Transliteration Mining Using Large Training and Test Sets

Ali El Kahki, Kareem Darwish, Ahmed Saad El Din
Qatar Computing Research Institute
Qatar Foundation, Doha, Qatar
ame2154@columbia.edu,
kdarwish@qf.org.qa, ataei@qf.org.qa

Mohamed Abd El-Wahab
Faculty of Computers and Information,
Cairo University, Cairo, Egypt
wahab@writeme.com

Abstract

Much previous work on Transliteration Mining (TM) was conducted on short parallel snippets using limited training data, and successful methods tended to favor recall. For such methods, increasing training data may impact precision and application on large comparable texts may impact precision and recall. We adapt a state-of-the-art TM technique with the best reported scores on the ACL 2010 NEWS workshop dataset, namely graph reinforcement, to work with large training sets. The method models observed character mappings between language pairs as a bipartite graph and unseen mappings are induced using random walks. Increasing training data yields more correct initial mappings but induced mappings become more error prone. We introduce parameterized exponential penalty to the formulation of graph reinforcement and we estimate the proper parameters for training sets of varying sizes. The new formulation led to sizable improvements in precision. Mining from large comparable texts leads to the presence of phonetically similar words in target and source texts that may not be transliterations or may adversely impact candidate ranking. To overcome this, we extracted related segments that have high translation overlap, and then we performed TM on them. Segment extraction produced significantly higher precision for three different TM methods.

1. Introduction

Transliteration Mining (TM) is the process of finding transliterations in parallel or comparable texts of different languages. For example, given the Arabic-English word sequence pairs: (الملك هالي سلاسي, Haile Selassie I of Ethiopia), successful TM would mine the transliterations: (هالي, Haile) and (سلاسي, Selassie). TM has been shown to be effective in several Information Retrieval (IR) and Natural Language Processing (NLP) applications. For example, in cross language IR, TM was used to handle out-of-vocabulary query words by mining transliterations between words in queries and top *n* retrieved documents and then using transliterations to expand queries (Udupa et al., 2009a). In Machine Translation (MT), TM can improve alignment at training time and help enrich phrase tables with named entities that may not appear in parallel training data. More broadly, TM is a character mapping problem. Having good character mapping models can be beneficial in a variety of applications such as learning stemming models, learning spelling transformations between similar languages, and finding variant spellings of names (Udupa and Kumar, 2010b).

TM has attracted interest in recent years with a dedicated evaluation in the ACL 2010 NEWS workshop. In that evaluation, TM was performed using limited training data, namely 1,000 parallel transliteration word-pairs, on short parallel text segments, namely cross-language Wikipedia titles which were typically a few words long. Since TM was performed on very short parallel segments, the chances that two phonetically similar words would appear within such a short text segment in one language were typically very low. Also, since TM training datasets were small, many valid mappings were not observed in training. For these two reasons, most of the successful techniques related to that evaluation have focused on improving recall, while hurting precision slightly. Some of these techniques involved the use of letter conflation based on a SOUNDEX like scheme (Darwish, 2010; Oh and Choi, 2006) and character n-gram similarity. The most successful technique on ACL-NEWS dataset, involved the use of graph reinforcement in which observed mappings between language pairs were modeled using a bipartite graph and unseen mappings were induced using random walks (El-Kahki et al., 2011).

In this paper, we focus on improving TM between Arabic and English in more realistic settings, compared to the NEWS workshop dataset. Specifically, we focus on the cases where:

1. Relatively large TM training sets, which are typical of production systems, are available. As we will show, using more training data in conjunction with recall-oriented techniques that perform well on small training sets can adversely hurt precision, leading to drops in F-measure. A more fundamental question is what constitutes “large” versus “small” training sets. Ideally, we want a unified solution for training sets of varying sizes.
2. TM is performed on large comparable texts which are ubiquitously available from different sources such as cross language news and Wikipedia articles. In this case, there are two phenomena that arise. First, there is an increased probability (compared to short texts) that words in the target and source texts may be phonetically similar, while not being transliterations of each other. One such example is the Arabic word “من”, which means “in” and is pronounced as “min” and the English word “men”. Such cases adversely affect precision. Second, given a source language word, there may be multiple target language words that are phonetically similar and TM may rank a wrong word higher than the correct one. For example, consider the Arabic word “جو”, which is pronounced as “joe” but is in fact the rendition of the Chinese name “Zhou”. If the English text has words such as “jaw”, “joe”, “jo”, “joy”, etc., one of them may rank higher than “Zhou”. Since only the top choice is considered, this phenomenon would hurt precision and recall.

