Language Independent Transliteration Mining System Using Finite State Automata Framework

Sara Noeman and Amgad Madkour
Human Language Technologies Group
IBM Cairo Technology Development Center
P.O.Box 166 El-Haram, Giza, Egypt
{noemans,amadkour}@eg.ibm.com

Abstract

We propose a Named Entities transliteration mining system using Finite State Automata (FSA). We compare the proposed approach with a baseline system that utilizes the Editex technique to measure the length-normalized phonetic based edit distance between the two words. We submitted three standard runs in NEWS2010 shared task and ranked first for English to Arabic (WM-EnAr) and obtained an F-measure of 0.915, 0.903, and 0.874 respectively.

1 Introduction

Named entities transliteration is a crucial task in many domains such as cross lingual information retrieval, machine translation, and other natural language processing applications. In the previous NEWS 2009 transliteration task, we introduced a statistical approach for transliteration generation only using the bilingual resources (about 15k parallel names) provided for the shared task. For NEWS2010, the shared task focuses on acquisition of a reasonably sized, good quality names corpus to complement the machine transliteration task. Specifically, the task focuses on mining the Wikipedia paired entities data (inter-wiki-links) to produce high-quality transliteration data that may be used for transliteration generation tasks.

2 Related Work

Finite state Automata is used to tackle many Natural Language Processing challenges. Hassan (2008) et al. proposed the use of finite state automata for language-independent text correction. It consists of three phases : detecting misspelled words, generating candidate corrections for them and ranking corrections. In detecting the mispelled words, they compose the finite state automaton representation of the dictionary with the input string. Onaizan (2002) et al. proposed the use of probabilistic finite state machines for machine transliteration of names in Arabic text. They used a hybrid approach between phonetic-based and spelling-based models. Malik (2008) et al. proposed a Hindi Urdu machine transliteration system using finite state machines. They introduced UIT (universal intermediate transcription) on the same pair according to thier phonetic properties as a means of representing the language and created finite state transducers to represent them. Sherif (2007) proposed the use of memoryless stochastic transducer for extracting transliteration through word similarity metrics.

Other approaches for transliteration include translation of names through mining or through using machine translation systems resources. Hassan (2007) et al. proposed a framework for extraction of named entity translation pairs. This is done through searching for candidate documents pairs through an information retrieval system and then using a named entity matching system which relies on the length-normalized phonetic based edit distance between the two words. They also use a phrase-based translation tables to measure similarity of extracted named entities. Noeman (2009) also used a phrase based statistical machine translation (PBSMT) approach to create a substring based transliteration system through the generated phrase table, thus creating a language independent approach to transliteration. Other resources have been used to perform transliteration. Chang (2009) et. al proposed the use of a romanization table in conjunction with an unsupervised constraint driven learning algorithm in order to identify transliteration pairs without any labelled data.

3 System architecture

The approach consists of three main phases which are (1) Transliteration model learning, (2) Fi-

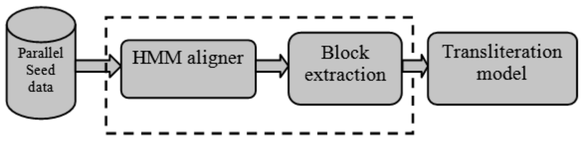

Figure 1: Transliteration table learning in PBSMT

nite State machine formalization of the generated transliteration model and (3) Generating Candidate transliterations. Figure (1) illustrates Transliteration table learning in PBSMT framework. A detailed description of each phase is given in the following sections.

3.1 Transliteration model learning

The objective of NEWS2010 shared task is to develop a system for mining single word transliteration pairs from the standard Wikipedia paired topics (Wikipedia Inter-Language Links, or WIL1), using a seed data of only 1000 parallel names. The aim is to learn one-to-many character sequence mappings on both directions.

