An Algorithm for Unsupervised Transliteration Mining with an Application to Word Alignment

Hassan Sajjad **Alexander Fraser** **Helmut Schmid**
Institute for Natural Language Processing
University of Stuttgart
{sajjad,fraser,schmid}@ims.uni-stuttgart.de

Abstract

We propose a language-independent method for the automatic extraction of transliteration pairs from parallel corpora. In contrast to previous work, our method uses no form of supervision, and does not require linguistically informed preprocessing. We conduct experiments on data sets from the NEWS 2010 shared task on transliteration mining and achieve an F-measure of up to 92%, outperforming most of the semi-supervised systems that were submitted. We also apply our method to English/Hindi and English/Arabic parallel corpora and compare the results with manually built gold standards which mark transliterated word pairs. Finally, we integrate the transliteration module into the GIZA++ word aligner and evaluate it on two word alignment tasks achieving improvements in both precision and recall measured against gold standard word alignments.

1 Introduction

Most previous methods for building transliteration systems were supervised, requiring either handcrafted rules or a clean list of transliteration pairs, both of which are expensive to create. Such resources are also not applicable to other language pairs.

In this paper, we show that it is possible to extract transliteration pairs from a parallel corpus using an unsupervised method. We first align a bilingual corpus at the word level using GIZA++ and create a list of word pairs containing a mix of non-transliterations and transliterations. We train a statistical transliterator on the list of word pairs. We then filter out a few word pairs (those which have the lowest transliteration probabilities according to the trained transliteration system) which are likely to be non-transliterations. We retrain the transliterator on the filtered data set. This process is iterated, filtering out more and more non-transliteration pairs until a nearly clean list of transliteration word pairs is left. The optimal number of iterations is automatically determined by a novel stopping criterion.

We compare our unsupervised transliteration mining method with the semi-supervised systems presented at the NEWS 2010 shared task on transliteration mining (Kumaran et al., 2010) using four language pairs. We refer to this task as NEWS10. These systems used a manually labelled set of data for initial supervised training, which means that they are semi-supervised systems. In contrast, our system is fully unsupervised. We achieve an F-measure of up to 92% outperforming most of the semi-supervised systems.

The NEWS10 data sets are extracted Wikipedia InterLanguage Links (WIL) which consist of parallel phrases, whereas a parallel corpus consists of parallel sentences. Transliteration mining on the WIL data sets is easier due to a higher percentage of transliterations than in parallel corpora. We also do experiments on parallel corpora for two language pairs. To this end, we created gold standards in which sampled word pairs are annotated as either transliterations or non-transliterations. These gold standards have been submitted with the paper as supplementary material as they are available to the research community.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 430–439,
Portland, Oregon, June 19-24, 2011.

Finally we integrate a transliteration module into the GIZA++ word aligner and show that it improves word alignment quality. The transliteration module is trained on the transliteration pairs which our mining method extracts from the parallel corpora. We evaluate our word alignment system on two language pairs using gold standard word alignments and achieve improvements of 10% and 13.5% in precision and 3.5% and 13.5% in recall.

The rest of the paper is organized as follows. In section 2, we describe the filtering model and the transliteration model. In section 3, we present our iterative transliteration mining algorithm and an algorithm which computes a stopping criterion for the mining algorithm. Section 4 describes the evaluation of our mining method through both gold standard evaluation and through using it to improve word alignment quality. In section 5, we present previous work and we conclude in section 6.

2 Models

Our algorithms use two different models. The first model is a joint character sequence model which we apply to transliteration mining. We use the grapheme-to-phoneme converter g2p to implement this model. The other model is a standard phrase-based MT model which we apply to transliteration (as opposed to transliteration mining). We build it using the Moses toolkit.

2.1 Joint Sequence Model Using g2p

Here, we briefly describe g2p using notation from Bisani and Ney (2008). The details of the model, its parameters and the utilized smoothing techniques can be found in Bisani and Ney (2008).

The training data is a list of word pairs (a source word and its presumed transliteration) extracted from a word-aligned parallel corpus. g2p builds a joint sequence model on the character sequences of the word pairs and infers m-to-n alignments between source and target characters with Expectation Maximization (EM) training. The m-to-n character alignment units are referred to as "multigrams".