We address these two situations by making the following two contributions:

1. Modifying the TM technique with the best reported results on the ACL 2010 NEWS workshop, namely graph reinforcement (El-Kahki et al., 2011) to handle training sets of arbitrary sizes by introducing parameterized exponential penalty to the mapping induction process. We show that we can effectively learn the parameters that tune the penalty for two different training sets of varying sizes. In doing so, we achieve better results for graph reinforcement with larger training sets.
2. For large comparable texts, we use contextual clues, namely translations of neighboring words, to constrain TM and to preserve precision. Specifically, we initially extract text segments that are “related” based on cross lingual lexical overlap, and then we perform TM on these segments. Though there have been some papers on extracting sub-sentence alignments from comparable text (Hewavitharana and Vogel, 2011; Munteanu and Marcu, 2006), extracting related (as opposed to parallel) text segments may be preferable because: 1) transliterations may not occur in parallel contexts; 2) using simple lexical overlap is efficient; and as we will show 3) simultaneous use of phonetic and contextual evidences may be sufficient to produce high TM precision. Alternate solutions focused on performing TM on extracted named entities only (Udupa et al., 2009b). Some drawbacks of such an approach are: 1) named entity recognition (NER) may not be available for many languages; and 2) NER has inherently low recall for languages such as Arabic where no discriminating features such as capitalization exist.

The remainder of the paper is organized as follows: Section 2 provides background on TM; Section 3 describes the basic TM system that is used in the paper; Section 4 describes graph reinforcements, shows how it fairs in the presence of a large training set, and introduces modifications to graph reinforcement to improve its effectiveness with such data; Section 5 introduces the use of contextual clues to improve TM and reports on its effectiveness; and Section 6 concludes the paper.

2. Background

Much work has been done on TM for different language pairs such as English-Chinese (Kuo et al., 2006; Kuo et al., 2007; Kuo et al., 2008; Jin et al. 2008;), English-Tamil (Saravanan and Kumaran, 2008; Udupa and Khapra, 2010), English-Korean (Oh and Isahara, 2006; Oh and Choi, 2006), English-Japanese (Qu et al., 2000; Brill et al., 2001; Oh and Isahara, 2006), English-Hindi (Fei et al., 2003; Mahesh and Sinha, 2009), and English-Russian (Klementiev and Roth, 2006). TM typically involves finding character mappings between two languages and using these mappings to ascertain if two words are transliterations or not.

2.1 Finding Character Mappings

To find character sequence mappings between two languages, the most common approach entails using automatic letter alignment of transliteration pairs. Automatic alignment can be performed using different algorithms such as EM (Kuo et al., 2008; Lee and Chang, 2003) or HMM-based alignment (Udupa et al., 2009a; Udupa et al., 2009b). Another method uses automatic speech recognition confusion tables to extract phonetically equivalent character sequences to discover monolingual and cross-lingual pronunciation variations (Kuo and Yang, 2005). Alternatively, letters can be mapped into a common character set using a predefined transliteration scheme (Darwish, 2010; Oh and Choi, 2006).

2.2 Transliteration Mining

For the problem of ascertaining if two words can be transliterations of each other, a common approach involves using a generative model that attempts to generate all possible transliterations of a source word, given the character mappings between two languages, and restricting the output to words in the target language (Fei et al., 2003; Lee and Chang, 2003, Udupa et al., 2009a). This is similar to the baseline approach that we used in this paper. Noeman and Madkour (2010) implemented this technique using a finite state automaton by generating all possible transliterations along with weighted edit distance and then filtered them using appropriate thresholds and target language words. El-Kahki et al. (2011) combined a generative model with so-called graph reinforcement, which is described in greater detail in Section 4. They reported the best TM results on the ACL 2010 NEWS workshop dataset for 4 different languages. Alternatively back-transliteration can be used to determine if one sequence could have been generated by successively mapping character sequences from one language into another (Brill et al., 2001; Bilac and Tanaka, 2005; Oh and Isahara, 2006).

Udupa and Khapra (2010) proposed a method in which transliteration candidates are mapped into a "low-dimensional common representation space". Then, the similarity between the resultant feature vectors for both candidates can be computed. A similar approach uses context sensitive hashing (Udupa and Kumar, 2010).