We propose the use of MOSES framework[1] for PBSMT training which was applied on the 1k parallel seed data. The proposed approach depends on the formulation of the transliteration problem using the PBSMT approach used in Machine translation. Giza++ Hidden Markov Model (HMM) aligner[2] proposed by Och (1999) was also used over the parallel character sequences. Heuristics were used to extend substring to substring mappings based on character-to-character alignment. This generated a substring to substring translation model such as in Koehn (2003). The phrase "substring" table was filtered out to obtain all possible substrings alignment of each single character in the language alphabet in both directions. This means that for each character in the source language (English) alphabet, substrings mapped to it are filtered with a threshold. Also for each character in the target language (Arabic) alphabet, all English substrings mapped to it are filtered with a threshold. These two one-to-many alignments were intersected in one "Transliteration Arabic-to-English mapping". We obtained a character alignment table which we refer to as "Ar2En list". Figure(2) illustrates a sample one-to-many alignment mapping.

[1]MOSES Framework: http://www.statmt.org/moses/
[2]GIZA++ Aligner: http://fjoch.com/GIZA++.html

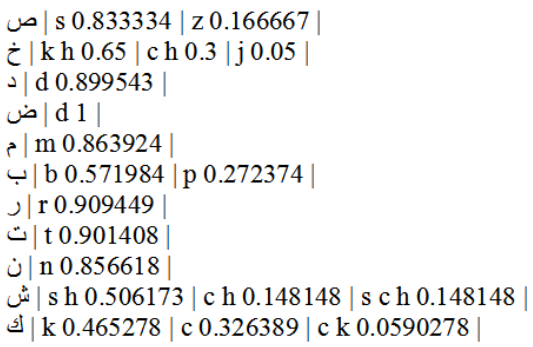
ص | s 0.833334 | z 0.166667 |
خ | k h 0.65 | c h 0.3 | j 0.05 |
د | d 0.899543 |
ض | d 1 |
م | m 0.863924 |
ب | b 0.571984 | p 0.272374 |
ر | r 0.909449 |
ت | t 0.901408 |
ن | n 0.856618 |
ش | s h 0.506173 | c h 0.148148 | s c h 0.148148 |
ك | k 0.465278 | c 0.326389 | c k 0.0590278 |

Figure 2: One to Many Alignment Mapping

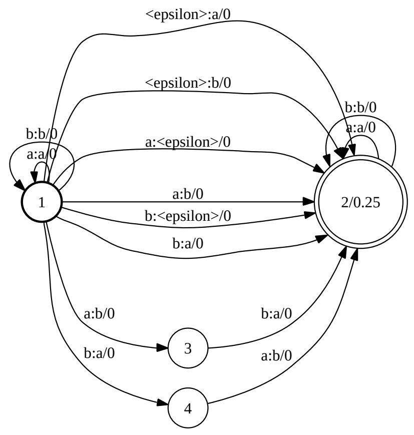

Figure 3: Edit distance 1 FSM

3.2 FSM formalization of Transliteration Model

The proposed method makes use of the finite state automaton representation for the Ar2En character alignment list, where the input is the source character and the output is the target character. We refer to this finite state transducer (FST) as "Ar2En FST". For each source word, we build a Finite State Acceptor (FSA), such that each candidate source word FSA is composed with the "Ar2En FST". For the target words list, we build a finite state acceptor (FSA) that contains a path for each word in the target Wiki-Link.

3.3 Generating Candidate transliterations

The task of generating candidate transliterations at edit distance k from initial source candidate transliterations using Levenshtein transducer can be divided into two sub tasks: Generating a list of words that have edit distance less than or equal k to the input word, and selecting the words inter-

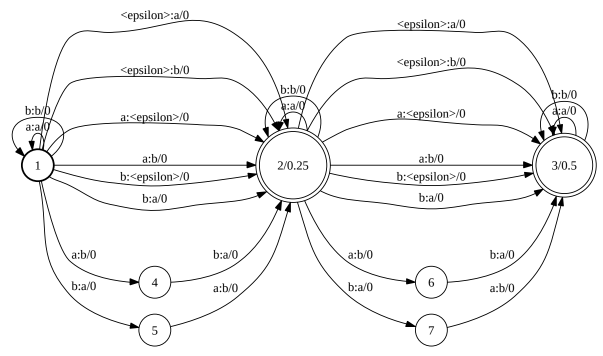

Figure 4: Edit distance 2 FSM

```
For each Wiki-Link-Title
{
   @EnWords = English word list;
   @ArWords_a = Arabic word list;
   Create FSA for English word list "@EnWords" =
FSA[@EnWords];
   Create FSA for Arabic word list "@ArWords" =
FSA[@ArWords];
   Out1_FST = Compose FSA[@ArWords] with FST[Ar2En];
   Out2_FST = Compose Out1_FST with FST[Edit-1];
   Out3_FST = Compose Out2_FST with FSA[@EnWords];
   If(Out3_FST is empty)
   {
             Arabic word @ArWords has no transliteration;
   }
   Else
   {
             Get Mined pair  @ArMatched with @EnMatched
    }
}
```