The model built on multigrams consisting of source and target character sequences greater than one learns too much noise (non-transliteration information) from the training data and performs poorly. In our experiments, we use multigrams with a maximum of one character on the source and one character on the target side (i.e., 0,1-to-0,1 character alignment units).

The N-gram approximation of the joint probability can be defined in terms of multigrams q_i as:

$$p(q_1^k) \approx \prod_{j=1}^{k+1} p(q_j | q_{j-N+1}^{j-1}) \quad (1)$$

where q_0, q_{k+1} are set to a special boundary symbol.

N-gram models of order > 1 did not work well because these models tended to learn noise (information from non-transliteration pairs) in the training data. For our experiments, we only trained g2p with the unigram model.

In test mode, we look for the best sequence of multigrams given a fixed source and target string and return the probability of this sequence.

For the mining process, we trained g2p on lists containing both transliteration pairs and non-transliteration pairs.

2.2 Statistical Machine Transliteration System

We build a phrase-based MT system for transliteration using the Moses toolkit (Koehn et al., 2003). We also tried using g2p for implementing the transliteration decoder but found Moses to perform better. Moses has the advantage of using Minimum Error Rate Training (MERT) which optimizes transliteration accuracy rather than the likelihood of the training data as g2p does. The training data contains more non-transliteration pairs than transliteration pairs. We don't want to maximize the likelihood of the non-transliteration pairs. Instead we want to optimize the transliteration performance for test data. Secondly, it is easy to use a large language model (LM) with Moses. We build the LM on the target word types in the data to be filtered.

For training Moses as a transliteration system, we treat each word pair as if it were a parallel sentence, by putting spaces between the characters of each word. The model is built with the default settings of the Moses toolkit. The distortion limit "d" is set to zero (no reordering). The LM is implemented as a five-gram model using the SRILM-Toolkit (Stolcke, 2002), with Add-1 smoothing for unigrams and Kneser-Ney smoothing for higher n-grams.

3 Extraction of Transliteration Pairs

Training of a supervised transliteration system requires a list of transliteration pairs which is expensive to create. Such lists are usually either built manually or extracted using a classifier trained on manually labelled data and using other language dependent information. In this section, we present an iterative method for the extraction of transliteration pairs from parallel corpora which is fully unsupervised and language pair independent.

Initially, we extract a list of word pairs from a word-aligned parallel corpus using GIZA++. The extracted word pairs are either transliterations, other kinds of translations, or misalignments. In each iteration, we first train g2p on the list of word pairs. Then we delete those 5% of the (remaining) training data which are least likely to be transliterations according to g2p.[1] We determine the best iteration according to our stopping criterion and return the filtered data set from this iteration. The stopping criterion uses unlabelled held-out data to predict the optimal stopping point. The following sections describe the transliteration mining method in detail.

3.1 Methodology

We will first describe the iterative filtering algorithm (Algorithm 1) and then the algorithm for the stopping criterion (Algorithm 2). In practice, we first run Algorithm 2 for 100 iterations to determine the best number of iterations. Then, we run Algorithm 1 for that many iterations.

Initially, the parallel corpus is word-aligned using GIZA++ (Och and Ney, 2003), and the alignments are refined using the grow-diag-final-and heuristic (Koehn et al., 2003). We extract all word pairs which occur as 1-to-1 alignments in the word-aligned corpus. We ignore non-1-to-1 alignments because they are less likely to be transliterations for most language pairs. The extracted set of word pairs will be called "*list of word pairs*" later on. We use the list of word pairs as the training data for Algorithm 1.

Algorithm 1 builds a joint sequence model using g2p on the training data and computes the joint probability of all word pairs according to g2p. We normalize the probabilities by taking the nth square root where n is the average length of the source and the target string. The training data contains mostly non-transliteration pairs and a few transliteration pairs. Therefore the training data is initially very noisy and the joint sequence model is not very accurate. However it can successfully be used to eliminate a few word pairs which are very unlikely to be transliterations.