Jiampojamarn et al. (2010) used classification to determine if source and target language words were valid transliterations. They used a variety of features including edit distance between an English token and the Romanized versions of the foreign token, forward and backward transliteration probabilities, and character n-gram similarity. Udupa et al. (2009b) used a similar classification-based approach.

3. Baseline Transliteration Mining

3.1 Description of the Baseline System

We used a generative TM model that was trained on a set of transliteration pairs. We automatically aligned these pairs at character level using an HMM-based aligner akin to that of He (2007). Alignment produced mappings between characters from both languages with associated probabilities. We restricted individual source language character sequences to be 3 characters at most. We always treated English as the target language and Arabic as the source language.

Briefly, we produced all possible segmentations of a source word along with their associated mappings into the target language. Valid target sequences were retained and sorted by the product of the constituent mapping probabilities. The candidate with the highest probability was generated given that the product of the mapping probabilities was higher than a certain threshold. Otherwise, no candidate was chosen.

The search for transliterated pairs was implemented as a variant of depth-first search (Pearl, 1984), where states represented valid mappings between source and target substrings. At each step, the mapping with the best score was selected and expanded using the mappings learnt from alignment. This process ran until mapping combinations produced target word(s) from a source word or until all possible states were explored. The pseudo code in Figure 1 describes the details of the algorithm. The implementation was optimized via incremental left to right processing of source words, the use of a radix tree to prune invalid paths, and the use of a sorted priority queue to insure that the highest weighing candidate was at the top of the queue.

```
Input: Mappings, set of mappings from source fragment to a list of target fragments and mapping Probability .
Input: SourceWord ($F_i \in F_1^n$), Source language word
Input: TargetWords, radix tree containing all target language words ($E_1^m$)
Data Structures: DFS, Priority queue to store candidate transliterations pair ordered by their transliteration score –
   Each candidate transliteration tuple = (SourceFragment, TargetTransliteration, TransliterationScore).
StartSymbol = ("", "", 1.0);  DFS={StartSymbol}
While (DFS is not empty)
    SourceFragment= DFS.Top().SourceFragment
    TargetFragment= DFS.Top().TargetTransliteration
    FragmentScore =DFS.Top().TransliterationScore
    If (SourceWord == SourceFragment)
        If (FragmentScore > Threshold) Return (SourceWord, TargetTransliteration, FragmentScore)
        Else Return Null
    DFS.RemoveTop()
    For SubFragmentLength = 1 to 3
        SourceSubString = SubString( SourceWord, SourceFragment.Length , SubFragmentLength)
        Foreach mapping in Mappings[SourceSubString]
            If ((TargetFragment + mapping.TargetFragemnt) is a sub-string in TargetWords)
                DFS.Add(SourceFragment + SourceSubString, TargetFragment + mapping.TargetFragement,
                mapping.Score * FragmentScore)
    DFS.RemoveTop()
End While
Return Null
```

Figure 1: Pseudo code for transliteration mining

3.2 Thresholding

We used a threshold on the minimum acceptable transliteration score to filter out unreliable transliterations. Fixing a uniform threshold would have caused the model to filter out long transliterations. Thus, we tied the threshold to the length of transliterated words. We assumed a threshold *d* for single character mappings and the transliteration threshold for a target word of length *l* would be d^l. Since we did not have a validation set to estimate *d*, we created a synthetic validation set from the training set and then used cross-validation to estimate *d* as follows: we split the training data into 5 folds for cross validation; we modified each validation fold by adding 5 random words to each target word in the transliteration pair; then we performed TM with varying thresholds on the validation fold and computed F-measure; and we ascertained the threshold that led to the highest F-measure for each fold and then took the average threshold.

3.3 Linguistic Processing

For Arabic, we performed letter normalization of the different forms of alef, alef maqsoura and ya, and ta marbouta and ha. For English, we case-folded all letters and removed accents, umlaut, and similar diacritic like marks (ex. á, â, ä, à, ã, ā, ą).

4. Modifying Graph Reinforcement

4.1 Original Graph Reinforcement

To motivate graph reinforcement, consider the following example: if alignment produced the mappings (ط, ti), (ط, ta), (ت, ti), and (ت, t), then the mappings (ط, t) and (ت, ta) are likely valid – though not observed. These mappings can be induced by traversing the following paths: ط ➔ ti ➔ ت ➔ t and ت ➔ ti ➔ ط ➔ ta respectively.

In graph reinforcement, observed mappings were modeled as a bipartite graph with source (S) and target (T) character sequences and weighted with the learnt alignment probabilities (M). Thus the mapping between s ∈ S and t ∈ T was m(s,t).

