, a source sentence, and a collection of spans covering the sentence where each span contains a collection of constraints. Spans containing only soft constraints are allowed to overlap, whereas hard constraint spans may not overlap. Constraints are drawn from three types: lefthand side (LHS) constraints, righthand side (RHS) constraints, and rule constraints. LHS constraints are hard constraints specifying that the span be treated as a specified nonterminal, thus filtering the regular grammar to generate translations only from that nonterminal. One use for LHS constraints is to integrate chunking or tagging information before decoding. RHS constraints are hard constraints filtering the regular grammar such that only rules generating the desired translation can be used. A use for RHS constraints would be integrating word sense disambiguation before decoding. Rule constraints specify a new grammar rule including a LHS nonterminal, source RHS (derived from the source sentence), target RHS, and feature values. Rule constraints can be either hard or soft; if they are hard, they override the regular grammar; if they are soft, they are considered an addition to the regular grammar and will compete with regular rules. Rule constraints for any given span must be marked collectively as all hard or all soft.

2.1. XML Markup

The XML format follows straightforwardly from the specification of segments. The XML file must be valid XML, and thus must have a root element. Underneath the root element are some number of `<seg>` tags with a required `id` attribute that specifies the input segment number. The source sentence is given as raw text under the `<seg>` tag. Each `<seg>` tag may contain one or more `<span>` tags with required `start` and `end` attributes and an optional `hard` attribute for rule constraints. Each `<span>` tag must contain one or more `<constraint>` tags each of which contains an optional `<lhs>` tag, containing nonterminal text, followed by an optional `<rhs>` tag with an optional `features` attribute and containing target text. Any other tags are ignored by the XML parser.

This specification of the XML format is overly liberal and could admit files which are non-sensical or which cannot be represented internally. In order to rule out such files, the generated objects are run through a type checker to ensure semantic validity. The type checker verifies the following invariants:

- Each constraint adheres to one of the three types, thus it has
  - only a `<lhs>` tag, or
  - only a `<rhs>` tag with no `features` attribute, or
  - a `<lhs>` tag, a `<rhs>`, and a `features` attribute

- For each span,
  - the start and end indices are within the width of the sentence
  - the start index is smaller than the end index
- There are no overlapping hard spans

While the `features` attribute is considered optional in terms of the DTD for the XML grammar, that is only because DTDs are unable to capture the dependency relation between the three valid constraint types.[5] If both `<lhs>` and `<rhs>` tags are provided but there is no `feature` attribute, this is considered a type error since the constraint does not belong to any of the three types: LHS constraints do not have `<rhs>` tags, RHS constraints do not have `<lhs>` tags, and rule constraints require a `features` attribute.

3. Decoder Integration

To enforce the three kinds of constraints (i.e., rule, LHS, and RHS) during decoding, we modified the regular chart-based decoding algorithm in Joshua. Rule constraints can be hard or soft. A rule constraint provides a new translation option for a source span, in addition to those translation options (hereafter called grammar-based translations) provided by the regular grammars. If the rule constraint is hard, all the grammar-based translations will be disallowed in the final translation output. Otherwise, the new translation option will compete with those grammar-based translations, in a probabilistic manner. Different from a rule constraint, LHS and RHS constraints are always hard, meaning that a grammar-based translation will be disallowed if its LHS or RHS does not match the LHS or RHS constraint.

Figure 1 presents the modified algorithm, where lines 1, 4, and 5 are added to the regular chart-parsing algorithm in order to support manual constraints. As shown in line 1, the algorithm first adds the rule constraints (regardless of being soft or hard) into the chart so that the decoding algorithm will consider these rule translations as candidate translations. To support the **hard** constraints, the algorithm will run through two filtering processes. In line 4, if a span is within the coverage of a **hard** rule constraint, all the grammar-based rules applicable to this span will be disallowed. Similarly, all the applicable grammar rules that do not match any LHS or RHS constraints for the span will be filtered out, as shown in line 5.