Algorithm 1 Mining of transliteration pairs

1: training data ←list of word pairs
2: I ← 0
3: **repeat**
4: Build a joint source channel model on the training data using g2p and compute the joint probability of every word pair.
5: Remove the 5% word pairs with the lowest length-normalized probability from the training data. {and repeat the process with the filtered training data}
6: I ← I+1
7: **until** I = Stopping iteration from Algorithm 2

On the filtered training data, we can train a model which is slightly better than the previous model. Using this improved model, we can eliminate further non-transliterations.

Our results show that at the iteration determined by our stopping criterion, the filtered set mostly contains transliterations and only a small number of transliterations have been mistakenly eliminated (see section 4.2).

Algorithm 2 automatically determines the best stopping point of the iterative transliteration mining process. It is an extension of Algorithm 1. It runs the iterative process of Algorithm 1 on half of the list of word pairs (training data) for 100 iterations. For every iteration, it builds a transliteration system on the filtered data. The transliteration system is tested on the source side of the other half of the list of word pairs (held-out). The output of the transliteration system is matched against the target side of the held-out data. (These target words are either transliterations, translations or misalignments.) We match the target side of the held-out data under the assumption that all matches are transliterations. The iteration where the output of the transliteration system best matches the held-out data is chosen as the stopping iteration of Algorithm 1.

[1] Since we delete 5% from the filtered data, the number of deleted data items decreases in each iteration.

Algorithm 2 Selection of the stopping iteration for the transliteration mining algorithm

1: Create clusters of word pairs from the list of word pairs which have a common prefix of length 2 both on the source and target language side.
2: Randomly add each cluster either to the training data or to the held-out data.
3: I $\leftarrow$ 0
4: **while** I $<$ 100 **do**
5: Build a joint sequence model on the training data using g2p and compute the length-normalized joint probability of every word pair in the training data.
6: Remove the 5% word pairs with the lowest probability from the training data. {The training data will be reduced by 5% of the rest in each iteration}
7: Build a transliteration system on the filtered training data and test it using the source side of the held-out and match the output against the target side of the held-out.
8: I $\leftarrow$ I+1
9: **end while**
10: Collect statistics of the matching results and take the median from 9 consecutive iterations (median9).
11: Choose the iteration with the best median9 score for the transliteration mining process.

We will now describe Algorithm 2 in detail. Algorithm 2 initially splits the word pairs into training and held-out data. This could be done randomly, but it turns out that this does not work well for some tasks. The reason is that the parallel corpus contains inflectional variants of the same word. If two variants are distributed over training and held-out data, then the one in the training data may cause the transliteration system to produce a correct translation (but not transliteration) of its variant in the held-out data. This problem is further discussed in section 4.2.2. Instead of randomly splitting the data, we first create clusters of word pairs which have a common prefix of length 2 both on the source and target language side. We randomly add each cluster either to the training data or to the held-out data.

We repeat the mining process (described in Algorithm 1) to eliminate non-transliteration pairs from the training data. For each iteration of Algorithm 2, i.e., steps 4 to 9, we build a transliteration system on the filtered training data and test it on the source side of the held-out. We collect statistics on how well the output of the system matches the target side of the held-out. The matching scores on the held-out data often make large jumps from iteration to iteration. We take the median of the results from 9 consecutive iterations (the 4 iterations before, the current and the 4 iterations after the current iteration) to smooth the scores. We call this median9. We choose the iteration with the best smoothed score as the stopping point for the filtering process. In our tests, the median9 heuristic indicated an iteration close to the optimal iteration.

Sometimes several nearby iterations have the same maximal smoothed score. In that case, we choose the one with the highest unsmoothed score. Section 4.2 explains the median9 heuristic in more detail and presents experimental results showing that it works well.

4 Experiments

We evaluate our transliteration mining algorithm on three tasks: transliteration mining from Wikipedia InterLanguage Links, transliteration mining from parallel corpora, and word alignment using a word aligner with a transliteration component. On the WIL data sets, we compare our fully unsupervised system with the semi-supervised systems presented at the NEWS10 (Kumaran et al., 2010). In the evaluation on parallel corpora, we compare our mining results with a manually built gold standard in which each word pair is either marked as a transliteration or as a non-transliteration. In the word alignment experiment, we integrate a transliteration module which is trained on the transliterations pairs extracted by our method into a word aligner and show a significant improvement. The following sections describe the experiments in detail.