Graph reinforcement was performed by traversing the graph from S ➔ T ➔ S ➔ T in order to deduce new mappings. Given a source sequence s'∈ S and a target sequence t' ∈ T, the deduced mapping weights were computed as follows:

$$m(t'|s') = 1 - \prod_{\forall s \in S, t \in T} \left(1 - m(t'|s)m(s|t)m(t|s')\right)$$

where the term $\left(m(t'|s)m(s|t)m(t|s')\right)$ is the score of the path between t' and s'. De Morgan's law was applied to aggregate different paths using an OR operator, which involved taking the negation of negations of all possible paths aggregated by an AND operator. Hence, the probability of an inferred mapping would be boosted if it was obtained from multiple paths.

Since some characters, mainly vowels, have a tendency to map to many other characters, link reweighting was applied after each iteration. Link reweighting had the effect of decreasing the weights of target character sequences that have many source character sequences mapping to them and hence reducing the effect of incorrectly inducing mappings. Link reweighting was performed as follows:

$$m'(s|t) = \frac{m(s|t)}{\sum_{s_i \in S} m(s_i|t)}$$

Where $s_i \in S$ is a source sequence that maps to *t*. This is akin to normalizing conditional probabilities.

4.2 Graph Reinforcement Results

We tested graph reinforcement using 10 iterations in 2 different settings, namely:

1. NEWS-1k: Using the ACL-NEWS workshop dataset. The dataset contained 1,000 parallel transliteration word pairs for training and 1,000 parallel Wikipedia titles for testing.
2. NEWS-10k: Using the test part of the ACL-NEWS dataset, while training with 10,000 manually curated parallel transliterations.

Table 1 reports on the results of the graph reinforcement results for the two setups. In the NEWS-1k setup, graph reinforcement generally had a positive effect on recall at the expense of precision. However, as we suspected, increasing the amount of training data (as in the NEWS-10k) led to more initial mappings from alignment, but with many erroneously induced mappings that adversely impacted precision. Though recall improved significantly, precision deteriorated significantly, leading to lower F-measure.

Table 1. Results for NEWS-1k and NEWS-10k

		Baseline	Reinforcement
NEWS-1k	P	0.988	0.977
	R	0.583	0.912
	F	0.733	0.943
NEWS-10k	P	0.917	0.689
	R	0.759	0.960
	F	0.787	0.802

4.3 Modifying Graph Reinforcement with Parameterized Exponential Penalty

To overcome the problem demonstrated in the NEWS-10k setup, we adjusted the graph reinforcement formula to give more confidence to mappings that were observed due to initial alignment and to successively penalize mappings that were induced in later graph reinforcement iterations. The adjustment was as follows:

$$m^i(t'|s') = 1 - (1 - m^{i-1}(t'|s')) \cdot \prod_{s \in S, t \in T} 1 - e^{-i\alpha}\, m^{i-1}(t'|s) m^{i-1}(s|t) m^{i-1}(t|s')$$

Where the parameter α adjusts how much we penalize induced mappings and *i* is the number of iterations. $m^i(t'|s')$ is the mapping score at iteration *i*. Basically, newly seen links at iteration *i* are penalized by $e^{-i\alpha}$. The equation is similar to the earlier reinforcement equation but with all paths except the original path $s' \rightarrow t'$ multiplied by exponential penalty $e^{-i\alpha}$. Since the ACL-NEWS dataset did not have a validation set to help us estimate α, we opted to use the approach we used earlier to estimate the proper thresholds, namely: we split the training data into 5 folds for cross validation; we modified each validation fold by adding 5 random words to each target word in the transliteration pair; and then we performed TM with varying values of α and with 10 graph reinforcement iterations on the validation fold and computed precision and recall. For the 10k training set, we opted to use a 90/10 training/validation split of the training data, where the validation part was modified in the same manner as the validation folds of the ACL-NEWS datasets. We varied the value of α between 0.0 and 1.0 with increments of 0.1 and with increments of 1 afterwards for values greater than 1. If two values of α yielded the same F-measure (up to 3 decimal places), we favored the larger α, favoring precision. Figures 2 and 3 plot the precision and recall respectively on the validation (-valid) and test (-test) sets for the 1,000 pair training set.