4. Transliteration Module

Here we present a specialized transliteration module that uses Joshua's new XML markup. We developed an Urdu-English transliterator, which is useful because our

[5]XML Schema and RelaxNG are also unable to capture all the necessary dependency relations. Even if they can capture the tag and attribute dependencies between the three constraint types, type checking would still be necessary due to the context-sensitive restrictions on valid start and end indices depending on the length in words of the source text.

```
DecodingWithConstraints(grammars, sentence, constraints)
1  Add the rule constraints into the Chart
2  For each span [i, j] with increasing length (i.e., j − i + 1)
3      Identify applicable grammar rules for the span
4      Filter the grammar rules based on hard rule constraints
5      Filter the grammar rules based on LHS and RHS constraints
6      Add the surviving grammar rules into the chart
```

Figure 1. Constraining Chart-based Decoding with Manual Constraints.

LDC Urdu Language Pack bilingual parallel corpus [6] has only 88,108 sentence pairs with 1,586,065 English tokens and 1,664,409 Urdu tokens. In this data, we observed that 2% of words in a development set were out-of-vocabulary (OOV) with respect to the training bitext. With the help of a human annotator, we found that approximately 33% of these words were phonetically transliterable; for example, proper names or borrowed words. Introducing a module for generating transliterations and integrating that output into the output of an end-to-end MT system clearly has the potential to improve performance.

4.1. Basic Framework

We treat transliteration as a *monotone character translation* task, similar to the work of (Knight and Graehl, 1997). We used the Joshua MT system to build an Urdu transliteration module using a semantically-informed framework, described below, and several sources of transliteration pairs for training.

At training time, given a list of Urdu–English name pairs, we first perform character-to-character alignment using the freely available Berkeley Word Aligner.[7] Next, we find character-sequence pairs which conform to the alignment graph for a word pair; these are analogous to phrase pairs in phrase-based statistical MT. We build a table of such character-sequence mappings, annotated with translation probabilities. Finally we extract a frequency-annotated list of 1.3 million names from the English Gigaword corpus using a named entity tagger (Finkel and Manning, 2009). We then use this to train a character language model prior. Having trained these components, we use them in conjunction with the off-the-shelf Joshua MT decoder.

During decoding, a novel Urdu word is segmented into sequences of characters, and each character sequence is translated to an English character sequence. Unlike in the closely analogous process of phrasal machine translation, in phonetic transliteration the translated character sequences are never reordered. Transliteration hy-

[6]LDC catalog number LDC2009E12, "NIST Open MT08 Urdu Resources"

[7]http://code.google.com/p/berkeleyaligner/

potheses are scored using a log-linear model which makes use of character sequence translation scores and a character-LM prior score. The result is a single English phonetic gloss of the Urdu word.

4.2. Semantically-targeted transliteration

We trained two transliteration systems, one for person names and one for all other semantic types (including non-names). These two systems shared all components except for the character LM and the dataset used for decoder weight tuning. For the person-name transliteration system, we trained our character language model from the large list of English person names automatically extracted from the Gigaword corpus. In the non-person-name transliteration system, we trained a model from a large list of English words, without regard to semantic type.

4.3. Training the module

In order to train the transliteration module, we gathered pairs of names that were likely to be transliterations of one another. We obtained unique name pairs from three sources: the Urdu–English parallel corpus (about 2,000 pairs, extracted from alignments and checked by a human annotator), Amazon's Mechanical Turk system (over 12,000 pairs), and linked people pages from the Urdu and English Wikipedias (about 1,000 pairs). Our final system used about 15,000 pairs of Urdu–English transliterated name pairs.[8] Transliteration performance improved with increasing amounts of training data, and our final module outperformed the baseline system available from the LDC Urdu Language Pack.