4.1 Experiments Using Parallel Phrases of Wikipedia InterLanguage Links

We conduct transliteration mining experiments on the English/Arabic, English/Hindi, English/Tamil and English/Russian Wikipedia InterLanguage Links (WIL) used in the NEWS10.[2] All data sets

[2]We do not evaluate on the English/Chinese data because the Chinese data requires word segmentation which is beyond the scope of our work. Another problem is that our extraction method was developed for alphabetic languages and probably needs to be adapted before it is applicable to logographic languages such as Chinese.

	Our	S-Best	S-Worst	Systems	Rank
EA	87.4	91.5	70.2	16	3
ET	90.1	91.4	57.5	14	3
EH	92.2	94.4	71.4	14	3

Table 1: Summary of results on NEWS10 data sets where "EA" is English/Arabic, "ET" is English/Tamil and "EH" is English/Hindi. "Our" shows the F-measure of our filtered data against the gold standard using the supplied evaluation tool, "Systems" is the total number of participants in the subtask, and "Rank" is the rank we would have obtained if our system had participated.

contain training data, seed data and reference data. We make no use of the seed data since our system is fully unsupervised. We calculate the F-measure of our filtered transliteration pairs against the supplied gold standard using the supplied evaluation tool.

For English/Arabic, English/Hindi and English/Tamil, our system is better than most of the semi-supervised systems presented at the NEWS 2010 shared task for transliteration mining. Table 1 summarizes the F-scores on these data sets.

On the English/Russian data set, our system achieves 76% F-measure which is not good compared with the systems that participated in the shared task. The English/Russian corpus contains many cognates which – according to the NEWS10 definition – are not transliterations of each other. Our system learns the cognates in the training data and extracts them as transliterations (see Table 2).

The two best teams on the English/Russian task presented various extraction methods (Jiampojamarn et al., 2010; Darwish, 2010). Their systems behave differently on English/Russian than on other language pairs. Their best systems for English/Russian are only trained on the seed data and the use of unlabelled data does not help the performance. Since our system is fully unsupervised, and the unlabelled data is not useful, we perform badly.

4.2 Experiments Using Parallel Corpora

The Wikipedia InterLanguage Links shared task data contains a much larger proportion of transliterations than a parallel corpus. In order to examine how well our method performs on parallel corpora, we apply it to parallel corpora of English/Hindi and English/Arabic, and compare the transliteration mining results with a gold standard.

English	Russian	English	Russian
Studio	Студия/Studiya	Catalonia	Каталонии/Katalonii
Estonia	Эстонии/Estonii	Monastery	Монастырь/Monastyr
Tartary	Тартария/Tartariya	Geography	География/Geografiya

Table 2: Cognates from English/Russian corpus extracted by our system as transliteration pairs. None of them are correct transliteration pairs according to the gold standard.

We use the English/Hindi corpus from the shared task on word alignment, organized as part of the ACL 2005 Workshop on Building and Using Parallel Texts (WA05) (Martin et al., 2005). For English/Arabic, we use a freely available parallel corpus from the United Nations (UN) (Eisele and Chen, 2010). We randomly take 200,000 parallel sentences from the UN corpus of the year 2000. We create gold standards for both language pairs by randomly selecting a few thousand word pairs from the lists of word pairs extracted from the two corpora. We manually tag them as either transliterations or non-transliterations. The English/Hindi gold standard contains 180 transliteration pairs and 2084 non-transliteration pairs and the English/Arabic gold standard contains 288 transliteration pairs and 6639 non-transliteration pairs. We have submitted these gold standards with the paper. They are available to the research community.