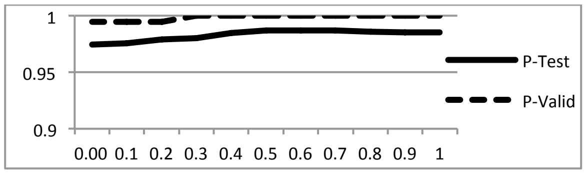

Figure 2. Precision (y-axis) on test and validation sets for varying values of α (x-axis) for the 1k set

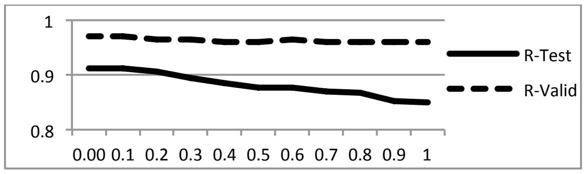

Figure 3. Recall (y-axis) on test and validation sets for varying values of α (x-axis) for the 1k set

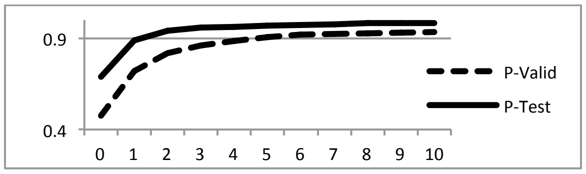

Figure 4. Precision (y-axis) on test and validation sets for varying values of α (x-axis) for the 10k set

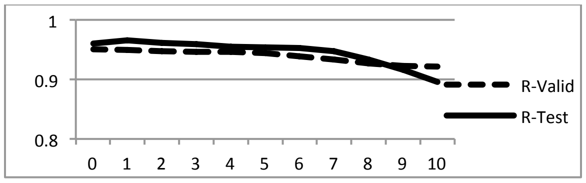

Figure 5. Recall (y-axis) on test and validation sets for varying values of α (x-axis) for the 10k set

Figures 4 and 5 plot the same for the 10k pair training set. The precision and recall values on the validation sets are indicative of their behavior on the test set. Due to the difference in training data sizes, the best values of α were significantly larger for the 10k dataset compared to the 1k dataset.

4.4 Modified Graph Reinforcement Results

We applied exponential penalty on graph reinforcement with the estimated value of α on the ACL-NEWS dataset as well as the 10k training set. Table 2 lists the estimated and optimal values of α for the different datasets on the training and test sets respectively along with the F-measure obtained for these values of α. Table 2 also compares the results to the results from baseline and graph reinforcement without exponential penalty. Tables 3 and 4 show precision, recall, and F-measure results for training using ACL-NEWS datasets and the larger training set respectively.

For the large dataset of 10k training words, using exponential penalty improved results noticeably, with a 16 basis points improvement in F-measure, and we were able to estimate the optimal α. For the smaller training set, using exponential penalty with the estimated α marginally changed overall results by (-0.006) compared to the optimal α. The change in overall F-measure was generally small, with most of the degradation in recall being offset by improvements in precision. The small error in estimating α for the ACL-NEWS dataset can be attributed to the small size of the validation set. Generally, smaller training sets require smaller values of α to allow reinforcement to deduce more unseen mappings, increasing recall. Larger training sets require larger values of α and exponential penalty becomes more important. The advantage of this formulation is that α can be learned to match training sets of varying sizes.

Table 2. F-measure for baseline, reinforcement, and exponential penalty at estimated and optimal α

	NEWS-1k	NEWS-10k
Baseline	0.757	0.787
Reinforcement (α=0)	0.941	0.802
Estimated α	0.3	6.0
@ Estimated α	0.935	0.963
Optimal α (on test)	0.1	6.0
@ optimal α	0.943	0.963

Table 3. Results for training using 1k training set

	P	R	F1
Baseline	0.975	0.619	0.757
Reinforcement (α=0)	0.975	0.912	0.941
@ estimated α	0.980	0.894	0.935

Table 4. Results for training using 10k training set

	P	R	F1
Baseline	0.917	0.759	0.787
Reinforcement (α=0)	0.689	0.960	0.802
@ estimated α	0.976	0.948	0.963

5. TM from Large Comparable Text

5.1 Baseline TM to Large Comparable Text

We tested TM using the 1,000 training pairs from the ACL-NEWS workshop on the longest 30 English Wikipedia articles with equivalent Arabic Wikipedia articles. The test articles had the following properties:

	Max. Len	Min. Len	Avg. Len
Arabic	10,165	1,837	3,614
English	10,710	3,133	4,896

The article pairs had 64.7 transliterations on average (with 1,942 in total).