4.4. Integrating the module

When using the rule constraint mechanism to add externally constructed theories to Joshua's search space, it is necessary to specify a left-hand-side nonterminal for each rule. Since this nonterminal label is used by Joshua in decoding, it is helpful to choose a label which accurately represents the grammatical content of the covered phrase. Rather than compute a single guess for this label, we add multiple arcs to the search space, each with a different nonterminal. We use the automatically determined named-entity-category of the source word to construct the set of possible labels for a transliterated word.

In particular, before decoding we compute from training the N most frequent target nonterminal labels assigned to low-frequency words of each source name category (the name categories are: Person, Location, Organization, Geopolitical Entity, Facility). At decoding time, we use an automatic name tagger to compute the semantic category of each source word. For every transliteration candidate, we add N arcs to

[8]The complete name pair list is freely available at http://www.clsp.jhu.edu/~anni/

```
<seg id="20">
پروفیسر الزبت گارڈنر کے مطابق ڈائٹنگ کرنے والے افراد فلو سے ن صرف بت جلد متاثر وجأیں گے بلک ان کو صحتیاب ونے میں بی زیاد وقت لگے گا .
<span start="2" end="3" hard="false">
 <constraint>
   <lhs>[NNP-PERSON]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0.3361808473;
   -9.0926338104;-0.5569314994;0">gardner</rhs></constraint>
 <constraint>
   <lhs>[NNP-PERSON]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0;
   -10.4980070804;-0.7064663926;0">gardiner</rhs></constraint>
 <constraint>
   <lhs>[NNP-PERSON]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0;
   -11.1132723471;-0.8203325663;0">gardener</rhs></constraint>
 <constraint>
   <lhs>[NNP-PERSON]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0
   ;0;0">##UNKNOWN##</rhs></constraint>
 <constraint>
 <lhs>[NN-PERSON]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0.3361808473;
   -9.0926338104;-0.5569314994;0">gardner</rhs></constraint>
 . . . . .
</span>
<span start="5" end="6" hard="false">
 <constraint>
   <lhs>[NNP]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0.003016175;
   -13.1667265966;-0.863364734;1">datong</rhs></constraint>
 <constraint>
   <lhs>[NNP]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0;-12.3903889565;
   -0.7339666773;1">dating</rhs></constraint>
 <constraint><lhs>[NNP]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;
   0;0;0">##UNKNOWN##</rhs></constraint>
 <constraint>
   <lhs>[NN]</lhs><rhs features="0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;-6.907755279;0.003016175;-13.1667265966;
   -0.863364734;1">datong</rhs></constraint>
 . . . . .
</seg>
```

Figure 2. An example of XML markup on Urdu text. Each span has 21 associated feature weights. The first 13 features represent the feature space of traditionally extracted Joshua translation rules. Since transliteration rules do not exist in the same feature space, these feature values are always set to 0 for transliteration rules.

the search space, each with a different label from the semantically-targeted set. In our experiments, we use the value N=5.

An example of the XML markup for an input Urdu sentence is shown in Figure 2. In this example, two word spans are tagged with constraints and hypothesis transliterations. Each span and transliteration hypothesis is tagged with the N most frequent target nonterminal labels (e.g. NNP, NN, etc. for a PERSON tag).

5. Results

We tested the impact of transliterator integration in a small number of blind submissions to the NIST MT09 Urdu–English evaluation. We integrated the transliterator into the current Joshua system. In one experiment, we transliterated all low-frequency words, while in a second experiment we transliterated only low-frequency

Joshua Translation System	**NIST MT09 BLEU Score**
No Transliteration	.2958
Transliterate names only	.2980
Transliterate all types	.3010

Table 1. Impact of transliteration on BLEU in submissions to NIST MT09 evaluation.

person names. Our baseline for comparison was the identical Joshua system without transliterations.