In the following sections, we describe the median9 heuristic and the splitting method of Algorithm 2. The splitting method is used to avoid early peaks in the held-out statistics, and the median9 heuristic smooths the held-out statistics in order to obtain a single peak.[3]

4.2.1 Motivation for Median9 Heuristic

Algorithm 2 collects statistics from the held-out data (step 10) and selects the stopping iteration. Due to the noise in the held-out data, the transliteration accuracy on the held-out data often jumps from iteration to iteration. The dotted line in figure 1 (right) shows the held-out prediction accuracy for the En-

[3]We do not use the seed data in our system. However, to check the correctness of the stopping point, we tested the transliteration system on the seed data (available with NEWS10) for every iteration of Algorithm 2. We verified that the median9 held-out statistics and accuracy on the seed data have their peaks at the same iteration.

glish/Hindi parallel corpus. The curve is very noisy and has two peaks. It is difficult to see the effect of the filtering. We take the median of the results from 9 consecutive iterations to smooth the scores. The solid line in figure 1 (right) shows a smoothed curve built using the median9 held-out scores. A comparison with the gold standard (section 4.2.3) shows that the stopping point (peak) reached using the median9 heuristic is better than the stopping point obtained with unsmoothed scores.

4.2.2 Motivation for Splitting Method

Algorithm 2 initially splits the list of word pairs into training and held-out data. A random split worked well for the WIL data, but failed on the parallel corpora. The reason is that parallel corpora contain inflectional variants of the same word. If these variants are randomly distributed over training and held-out data, then a non-transliteration word pair such as the English-Hindi pair "change – badlao" may end up in the training data and the related pair "changes – badlao" in the held-out data. The Moses system used for transliteration will learn to "transliterate" (or actually translate) "change" to "badlao". From other examples, it will learn that a final "s" can be dropped. As a consequence, the Moses transliterator may produce the non-transliteration "badlao" for the English word "changes" in the held-out data. Such matching predictions of the transliterator which are actually translations lead to an overestimate of the transliteration accuracy and may cause Algorithm 2 to predict a stopping iteration which is too early.

By splitting the list of word pairs in such a way that inflectional variants of a word are placed either in the training data, or in the held-out, but not in both, this problem can be solved.[4]

The left graph in Figure 1 shows that the median9 held-out statistics obtained after a random data split of a Hindi/English corpus contains two peaks which occur too early. These peaks disappear in the right graph of Figure 1 which shows the results obtained after a split with the clustering method.

The overall trend of the smoothed curve in figure 1 (right) is very clear. We start by filtering out non-transliteration pairs from the data, so the results

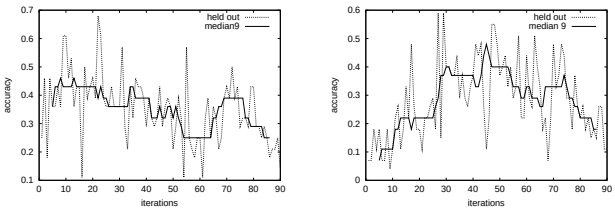

Figure 1: Statistics of held-out prediction of English/Hindi data using modified Algorithm 2 with random division of the list of word pairs (left) and using Algorithm 2 (right). The dotted line shows unsmoothed held-out scores and solid line shows median9 held-out scores

of the transliteration system go up. When no more non-transliteration pairs are left, we start filtering out transliteration pairs and the results of the system go down. We use this stopping criterion for all language pairs and achieve consistently good results.

4.2.3 Results on Parallel Corpora

According to the gold standard, the English/Hindi and English/Arabic data sets contain 8% and 4% transliteration pairs respectively. We repeat the same mining procedure – run Algorithm 2 up to 100 iterations and return the stopping iteration. Then, we run Algorithm 1 up to the stopping iteration returned by Algorithm 2 and obtain the filtered data.

	TP	FN	TN	FP
EH Filtered	170	10	2039	45
EA Filtered	197	91	6580	59

Table 3: Transliteration mining results using the parallel corpus of English/Hindi (EH) and English/Arabic (EA) against the gold standard

Table 3 shows the mining results on the English/Hindi and English/Arabic corpora. The gold standard is a subset of the data sets. The English/Hindi gold standard contains 180 transliteration pairs and 2084 non-transliteration pairs. The English/Arabic gold standard contains 288 transliteration pairs and 6639 non-transliteration pairs. From the English/Hindi data, the mining system has mined 170 transliteration pairs out of 180 transliteration pairs. The English/Arabic mined data contains 197 transliteration pairs out of 288 transliteration pairs. The mining system has wrongly identified a few non-transliteration pairs as transliterations (see

[4]This solution is appropriate for all of the language pairs used in our experiments, but should be revisited if there is inflection realized as prefixes, etc.

table 3, last column). Most of these word pairs are close transliterations and differ by only one or two characters from perfect transliteration pairs. The close transliteration pairs provide many valid multigrams which may be helpful for the mining system.