To show the generality of using contextual clues, we tested TM using 3 different techniques, namely: the aforementioned baseline system, graph reinforcement, and using SOUNDEX-like letter conflation for English in the manner suggested by Darwish (2010). This letter conflation involved removing vowels, “H”, and ‘W”; and performing the following mappings:

B, F, P, V $\rightarrow$ 1	C, G, J, K, Q, S, X, Z $\rightarrow$ 2
D,T $\rightarrow$ 3	L $\rightarrow$ 4
M,N $\rightarrow$ 5	R $\rightarrow$ 6

Such letter conflation was shown to improve TM F-measure on the ACL-NEWS workshop from 0.73 to 0.85 (Darwish, 2010).

Table 5. Results for TM on full Wikipedia articles

	Baseline	SOUNDEX	Reinforcement
P	0.610	0.059	0.650
R	0.415	0.402	0.500
F	0.494	0.103	0.565

Table 5 reports the TM results on the Wikipedia articles. The increased size of the comparable text on which we were performing TM led to adverse effects on precision and recall for the baseline, graph reinforcement, and SOUNDEX setups – with 0.059 precision for SOUNDEX. Graph reinforcement performed slightly better than the baseline both in terms of precision and recall, but with such low precision values, TM may not be useful for many applications. As highlighted earlier, the reason behind the drop in precision was due to phonetically similar words that were in fact not transliterations. The reason behind the drop in recall was due to the following: when TM is performed, often the correct transliteration was found but not as the first candidate. Given that for evaluation we were considering the first candidate only, this hurt both precision and recall.

5.2 Using Context to Improve TM

To overcome the precision and recall problems, we used contextual information to improve TM for large comparable text. To do so, we filtered articles to extract potentially related fragments and then we applied TM on the extracted fragments. The filtering was performed based on lexical similarity between fragments. The idea was that words that do not share enough contexts were not likely to be transliterations. A byproduct of this approach was a significant reduction in TM running time since the search space was reduced. On the downside, this likely hurt recall as transliterations that do not share similar contexts could not be mined.

To extract fragments with similar context we used a phrase table from a phrase-based MT system, which was akin to Moses (Koehn et al., 2007), to detect similarity between fragments in articles. The MT system was trained using 14 million parallel Arabic-English sentence pairs. The extraction algorithm aimed to extract maximum length fragments that share contexts greater than a specific percentage of fragment lengths. The threshold that we used in our experimentation was 30%. When picking the threshold, our goal was to find transliterations that appear in similar and not necessarily identical contexts. The threshold was determined qualitatively on a validation set.

A brute force fragment extraction approach would extract all possible fragments in source and target articles, iterate on each word in each pair of fragments to find the mappings, and then include a fragment if the mappings count exceed the threshold. Such a brute force approach would have an order of N^3M^3, where N and M are the number of words in the source and target articles respectively. To improve the running time, we first removed stop words from the source list. Then, we created a list that contained the positions of each of the matching pairs in source and target articles sorted by source words’ position. This operation had a complexity of O(N*log*M). Next, we iterated on source fragments of different size, which was O(N^2), and added the positions of matches in the target article in a sorted list. This operation was O(K*log*K) where K is the number of matches. Then, we iterated on extracted matches to find target fragment that satisfied the condition:

$$\frac{\text{Fragment Length}}{\text{number of mappings}} \geq .3$$

The last step was O(K) in the worst case. The total complexity of this algorithm was O(N^2K*log*K) in the worst case, which had a much lower complexity than the brute force approach. In practice, the algorithm filtered 30 comparable pairs of articles with an average of 4.9k words for English and 3.6k for Arabic in less than 5 minutes. Details of the algorithm are shown in Figure 2.