We compared the baseline against the transliteration-aware systems both quantitatively and qualitatively. In a quantitative comparison, whose results are in Table 1, transliteration yielded a small but notable BLEU improvement. As shown in the table, transliterating words of all semantic types yielded slightly better performance than transliterating only words marked as person names.

We also qualitatively compared the best transliteration-aware system with the baseline system via manual inspection of decoder output. As expected, some sentences showed clear improvement via the increased lexical coverage allowed by the transliteration model, while other sentences showed little benefit. In some sentences, the transliteration model hypothesized incorrect transliterations for OOV words. More effectively filtering such incorrect translation options, such as through a more developed measure of confidence, is a potential avenue for future work. Tables 2 and 3 show examples of Joshua decoder output with and without the transliteration feature.

6. Conclusion

In this work, we created an XML format to markup the output of specialized subtask modules and integrate alternate translations into the SMT decoder. We created a transliteration module using a character-based statistical MT system and several thousand pairs of transliterated words. The results are promising. In particular, a qualitative analysis suggests that the transliterations were able to appropriately compete with the phrase-based translation output. This work has also opened the door to integrating additional specialized translation modules. Such modules have a potential to increase translation performance, particularly in low-resource conditions.[9]

[9]This research was supported in part by the European Commission through the EuroMatrixPlus project, by the US National Science Foundation under grant IIS-0713448, and by the Human Language Technology Center of Excellence. Any opinions, findings, and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect the views of the sponsor.

Without Transliteration	**[UNKNOWN]** Members said that the economic plan, expensive, and will not be effective.
With Transliteration	**Republican** members said that the economic plan, expensive, and will not be effective.
Reference	The republican members said that this economic plan is very "expensive" and will not be effective.
Without Transliteration	However, **[UNKNOWN]** said that he **[UNKNOWN]** president to respect their age and are also due to yell at them, but they were saying truth from miles away.
With Transliteration	"However, **Erdogan** said that he respects the **Israeli** President and his age as a result of which they yell at them , but they were saying the truth from miles away.
Reference	However, later **Erdogan** said that he respects **Israeli** President and his age as well which is why he did not yell at him but whatever he was saying was miles away from truth.

Table 2. Examples of improvements from transliteration.

Without Transliteration	In southern germany **[UNKNOWN]** a resident of the area of **[UNKNOWN] [UNKNOWN]** has left behind a big business group
With Transliteration	A resident of the area of **Cuba** in south Germany **Adolf Merkel** has left behind a big business group
Reference	**Adolf Merckle** of southern Germany's **Swabia** area has left a large business group behind

Table 3. Impact of transliteration. Note that the location name "Swabia" was incorrectly transliterated to "Cuba." This example indicates the future room for improvement.

Bibliography

Chiang, David. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, 2005.

Chiang, David. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

Dugast, Loïc, Jean Senellart, and Philipp Koehn. Statistical post-editing on systran's rule-based translation system. In *Proceedings of the Workshop on Statistical Machine Translation, part of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*, 2007.

Finkel, Jenny Rose and Christopher D. Manning. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, 2009.

Knight, Kevin and Jonathan Graehl. Machine transliteration. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1997)*, 1997.

Koehn, Philipp. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2004)*, 2004.

Koehn, Philipp and Kevin Knight. Feature-rich statistical translation of noun phrases. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, 2004.

Li, Zhifei, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Workshop on Statistical Machine Translation, part of the Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2009)*, 2009a.

Li, Zhifei, Chris Callison-Burch, Sanjeev Khudanpur, and Wren Thornton. Decoding in joshua: Open source, parsing-based machine translation. In *Prague Bulletin of Mathematical Linguistics*, number 91, 2009b.

Senellart, Jean, Christian Boitet, and Laurent Romary. XML machine translation. In *Proceedings of the 9th Machine Translation Summit*, 2003.

Yang, Mei and Katrin Kirchhoff. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, 2006.

Zollmann, Andreas, Ashish Venugopal, Franz Och, and Joy Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, 2008.