4.3 Integration into Word Alignment Model

In the previous section, we presented a method for the extraction of transliteration pairs from a parallel corpus. In this section, we will explain how to build a transliteration module on the extracted transliteration pairs and how to integrate it into MGIZA++ (Gao and Vogel, 2008) by interpolating it with the t-table probabilities of the IBM models and the HMM model. MGIZA++ is an extension of GIZA++. It has the ability to resume training from any model rather than starting with Model1.

4.3.1 Modified EM Training of the Word Alignment Models

GIZA++ applies the IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996) in both directions, i.e., source to target and target to source. The alignments are refined using the grow-diag-final-and heuristic (Koehn et al., 2003). GIZA++ generates a list of translation pairs with alignment probabilities, which is called the t-table. In this section, we propose a method to modify the translation probabilities of the t-table by interpolating the translation counts with transliteration counts. The interpolation is done in both directions. In the following, we will only consider the **e**-to-**f** direction. The transliteration module which is used to calculate the conditional transliteration probability is described in Algorithm 3.

We build a transliteration system by training Moses on the filtered transliteration corpus (using Algorithm 1) and apply it to the **e** side of the list of word pairs. For every source word, we generate the list of 10-best transliterations $nbestTI(e)$. Then, we extract every **f** that cooccurs with **e** in a parallel sentence and add it to $nbestTI(e)$ which gives us the list of candidate transliteration pairs $candidateTI(e)$. We use the sum of transliteration probabilities $\sum_{f' \in CandidateTI(e)} p_{moses}(f', e)$ as an approximation for the prior probability $p_{moses}(e) = \sum_{f'} p_{moses}(f', e)$ which is needed to convert the joint transliteration probability into a conditional probability. We use the constraint decoding option of Moses to compute the joint probability of **e** and **f**. It computes the probability by dividing the translation score of the best target sentence given a source sentence by the normalization factor.

Algorithm 3 Estimation of transliteration probabilities, **e**-to-**f** direction

```
unfiltered data ←list of word pairs
filtered data ←transliteration pairs extracted using Algorithm 1
Train a transliteration system on the filtered data
for all e do
    nbestTI(e) ← 10 best transliterations for e according to the transliteration system
    cooc(e) ← set of all f that cooccur with e in a parallel sentence
    candidateTI(e) ← cooc(e) ∪ nbestTI(e)
end for
for all f do
    p_moses(f, e) ← joint transliteration probability of e and f according to the transliterator
    p_ti(f|e) ← p_moses(f,e) / Σ_{f'∈CandidateTI(e)} p_moses(f',e)
end for
```

We combine the transliteration probabilities with the translation probabilities of the IBM models and the HMM model. The normal translation probability $p_{ta}(f|e)$ of the word alignment models is computed with relative frequency estimates.

We smooth the alignment frequencies by adding the transliteration probabilities weighted by the factor λ and get the following modified translation probabilities

$$\hat{p}(f|e) = \frac{f_{ta}(f, e) + \lambda p_{ti}(f|e)}{f_{ta}(e) + \lambda} \qquad (2)$$

where $f_{ta}(f, e) = p_{ta}(f|e)f(e)$. $p_{ta}(f|e)$ is obtained from the original t-table of the alignment model. $f(e)$ is the total corpus frequency of **e**. λ is the transliteration weight which is optimized for every language pair (see section 4.3.2). Apart from the definition of the weight λ, our smoothing method is equivalent to Witten-Bell smoothing.