5.3 Testing TM on Extracted Segments

Table 6 reports TM results on the extracted segments. As the results show, TM on extracted segments dramatically improved precision for all setups compared to TM on the full articles (as in Table 6). Except for the SOUNDEX setup, recall dropped by 9.3 and 8.3 basis points for the baseline and graph reinforcement setups respectively. Though F-measure dropped slightly for the baseline case and improved slightly for the reinforcement case, what is noteworthy is that

```
1:  Input:  Matches, a list of matches between word position in source article and its mapping in the target article sorted by
      source position
2:  Input:  Source, list of source words; Target, list of target words
4:  Output: ParallelFragments : List of pairs of parallel fragments
5:        For startPosition=0 To Source.Lenght
6:              For endPosition = startPosition + MinimumFragmentLengh To Source.Lenght
7:                    SortedList TargetMatches =[ ]
8:              ForEach match Between startPosistion And endPosition In Matches
9:                    TargetMatches.Add(Matching[match].targetPosition)
10:             startItr=0;  endItr=TargetMatches.Length - 1
12:             For i=0 to TargetMatches.Length
13:                   If( (endItr-startItr+1)/ (TargetMatches [endItr]-TargetMatches[startItr]) >.3) Then
14:                         ParallelFragments.Add(Source.GetRange(startPosition,
                            endPosition),Target.GetRange(TargetMatches[startItr], TargetMatches[endItr]))
15:                   Break
16:                   Else
17:                      If(TargetMatches[endItr]-TargetMatches[startItr+1]>TargetMatches[endItr-1]-
                         TargetMatches[startItr]) Then startItr++
19:                      Else endItr++
21:                      End If
22:                End If
23:             End Loop
24:          End Loop
25:       End Loop
26:       Return ParallelFragments
```

Figure 2. Pseudo code for the fragment extraction algorithm

precision was high enough to make TM practically useful for a variety of applications. The major advantage of the proposed technique is the achievement of relatively high precision – comparable to precision on small text snippets. Though recall is relatively low, the ubiquity of comparable texts can help produce large mined transliterations of high quality.

Table 6. Results for TM on extracted segments

	Baseline	SOUNDEX	Reinforcement
P	0.962	0.524	0.946
R	0.322	0.418	0.417
F	0.482	0.465	0.579

6. Conclusion

In this paper, we explored the use of transliteration mining in the context of using large training and test sets. Since recent work was conducted on small parallel text segments that were just a few words long with limited training data, the state-of-the-art techniques generally favored recall by inducing mappings that were unseen in training. Since the parallel test segments were short, improvements in recall had a very small effect on precision. When we applied the best reported method in the literature using large training data or when performing TM on large comparable texts, drops in precision and recall were substantial.

We modified the formulation of graph reinforcement by introducing a parameterized exponential penalty to allow for the discovery of new letter mappings using graph walks while penalizing mappings that required more graph walk steps to be induced. We showed how to effectively estimate the exponential penalty parameter for training sets of different sizes. In the context of performing TM on short parallel segments using 10k training words, we improved TM precision from 0.689 to 0.976 at the expense of a small drop in recall from 0.960 to 0.948.

What we observed for graph reinforcement is symptomatic of algorithms that may fail when more data is present. Other such examples include stemming for MT and IR. Generally, with more MT parallel data or bigger IR collections, stemming may become less useful or harmful. It is advantageous to parameterize algorithms for tuning for dataset of different sizes.

When performing TM on large comparable texts, we initially filtered the text to produce short comparable text segments and then we performed TM on them. Though the approach is relatively simple, it led to pronounced improvement in TM precision from 0.650 to 0.946, with a drop in recall from 0.500 to 0.417. Given that comparable texts are ubiquitous, improvements in precision are likely more important than drops in recall.

For future work, we want to test the effect of improved TM in the context of different NLP applications such as MT and cross language IR.

Acknowledgements

Some of the experiments were conducted while the authors were at the Cairo Microsoft Innovation Center.

References

C. Bannard, C. Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. ACL-2005, pp. 597—604.

S. Bilac, H. Tanaka. 2005. Extracting transliteration pairs from comparable corpora. NLP-2005.

E. Brill, G. Kacmarcik, C. Brockett. 2001. Automatically harvesting Katakana-English term pairs from search engine query logs. NLPRS 2001, pp. 393–399.

K. Darwish. 2010. Transliteration Mining with Phonetic Conflation and Iterative Training. ACL NEWS workshop 2010.

A. El Kahki, K. Darwish, A. Saad El Din, M. Abd El-Wahab, A. Hefny, W. Ammar. Improved Transliteration Mining Using Graph Reinforcement. EMNLP-2011. pp. 1384—1393.

H. Fei, S. Vogel, A. Waibel. 2003. Extracting Named Entity Translingual Equivalence with Limited Resources. TALIP, 2(2):124–129.

X. He. 2007. Using Word-Dependent Transition Models in HMM based Word Alignment for Statistical Machine Translation. ACL-07 2nd SMT workshop.

S. Hewavitharana, S. Vogel. 2011. Extracting parallel phrases from comparable data. The 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web, June 24-24, 2011, Portland, Oregon

S. Jiampojamarn, K. Dwyer, S. Bergsma, A. Bhargava, Q. Dou, M.Y. Kim and G. Kondrak. 2010. Transliteration Generation and Mining with Limited Training Resources. ACL NEWS workshop 2010.