Figure 5: Using Levenshtein edit-1 FST

secting with the target inter-wiki-link words. This is similar to the spelling correction technique that used FSM which was introduced by Hassan (2008) et. al. In the spelling correction task , after generating the list of words within edit distance *k* to the input word, the system selects a subset of those words that exist in a large dictionary. In order to accomplish this same scenario, we created a single transducer (Levenshtein transducer) that when composed with an FSM representing a word generates all words within an edit distance *k* from the input word. We then compose the resulting FSM with an FSA (finite state acceptor) of all words in the target inter-wiki-link. The Levenshtein transducer is language independent and is built only using the alphabet of the target language. Figure (3) and Figure (4) illustrate the Levenshtein transducer for edit distance 1 and 2 over a limited set of vocabulary (a, b).

4 Data and Resources Processing

After revising the training data (inter-wiki-links) released, we discovered that English and Arabic words contained many stress marks and non normalized characters. We therefore applied normalization on Arabic and English characters to increase source target matching probability, thus increasing the recall of data mining. We also normalized Arabic names, removing all diacritics and kashida. Kashida is a type of justification used in some cursive scripts such as Arabic. Also we normalized Alef () with hamza and madda to go to ”bare Alef”.

5 Standard runs

We submitted 6 runs derived from 3 experiments. For each experiment, we submitted 2 runs, one with normalized Arabic and English characters, and the other with the stress marks and special characters. It is important to note that we run the mining in the Arabic to English direction, thus the Arabic side is the source and the English side is the target.

5.1 Using Levenshtein edit distance 1 FST

Figure (5) illustrates the algorithm used to conduct the first experiment. We subjected all source words to be composed with Levenshtein edit distance 1. For each Wiki-Link, we build a finite state acceptor (FSA) that contains a path for each word in the Arabic Wiki-Link. We refer to it as FSA[@ArWords]. Similarly, for the English name candidates we build a finite state acceptor (FSA) that contains a path for each word in the English Wiki-Link. We refer to it as FSA[@EnWords]. The generated @ArWords and @EnWords are the lists of words in the Arabic and English wiki-links respectively. The result of this experiment was reported as Standard-3 ”normalized characters” and Standard-4 ”without normalized characters”.

5.2 Using Levenshtein up to edit distance 2 FST

Figure (6) illustrates the algorithm used to conduct the second experiment. We use a threshold on the number of characters in a word to decide whether it will be subjected for ”composed with” edit dis-

```
For each Wiki-Link-Title
{
  @EnWords = English word list;
  @ArWords = Arabic word list;
  Create FSA for English word list "@EnWords" =
FSA[@EnWords];
  For each Arabic word "wa" (@ArWords)
  {
         Create FSA [wa];
         Out1_FST = Compose FSA[wa] with FST[Ar2En];
         Where |wa| = length of "wa"
         If( |wa| <= 3 letters)
         {
           Out2_FST = Out1_FST;
         }
         Else if( 3 letters < |wa| <= 7 letters )
         {
           Out2_FST = Compose Out1_FST with FST[Edit-1];
         }
         Else
         {
           Out2_FST = Compose Out1_FST with FST[Edit-2];
         }
         Out3_FST = Compose Out2_FST with FSA[@EnWords];
         If(Out3_FST is empty)
         {
           Arabic word "wa" has no transliteration.
         }
         Else
         {
           Get best path: mined pair "wa" with best path output
         }
    }
  }
```

Figure 6: Using Levenshtein edit-2 FST

tance 0 or 1 or 2. We use a threshold of 3 for edit distance 1 and a threshold of 7 for edit distance 2. The threshold values are set based on our previous experience from dealing with Arabic text and could be derived from the data we obtained. If word length is less than or equal 3 letters, then it is not composed with Levenshtein FSTs, and if word length is between 4 to 7 letters, we compose it with edit distance 1 FST. Longer words are composed with edit distance 2 FST. The result of the experiment was reported in two submitted runs: Standard-5 ”normalized characters” and Standard-6 ”without normalized characters”.