We smooth after every iteration of the IBM models and the HMM model except the last iteration of each model. Algorithm 4 shows the smoothing for IBM Model4. IBM Model1 and the HMM model are smoothed in the same way. We also apply Algorithm 3 and Algorithm 4 in the alignment direction

Algorithm 4 Interpolation with the IBM Model4, **e**-to-**f** direction

1: {We want to run four iterations of Model4}
2: $f(e) \leftarrow$ total frequency of e in the corpus
3: Run MGIZA++ for one iteration of Model4
4: $I \leftarrow 1$
5: **while** $I < 4$ **do**
6: Look up $p_{ta}(f|e)$ in the t-table of Model4
7: $f_{ta}(f,e) \leftarrow p_{ta}(f|e)f(e)$ for all (f,e)
8: $\hat{p}(f|e) \leftarrow \frac{f_{ta}(f,e)+\lambda p_{ti}(f|e)}{f_{ta}(e)+\lambda}$ for all (f,e)
9: Resume MGIZA++ training for 1 iteration using the modified t-table probabilities $\hat{p}(f|e)$
10: $I \leftarrow I + 1$
11: **end while**

f to **e**. The final alignments are generated using the grow-diag-final-and heuristic (Koehn et al., 2003).

4.3.2 Evaluation

The English/Hindi corpus available from WA05 consists of training, development and test data. As development and test data for English/Arabic, we use manually created gold standard word alignments for 155 sentences extracted from the Hansards corpus released by LDC. We use 50 sentences for development and 105 sentences for test.

Baseline: We align the data sets using GIZA++ (Och and Ney, 2003) and refine the alignments using the grow-diag-final-and heuristic (Koehn et al., 2003). We obtain the baseline F-measure by comparing the alignments of the test corpus with the gold standard alignments.

Experiments We use GIZA++ with 5 iterations of Model1, 4 iterations of HMM and 4 iterations of Model4. We interpolate translation and transliteration probabilities at different iterations (and different combinations of iterations) of the three models and always observe an improvement in alignment quality. For the final experiments, we interpolate at every iteration of the IBM models and the HMM model except the last iteration of every model where we could not interpolate for technical reasons.[5] Algorithm 4 shows the interpolation of the transliteration probabilities with IBM Model4. We used the same procedure with IBM Model1 and the HMM model.

The parameter λ is optimized on development data for every language pair. The word alignment system is not very sensitive to λ. Any λ in the range between 50 and 100 works fine for all language pairs. The optimization helps to maximize the improvement in word alignment quality. For our experiments, we use $\lambda = 80$.

On test data, we achieve an improvement of approximately 10% and 13.5% in precision and 3.5% and 13.5% in recall on English/Hindi and English/Arabic word alignment, respectively. Table 4 shows the scores of the baseline and our word alignment model.

Lang	P_b	R_b	F_b	P_{ti}	R_{ti}	F_{ti}
EH	49.1	48.5	51.2	59.1	52.1	55.4
EA	50.8	49.9	50.4	64.4	63.6	64

Table 4: Word alignment results on the test data of English/Hindi (EH) and English/Arabic (EA) where P_b is the precision of baseline GIZA++ and P_{ti} is the precision of our word alignment system

We compared our word alignment results with the systems presented at WA05. Three systems, one limited and two un-limited, participated in the English/Hindi task. We outperform the limited system and one un-limited system.

5 Previous Research

Previous work on transliteration mining uses a manually labelled set of training data to extract transliteration pairs from a parallel corpus or comparable corpora. The training data may contain a few hundred randomly selected transliteration pairs from a transliteration dictionary (Yoon et al., 2007; Sproat et al., 2006; Lee and Chang, 2003) or just a few carefully selected transliteration pairs (Sherif and Kondrak, 2007; Klementiev and Roth, 2006). Our work is more challenging as we extract transliteration pairs without using transliteration dictionaries or gold standard transliteration pairs.

Klementiev and Roth (2006) initialize their transliteration model with a list of 20 transliteration

[5] We had problems in resuming MGIZA++ training when training was supposed to continue from a different model, such as if we stopped after the 5th iteration of Model1 and then tried to resume MGIZA++ from the first iteration of the HMM model. In this case, we ran the 5th iteration of Model1, then the first iteration of the HMM and only then stopped for interpolation; so we did not interpolate in just those iterations of training where we were transitioning from one model to the next.

pairs. Their model makes use of temporal scoring to rank the candidate transliterations. A lot of work has been done on discovering and learning transliterations from comparable corpora by using temporal and phonetic information (Tao et al., 2006; Klementiev and Roth, 2006; Sproat et al., 2006). We do not have access to this information.