C. Jin, D.I. Kim, S.H. Na, J.H. Lee. 2008. Automatic Extraction of English-Chinese Transliteration Pairs using Dynamic Window and Tokenizer. Sixth SIGHAN Workshop on Chinese Language Processing, 2008.

A. Klementiev, D. Roth. 2006. Named Entity Transliteration and Discovery from Multilingual Comparable Corpora. HLT Conf. of the North American Chapter of the ACL, pages 82–88.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst (2007). Moses: Open Source Toolkit for Statistical Machine Translation. ACL-2007, Demo Session.

S. Kok, C. Brockett. 2010. Hitting the Right Paraphrases in Good Time. Human Language Technologies: NAACL-2010.

A. Kumaran, M. Khapra, H. Li. 2010. Report of NEWS 2010 Transliteration Mining Shared Task. 2010 Named Entities Workshop, ACL 2010, pages 21–28.

J.S. Kuo, H. Li, Y.K. Yang. 2006. Learning Transliteration Lexicons from the Web. COLING-ACL-2006, 1129 – 1136.

J.S. Kuo, H. Li, Y.K. Yang. 2007. A phonetic similarity model for automatic extraction of transliteration pairs. TALIP, 2007

J.S. Kuo, H. Li, C.L. Lin. 2008. Mining Transliterations from Web Query Results: An Incremental Approach. Sixth SIGHAN Workshop on Chinese Language Processing, 2008.

J.S. Kuo, Y.K. Yang. 2005. Incorporating Pronunciation Variation into Extraction of Transliterated-term Pairs from Web Corpora. Journal of Chinese Language and Computing, 15 (1): (33-44).

C.J. Lee, J.S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. Workshop on Building and Using Parallel Texts, HLT-NAACL-2003, 2003.

D.S. Munteanu, D. Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. ACL-2006, p.81-88.

S. Noeman, A. Madkour. 2010. Language Independent Transliteration Mining System Using Finite State Automata Framework. ACL NEWS workshop 2010.

R. Mahesh, K. Sinha. 2009. Automated Mining Of Names Using Parallel Hindi-English Corpus. 7th Workshop on Asian Language Resources, ACL-IJCNLP 2009, pages 48–54, 2009.

J.H. Oh, K.S. Choi. 2006. Recognizing transliteration equivalents for enriching domain specific thesauri. 3rd Intl. WordNet Conf., pp. 231–237, 2006.

Jong-Hoon Oh, Hitoshi Isahara. 2006. Mining the Web for Transliteration Lexicons: Joint-Validation

Approach. pp.254-261, 2006 IEEE/WIC/ACM Intl. Conf. on Web Intelligence (WI'06), 2006.

Yan Qu, Gregory Grefenstette, David A. Evans. 2003. Automatic transliteration for Japanese-to-English text retrieval. SIGIR 2003:353-360

P. Resnik, N. Smith. 2003. The Web as a parallel corpus. Computational Linguistics - Special issue on web as corpus, Vol. 29 Issue 3, Sept. 2003

K Saravanan, A Kumaran. 2008. Some Experiments in Mining Named Entity Transliteration Pairs from Comparable Corpora. The 2nd Intl. Workshop on Cross Lingual Information Access: Addressing the Need of Multilingual Societies, 2008.

J. Smith, C. Quirk, K. Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment, Human Language Technologies: NAACL-2010, p.403-411.

R. Udupa, K. Saravanan, A. Bakalov, A. Bhole. 2009a. "They Are Out There, If You Know Where to Look": Mining Transliterations of OOV Query Terms for Cross-Language Information Retrieval. ECIR-2009.

R. Udupa, K. Saravanan, A. Kumaran, J. Jagarlamudi. 2009b. MINT: A Method for Effective and Scalable Mining of Named Entity Transliterations from Large Comparable Corpora. EACL 2009.

R. Udupa, M. Khapra. 2010a. Transliteration Equivalence using Canonical Correlation Analysis. ECIR-2010, 2010.

R. Udupa, S. Kumar. 2010b. Hashing-based Approaches to Spelling Correction of Personal Names. EMNLP 2010.

G.W. You, S.W. Hwang, Y.I. Song, L. Jiang, Z. Nie. 2010. Mining Name Translations from Entity Graph Mapping. EMNLP-2010, pp. 430–439.

S. Zhao, H. Wang, T. Liu, S. Li. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. ACL-08: HLT, pp. 780–788.