5.3 Baseline

We use a length-normalized phonetic edit distance to measure the phonetic similarity between the source and target Named Entities in the inter-wiki-links. We use the Editex technique Zobel (1996) that makes use of the phonetic characteristics of individual characters to estimate their similarity. Editex measures the phonetic distance between pairs of words by combining the properties of edit distances with a letter grouping strategy that groups letters with similar pronunciations. The result of this experiment was reported in two submitted runs: Standard-1 ”normalized characters” and Standard-2 ”without normalized characters”.

Submission	F-Score	Precision	Recall
Standard-6	***0.915***	0.887	0.945
Standard-4	0.903	0.859	***0.952***
Standard-2	0.874	***0.923***	0.830
Standard-5	0.723	0.701	0.747
Standard-3	0.716	0.681	0.755
Standard-1	0.702	0.741	0.666

Table 1: Shared Task Results

6 Results

Table (1) illustrates the results of the shared task given on the runs we submitted.

Our baseline run (Standard-2) reports highest precision of 0.923 and lowest recall of 0.830 (lowest F-score = 0.874). The reason is that Editex technique measures the edit distance based on letter grouping strategy which groups letters with similar pronunciations. It operates on character to character level. Letters that are mapped to multi-characters will suffer a large edit distance and may exceed the matching threshold used.

The two runs Standard-4 and Standard-6 are implemented using edit-distance FSM matching between source and target. They cover one-to-many character mapping. We notice that Standard-6 run reports higher precision of 0.887 compared to 0.859 for Standard-4 run. This reflects the effect of using variable edit-distance according to the source word length. The Standard-6 reports a Recall of 0.945 producing our best F-Score of 0.915. Standard-6 recall degrades only 0.7% from Standard-4 Recall (0.952).

7 Conclusion

We proposed a language independent transliteration mining system that utilizes finite state automaton. We demonstrated how statistical techniques could be used to build a language independent machine transliteration system through utilizing PBMT techniques. We performed 3 standard experiments each containing two submissions. FSM edit distance matching outperformed Editex in F-Score and Recall. The proposed approach obtained the highest F-Score of 0.915 and a recall of 0.945 in the shared task.

References

Ahmed Hassan, Haytham Fahmy, Hany Hassan 2007. *Improving Named Entity Translation by Exploiting Comparable and Parallel Corpora*. AMML07

Ahmed Hassan, Sara Noeman, Hany Hassan 2008. *Language Independent Text Correction using Finite State Automata*. IJCNLP08.

Franz Josef Och, Christoph Tillmann, and Hermann Ney 1999. *Improved Alignment Models for Statistical Machine Translation*. EMNLP.

Justin Zobel and Philip Dart 1996. *Phonetic string matching: Lessons from information retrieval*. In Proceedings of the Annual ACM References Conference on Research and Development in Information Retrieval (SIGIR).

M. G. Abbas Malik, Christian Boitet, Pushpak Bhattacharyya 2008. *Hindi Urdu Machine Transliteration using Finite-state Transducers*. Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 537544

Ming-Wei Chang, Dan Goldwasser, Dan Roth, Yuancheng Tu 2009. *Unsupervised Constraint Driven Learning For Transliteration Discovery*. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, Daniel Marc 2003. *Statistical Phrase-Based Translation*. Proc. Of the Human Language Technology Conference, HLT-NAACL2003, May.

Sara Noeman 2009. *Language Independent Transliteration system using PBSMT approach on substrings*. Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration.

Tarek Sherif, Grzegorz Kondrak 2007. *Bootstrapping a Stochastic Transducer for Arabic-English Transliteration Extraction*. Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 864871

Yasser Al-Onaizan, Kevin Knight 2002. *Machine Transliteration of Names in Arabic Text*. ACL Workshop on Comp. Approaches to Semitic Languages.