Sherif and Kondrak (2007) train a probabilistic transducer on 14 manually constructed transliteration pairs of English/Arabic. They iteratively extract transliteration pairs from the test data and add them to the training data. Our method is different from the method of Sherif and Kondrak (2007) as our method is fully unsupervised, and because in each iteration, they add the most probable transliteration pairs to the training data, while we filter out the least probable transliteration pairs from the training data.

The transliteration mining systems of the four NEWS10 participants are either based on discriminative or on generative methods. All systems use manually labelled (seed) data for the initial training. The system based on the edit distance method submitted by Jiampojamarn et al. (2010) performs best for the English/Russian task. Jiampojamarn et al. (2010) submitted another system based on a standard n-gram kernel which ranked first for the English/Hindi and English/Tamil tasks.[6] For the English/Arabic task, the transliteration mining system of Noeman and Madkour (2010) was best. They normalize the English and Arabic characters in the training data which increases the recall.[7]

Our transliteration extraction method differs in that we extract transliteration pairs from a parallel corpus without supervision. The results of the NEWS10 experiments (Kumaran et al., 2010) show that no single system performs well on all language pairs. Our unsupervised method seems robust as its performance is similar to or better than many of the semi-supervised systems on three language pairs.

We are only aware of one previous work which uses transliteration information for word alignment. Hermjakob (2009) proposed a linguistically focused word alignment system which uses many features including hand-crafted transliteration rules for Arabic/English alignment. His evaluation did not explicitly examine the effect of transliteration (alone) on word alignment. We show that the integration of a transliteration system based on unsupervised transliteration mining increases the word alignment quality for the two language pairs we tested.

[6] They use the seed data as positive examples. In order to obtain also negative examples, they generate all possible word pairs from the source and target words in the seed data and extract the ones which are not transliterations but have a common substring of some minimal length.

[7] They use the phrase table of Moses to build a mapping table between source and target characters. The mapping table is then used to construct a finite state transducer.

6 Conclusion

We proposed a method to automatically extract transliteration pairs from parallel corpora without supervision or linguistic knowledge. We evaluated it against the semi-supervised systems of NEWS10 and achieved high F-measure and performed better than most of the semi-supervised systems. We also evaluated our method on parallel corpora and achieved high F-measure. We integrated the transliteration extraction module into the GIZA++ word aligner and showed gains in alignment quality. We will release our transliteration mining system and word alignment system in the near future.

Acknowledgments

The authors wish to thank the anonymous reviewers for their comments. We would like to thank Christina Lioma for her valuable feedback on an earlier draft of this paper. Hassan Sajjad was funded by the Higher Education Commission (HEC) of Pakistan. Alexander Fraser was funded by Deutsche Forschungsgemeinschaft grant Models of Morphosyntax for Statistical Machine Translation. Helmut Schmid was supported by Deutsche Forschungsgemeinschaft grant SFB 732.

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5).

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kareem Darwish. 2010. Transliteration mining with phonetic conflation and iterative training. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from United Nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, Columbus, Ohio, June. Association for Computational Linguistics.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, Morristown, NJ, USA. Association for Computational Linguistics.

Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. 2010. Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Morristown, NJ, USA.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*, pages 127–133, Edmonton, Canada.

A Kumaran, Mitesh M. Khapra, and Haizhou Li. 2010. Whitepaper of news 2010 shared task on transliteration mining. In *Proceedings of the 2010 Named Entities Workshop the 48th Annual Meeting of the ACL*, Uppsala, Sweden.

Chun-Jen Lee and Jason S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts*, Morristown, NJ, USA. ACL.

Joel Martin, Rada Mihalcea, and Ted Pedersen. 2005. Word alignment for languages with scarce resources. In *ParaText '05: Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Morristown, NJ, USA. Association for Computational Linguistics.

Sara Noeman and Amgad Madkour. 2010. Language independent transliteration mining system using finite state automata framework. In *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden. Association for Computational Linguistics.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Tarek Sherif and Grzegorz Kondrak. 2007. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *ACL*, Prague, Czech Republic.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *ACL*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado.

Tao Tao, Su-Yoon Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *16th International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of the 45th Annual Meeting of the ACL*, Prague, Czech Republic.